aws

Guía del usuario de Aurora

Amazon Aurora

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Aurora: Guía del usuario de Aurora

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas comerciales que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Aurora? 1
Modelo de responsabilidad compartida de Amazon RDS 2
Cómo funciona Amazon Aurora con Amazon RDS 2
Clústeres de base de datos de Aurora 4
Versiones de Aurora 6
Motores de bases de datos compatibles 6
Especificación de una versión de Aurora 7
Control de versiones de Aurora 7
Actualizaciones de clústeres de base de datos de Aurora 14
Compatibilidad de versiones de Aurora 17
Regiones y zonas de disponibilidad 19
AWSRegiones de 20
Zonas de disponibilidad 30
Zona horaria local para los clústeres de base de datos de 31
Funciones Aurora admitidas por región y motor 38
Convenciones de tabla 39
Implementaciones azul/verde 39
Configuración de los clústeres de Aurora 40
Secuencias de actividades de la base de datos 41
Exportación de datos del clúster a Amazon S3 51
Exportación de datos de instantáneas a Amazon S3 53
Bases de datos globales de Aurora 54
Autenticación de bases de datos de IAM 65
Autenticación de Kerberos 66
Machine Learning de Aurora 78
Información de rendimiento 89
Integraciones sin ETL 100
Proxy RDS 107
Integración de Secrets Manager 120
Aurora Serverless v2 121
Aurora Serverless v1 127
API de datos de RDS 132
Aplicación de parches sin tiempo de inactividad (ZDP) 143
Aurora PostgreSQL Limitless Database 143

Características nativas del motor ... 144
Conexiones de puntos de conexión ... 144
Tipos de puntos de conexión de Aurora ... 145
Visualización de puntos de enlace ... 147
Puntos de enlace y alta disponibilidad ... 148
Puntos de conexión de clúster ... 149
Puntos de enlace del lector ... 150
Puntos de conexión de instancia ... 151
Puntos de conexión personalizados ... 152
Clases de instancia de base de datos ... 173
Tipos de clase de instancia de base de datos ... 173
Motores de bases de datos compatibles ... 177
Determinación de la compatibilidad de la clase de instancia de base de datos en Regiones de AWS ... 194
Especificaciones de hardware ... 199
Almacenamiento ... 213
Información general del almacenamiento de Aurora ... 214
Contenido del volumen del clúster ... 214
Configuraciones de almacenamiento en clústeres de Aurora ... 215
Cómo cambia el tamaño del almacenamiento ... 216
Facturación de datos ... 217
Fiabilidad ... 218
Reparación automática del almacenamiento ... 218
Caché de páginas que puede sobrevivir ... 218
Recuperación de reinicios no planificados ... 219
Seguridad de Aurora ... 220
Uso de SSL con clústeres de base de datos de Aurora ... 222
Alta disponibilidad ... 222
Alta disponibilidad para datos de Aurora ... 223
Alta disponibilidad para instancias de bases de datos de Aurora ... 223
Alta disponibilidad en todas las regiones de AWS con bases de datos de Aurora globales ... 224
Tolerancia a errores ... 224
Alta disponibilidad con Amazon RDS Proxy ... 227
Replicación ... 227
Réplicas de Aurora ... 227
Aurora MySQL ... 230

Aurora PostgreSQL 230
Facturación de instancias de base de datos para Aurora 231
Instancias de base de datos bajo demanda 234
Instancias de base de datos reservadas 236
Configuración del entorno 253
Cómo crear una Cuenta de AWS 253
Creación de un usuario con acceso administrativo 254
Conceder acceso programático 255
Determinar las necesidades 257
Proporcione acceso al clúster de base de datos. 259
Introducción 263
Creación de un clúster de base de datos de Aurora MySQL y conexión a él 263
Requisitos previos 265
Paso 1: crear una instancia de EC2 265
Paso 2: crear un clúster de base de datos de Aurora MySQL 271
(Opcional) Crear una VPC, una instancia de EC2 y un clúster de Aurora MySQL mediante CloudFormation 277
Paso 3: conectarse a un clúster de base de datos de Aurora MySQL 279
Paso 4: eliminar la instancia de EC2 y el clúster de base de datos 282
(Opcional) Eliminar la instancia de EC2 y el clúster de base de datos creados con CloudFormation 283
(Opcional) Conecte el clúster de base de datos a una función de Lambda 284
Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él 284
Requisitos previos 286
Paso 1: crear una instancia de EC2 286
Paso 2: crear un clúster de base de datos de Aurora PostgreSQL 292
(Opcional) Crear una VPC, una instancia EC2 y un clúster de Aurora PostgreSQL mediante CloudFormation 297
Paso 3: conectarse a un clúster de base de datos de Aurora PostgreSQL 299
Paso 4: eliminar la instancia de EC2 y el clúster de base de datos 302
(Opcional) Eliminar la instancia de EC2 y el clúster de base de datos creados con CloudFormation 303
(Opcional) Conecte el clúster de base de datos a una función de Lambda 304
Tutorial: creación de un servidor web y de un clúster de base de datos de Amazon Aurora 305
Lanzamiento de una instancia EC2 para conectarse con el clúster de base de datos 307
Creación de un clúster de base de datos 313

Instalación de un servidor web ... 324
Tutoriales y código de muestra ... 337
Tutoriales en esta guía ... 337
Tutoriales en otras AWS guías ... 338
Tutoriales y código de muestra en GitHub ... 339
Manual de instrucciones de bases de datos de AWS ... 340
Taller de AWS: Amazon Aurora PostgreSQL ... 341
Taller de AWS: Amazon Aurora MySQL ... 343
Cómo utilizar los AWS SDK ... 344
Acceso mediante programación a Amazon Aurora ... 346
Console-to-Code ... 347
Configuración de un clúster de base de datos de Aurora ... 348
Creación de un clúster de base de datos ... 349
Requisitos previos ... 350
Creación de un clúster de base de datos ... 357
Opciones disponibles ... 368
Configuración que no se aplica a los clústeres de base de datos de Aurora ... 392
Configuración que no se aplica a las instancias de base de datos de Aurora ... 393
Creación de recursos con AWS CloudFormation ... 396
Aurora y plantillas de CloudFormation ... 396
Más información sobre CloudFormation ... 396
Conexión a un clúster de base de datos ... 397
Conexión a clústeres de bases de datos Aurora con los controladores de AWS ... 398
Conexión a Aurora MySQL ... 400
Conexión a Aurora PostgreSQL ... 408
Solución de problemas de conexiones ... 410
Grupos de parámetros ... 412
Descripción general de los grupos de parámetros ... 412
Grupos de parámetros de clúster de bases de datos ... 417
Grupos de parámetros de base de datos ... 437
Comparación de grupos de parámetros de la base de datos ... 455
Especificación de parámetros de base de datos ... 455
Migración de datos a un clúster de base de datos ... 461
Aurora MySQL ... 461
Aurora PostgreSQL ... 461
Creación de una caché de ElastiCache desde Amazon RDS ... 462

Información general sobre la creación de cachés de ElastiCache con ajustes del clúster de base de datos de Aurora ... 462
Creación de una caché de ElastiCache con ajustes de un clúster de base de datos de Aurora ... 463
Migración automática de bases de datos de EC2 ... 467
Descripción general ... 467
Requisitos previos ... 468
Limitaciones ... 469
Creación de recursos de IAM ... 470
Configuración de la migración de datos ... 478
Administración de migraciones ... 480
Monitorización ... 482
Tutorial: Creación de un clúster de bases de datos de MySQL con un grupo de parámetros personalizados ... 484
Requisitos previos ... 485
Creación de un grupo de parámetros personalizados ... 485
Modificación de el valor de parámetro ... 486
Creación de clúster de bases de datos de la base de datos ... 486
Administración de un clúster de base de datos de Aurora ... 488
Detención e inicio de un clúster ... 489
Información general de detención e inicio de un clúster ... 489
Limitaciones ... 491
Detención de un clúster de bases de datos ... 491
Mientras un clúster de bases de datos está detenido ... 492
Inicio de un clúster de bases de datos ... 493
Conexión a una instancia de EC2 ... 495
Información general ... 496
Conexión a una instancia de EC2 ... 501
Visualización de los recursos de computación conectados ... 504
Conexión a una instancia de base de datos que ejecuta un motor de base de datos específico ... 505
Conexión de una función de Lambda ... 506
Descripción general ... 508
Conexión de una función de Lambda ... 519
Visualización de los recursos de computación conectados ... 521
Modificación de un clúster de base de datos de Aurora ... 523

Modificación del clúster de base de datos con la consola, CLI y API 523
Modificación de una instancia de base de datos en un clúster de base de datos 526
Cambio de la contraseña del usuario maestro 529
Opciones disponibles 531
Configuración que no se aplica a los clústeres de base de datos de Aurora 569
Configuración que no se aplica a las instancias de base de datos de Aurora 570
Adición de réplicas de Aurora 572
Amazon EC2 Auto Scaling con réplicas de Aurora 579
Cómo añadir una política de escalado automático 586
Cómo editar una política de escalado automático 598
Cómo eliminar una política de escalado automático 601
Administración del rendimiento y el escalado 604
Escalado del almacenamiento 604
Escalado de instancia 611
Escalado de lectura 612
Administración de conexiones 612
Administración de planes de ejecución de consultas 613
Clonación de un volumen de clúster de base de datos de Aurora 614
Información general de la clonación de Aurora 614
Limitaciones de la clonación de Aurora 615
Cómo funciona la clonación de Aurora 617
Creación de un clon de Aurora 620
Clonación entre VPC 631
Clonación entre cuentas 650
Integración con los servicios de AWS 669
Aurora MySQL 669
Aurora PostgreSQL 669
Mantenimiento de un clúster de base de datos de Aurora 671
Descripción general de las actualizaciones de mantenimiento de un clúster de base de datos 671
Vista de mantenimiento pendiente 673
Selección de frecuencia de actualizaciones de mantenimiento de Aurora MySQL 675
Periodo de mantenimiento 676
Aplicación de actualizaciones 683
Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora 685

Actualizaciones del sistema operativo ... 691
Implementación de actualizaciones de AWS Organizations ... 700
Reinicio de un clúster o de una instancia de base de datos de Aurora ... 704
Reinicio de una instancia de base de datos dentro de un clúster de Aurora ... 705
Reinicio de un clúster de Aurora con disponibilidad de lectura ... 706
Reinicio de un clúster de Aurora con disponibilidad de lectura ... 708
Verificación del tiempo de actividad de clústeres e instancias de Aurora ... 709
Ejemplos de operaciones de reinicio de Aurora ... 712
Conmutación por error de un clúster de base de datos de Aurora ... 729
Eliminación de clústeres e instancias de Aurora ... 731
Eliminación de un clúster de base de datos de Aurora ... 731
Protección contra eliminación para clústeres de Aurora ... 740
Eliminación de un clúster detenido de Aurora ... 740
Eliminación de clústeres de Aurora MySQL que son réplicas de lectura ... 740
La instantánea final al eliminar un clúster ... 741
Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora ... 741
Etiquetado de los recursos de Aurora y RDS ... 744
¿Por qué usar etiquetas de RDS? ... 745
Funcionamiento de las etiquetas de RDS ... 746
Prácticas recomendadas ... 749
Copia de etiquetas a instantáneas de clúster de base de datos ... 750
Añadido y eliminación de etiquetas en Amazon RDS ... 751
Tutorial: Uso de etiquetas para especificar qué clústeres de base de datos de Aurora se deben detener ... 756
ARN en Amazon RDS ... 760
Creación de un nombre ARN ... 760
Obtención de un ARN existente ... 768
Actualizaciones de Aurora ... 772
Identificación de su versión de Amazon Aurora ... 773
Soporte extendido de RDS ... 774
Información general del Soporte extendido de RDS ... 775
Precios del Soporte extendido de RDS ... 776
Prevención de cargos del Soporte extendido de RDS ... 777
Versiones con el Soporte extendido de RDS ... 777
Responsabilidades con el Soporte extendido de RDS ... 778
Responsabilidades de Amazon Aurora ... 778

Sus responsabilidades ... 778
Creación de un clúster de base de datos de Aurora o un clúster global ... 779
Comportamiento del Soporte extendido de RDS ... 779
Observaciones sobre el Soporte extendido de RDS ... 780
Creación de un clúster de base de datos de Aurora o un clúster global con Soporte extendido de RDS ... 781
Visualización de la inscripción en el Soporte extendido de RDS ... 782
Visualización de las fechas de soporte ... 785
Restauración de un clúster de base de datos de Aurora o un clúster global ... 787
Comportamiento del Soporte extendido de RDS ... 788
Observaciones sobre el Soporte extendido de RDS ... 788
Restauración de un clúster de base de datos de Aurora o un clúster global con Soporte extendido de RDS ... 789
Uso de las implementaciones azul/verde para actualizar las bases de datos ... 791
Información general de las implementaciones azul/verde ... 792
Disponibilidad en regiones y versiones ... 793
Ventajas ... 793
Flujo de trabajo ... 793
Permitir el acceso ... 800
Limitaciones y consideraciones ... 801
Prácticas recomendadas ... 811
Creación de una implementación azul/verde ... 816
Preparación para una implementación azul/verde ... 816
Especificación de cambios ... 819
Creación de una implementación azul/verde ... 820
Opciones disponibles ... 822
Visualización de una implementación azul/verde ... 824
Cambio de una implementación azul/verde ... 828
Tiempo de espera de la conmutación ... 829
Barreras de protección de la conmutación ... 829
Acciones de conmutación ... 831
Prácticas recomendadas para realizar la conmutación ... 832
Verificación de las métricas de CloudWatch antes de la conmutación ... 833
Monitoreo del retardo de réplica antes de la transición ... 833
Conmutación de una implementación azul/verde ... 834
Después de la conmutación ... 837

Eliminación de una implementación azul/verde 839
Copias de seguridad y restauración de un clúster de base de datos de Aurora 843
Información general de copias de seguridad y restauración 844
Copias de seguridad 844
Intervalo de copia de seguridad 846
Restauración de datos 849
Clonación de base de datos 850
Backtrack 850
Retener copias de seguridad automatizadas 851
Periodo de retención 851
Visualización de copias de seguridad retenidas 852
Costos de retención 852
Limitaciones 853
Eliminación de las copias de seguridad automatizadas retenidas 853
Almacenamiento de copia de seguridad 856
Almacenamiento de copias de seguridad automatizadas 856
Almacenamiento de instantáneas 857
Métricas de CloudWatch para el almacenamiento de copias de seguridad 857
Cálculo del uso del almacenamiento de copias de seguridad 858
Preguntas frecuentes 860
Creación de una instantánea de clúster de base de datos 863
Determinación de si la instantánea está disponible 865
Restauración de una instantánea de clúster de base de datos 866
Grupos de parámetros 867
Grupos de seguridad 867
Consideraciones sobre Aurora 868
Restauración a partir de una instantánea 868
Copia de una instantánea de clúster de base de datos 871
Copia de una instantánea de clúster de base de datos con la Consola de administración de AWS 872
Limitaciones 873
Consideraciones 874
Copia de una instantánea de clúster de base de datos no cifrada 877
Copia de una instantánea de clúster de base de datos cifrada 879
Copia de una instantánea de clúster de base de datos entre cuentas 883
Compartir una instantánea de clúster de base de datos 888

Uso compartido de una instantánea ... 889
Uso compartido de instantáneas públicas ... 893
Cómo compartir instantáneas cifradas ... 895
Cancelación del uso compartido de instantáneas ... 899
Exportación de datos del clúster de base de datos a Amazon S3 ... 901
Consideraciones ... 903
Configuración del acceso a un bucket de S3 ... 905
Creación de tareas de exportación del clúster de base de datos ... 909
Supervisión de exportaciones del clúster de base de datos ... 912
Cancelación de una exportación de un clúster de base de datos ... 915
Resolución de problemas ... 916
Exportación de datos de instantánea del clúster de bases de datos a Amazon S3 ... 919
Consideraciones ... 921
Configuración del acceso a un bucket de S3 ... 934
Creación de tareas de exportación de instantáneas ... 939
Monitoreo de las exportaciones de instantáneas ... 943
Cancelación de una exportación de instantáneas ... 946
Rendimiento de exportación en Aurora MySQL ... 947
Resolución de problemas ... 948
Recuperación a un momento dado ... 951
Restauración de un clúster de base de datos a un momento determinado ... 952
Recuperación a un momento dado de una copia de seguridad automatizada retenida ... 955
Recuperación en un momento dado mediante AWS Backup ... 958
Eliminación de una instantánea de clúster de base de datos ... 964
Eliminación de una instantánea de clúster de base de datos ... 964
Tutorial: Restaurar un clúster de base de datos a partir de una instantánea ... 966
Restaurar un clúster de base de datos mediante la consola ... 967
Restaurar un clúster de base de datos mediante AWS CLI ... 971
Supervisión de métricas en un clúster de bases de datos de Aurora ... 978
Plan de monitoreo ... 978
Referencia de rendimiento ... 979
Directrices de rendimiento ... 979
Herramientas de monitoreo ... 981
Herramientas de monitoreo automatizadas ... 981
Herramientas de monitoreo manuales ... 983
Visualización del estado del clúster ... 985

Visualización de un clúster de base de datos ... 986
Ver el estado del clúster de base de datos ... 992
Visualización del ... 997
Recomendaciones para Amazon Aurora ... 1005
Visualización de las recomendaciones ... 1007
Aplicación de recomendaciones ... 1015
Descarte de recomendaciones ... 1021
Modificación de recomendaciones descartadas a activas ... 1023
Referencia de recomendaciones ... 1024
Consulta de métricas en la consola de Amazon RDS ... 1047
Visualización del panel de Información de rendimiento ... 1049
Elección de la nueva vista de monitorización en la pestaña Monitorización ... 1050
Elección de la nueva vista de monitorización desde la página Información de rendimiento 1052
Creación de un panel personalizado ... 1054
Elección del panel preconfigurado ... 1057
Monitorización de Aurora con CloudWatch ... 1059
Información general de Amazon Aurora y Amazon CloudWatch ... 1060
Ver métricas de CloudWatch en ... 1062
Exportación de las métricas de Información sobre rendimiento a CloudWatch ... 1068
Creación de alarmas de CloudWatch ... 1074
Supervisión con Información sobre las bases de datos ... 1076
Precios ... 1076
Compatibilidad con motor, región y clase de instancia ... 1077
Activación del modo avanzado ... 1081
Activación del modo estándar ... 1085
Supervisión de consultas lentas ... 1091
Consideraciones ... 1093
Supervisión de carga de base de datos con Performance Insights ... 1094
Información general sobre Performance Insights ... 1095
Activación y desactivación de Información de rendimiento ... 1107
Performance Schema para Aurora MySQL ... 1114
Políticas de información sobre rendimiento ... 1120
Análisis de métricas mediante el panel de Información sobre rendimiento ... 1135
Visualización de las recomendaciones proactivas de Información de rendimiento ... 1171
Recuperación de métricas con la API de Información sobre rendimiento ... 1174
Registro de llamadas de Performance Insights mediante el uso de AWS CloudTrail ... 1200

Puntos de conexión de VPC (AWS PrivateLink) 1203
Análisis de rendimiento con DevOps Guru for RDS 1207
Beneficios de DevOps Guru para RDS 1208
Cómo funciona DevOps Guru for RDS 1209
Configuración de DevOps Guru for RDS 1210
Supervisión del sistema operativo con Supervisión mejorada 1219
Descripción general de la supervisión mejorada 1219
Configuración y habilitación del monitoreo mejorado 1221
Visualización de métricas OS en la consola de RDS 1229
Visualización de métricas del sistema operativo mediante CloudWatch Logs 1231
Referencia de métricas de Aurora 1232
Métricas de CloudWatch para Amazon Aurora 1232
Dimensiones de CloudWatch para Aurora 1277
Disponibilidad de métricas de Aurora en la consola de Amazon RDS. 1277
Métricas de Amazon CloudWatch para Información sobre rendimiento 1281
Métricas de contador para Información de rendimiento 1284
Estadísticas de SQL para Performance Insights 1319
Métricas del sistema operativo en Supervisión mejorada 1329
Supervisión de secuencias de actividades de la base de datos 1337
Visualización de los registros, los eventos y los flujos en la consola de Amazon RDS 1338
Supervisión de eventos de Aurora 1343
Información general de los eventos para Aurora 1343
Consulta de eventos de Amazon RDS 1345
Uso de notificaciones de eventos de Amazon RDS 1349
Creación de una regla que se desencadena en función de un evento Amazon Aurora 1376
Categorías y mensajes de eventos de Amazon RDS para Aurora 1380
Supervisión de registros de Aurora 1421
Visualización y descripción de archivos de registro de base de datos 1421
Descarga de un archivo de registro de base de datos 1423
Ver un archivo de registro de base de datos 1424
Publicación en CloudWatch Logs 1426
Lectura del contenido del archivo de registro mediante REST 1429
Archivos de registro de base de datos de MySQL 1430
Archivos de registro de base de datos de PostgreSQL 1441
Supervisión de llamadas a la API de Aurora en CloudTrail 1453
Integración de CloudTrail con Amazon Aurora 1453

Entradas de archivos de registro de Amazon Aurora 1454
Supervisión de Aurora con flujos de actividad de la base de datos 1459
Descripción general 1459
Requisitos previos de red de Aurora MySQL 1464
Inicio de una secuencia de actividades de la base de datos 1466
Obtención del estado del flujo de actividad 1469
Detención de un flujo de actividad de la base de datos 1471
Monitoreo de secuencias de actividades 1472
Ejemplos de políticas de IAM para flujos de actividad 1511
Supervisión de amenazas con GuardDuty RDS Protection 1514
Uso de Aurora MySQL 1516
Información general de Aurora MySQL 1517
Mejoras del rendimiento de Amazon Aurora MySQL 1517
Aurora MySQL y los datos espaciales 1518
Aurora MySQL versión 3 compatible con MySQL 8.0 1519
Aurora MySQL versión 2 compatible con MySQL 5.7 1555
Seguridad con Aurora MySQL 1558
Privilegios de la cuenta de usuario maestro con Aurora MySQL. 1560
Conexiones TLS 1561
Actualización de aplicaciones para nuevos certificados TLS 1570
Determinación de si alguna aplicación se conecta a su clúster de base de datos de MySQL de Aurora mediante TLS 1571
Determinación de si un cliente necesita una verificación de certificados para conectarse .. 1571
Actualización del almacén de confianza de su aplicación 1573
Ejemplo de código Java para el establecimiento de conexiones TLS 1574
Uso de la autenticación Kerberos para Aurora MySQL 1576
Información general de la autenticación Kerberos para Aurora MySQL 1577
Limitaciones 1579
Configuración de la autenticación Kerberos para Aurora MySQL 1580
Conexión a Aurora MySQL con autenticación Kerberos 1591
Administración de un clúster de base de datos en un dominio 1595
Migración de datos a Aurora MySQL 1597
Migración de una base de datos MySQL externa a Aurora MySQL 1602
Migración desde una instancia de base de datos MySQL a Aurora MySQL 1632
Administración de Aurora MySQL 1661
Administración del rendimiento y el escalado para Amazon Aurora MySQL 1661

Búsqueda de datos anteriores de un clúster de base de datos 1673
Pruebas de Amazon Aurora MySQL por medio de consultas de inserción de errores 1698
Modificación de las tablas de Amazon Aurora con DDL rápido 1702
Visualización del estado del volumen para un clúster de base de datos de Aurora 1709
Ajuste de Aurora MySQL 1711
Conceptos esenciales para el ajuste de Aurora MySQL 1711
Ajuste de Aurora MySQL con eventos de espera 1715
Ajustar Aurora MySQL con estados de subprocesos 1774
Ajuste de Aurora MySQL con información proactiva de Amazon DevOps Guru 1782
Consulta paralela de Aurora MySQL 1789
Información general de consultas paralelas 1789
Creación de un clúster de consultas paralelas 1794
Activar y desactivar las consultas en paralelo 1798
Optimización de consultas paralelas 1802
Verificación del uso de consultas paralelas 1807
Monitoreo de Consultas en paralelo 1811
Constructos de SQL para consultas paralelas 1819
Auditoría avanzada con Aurora MySQL 1842
Habilitar la auditoría avanzada 1842
Visualización de registros de auditoría 1846
Detalles de los logs de auditoría 1846
Replicación con Aurora MySQL 1849
Réplicas de Aurora 1849
Opciones de replicación 1851
Rendimiento de replicación 1852
Reinicio sin tiempo de inactividad (ZDR) 1853
Filtros de replicación 1855
Monitoreo de replicación de 1863
Replicación entre regiones 1864
Replicación de registros binarios (binlog) 1882
Replicación basada en GTID 1934
Reenvío de escritura local 1941
Habilitación del reenvío de escritura local 1942
Comprobación de si un clúster de base de datos tiene habilitado el reenvío de escritura ... 1944
Compatibilidad de las aplicaciones con el reenvío de escritura 1945
Niveles de aislamiento para el reenvío de escritura 1947

Coherencia de lectura 1947
Ejecutar instrucciones multiparte con reenvío de escritura 1952
Transacciones con reenvío de escritura 1952
Parámetros de configuración para el reenvío de escritura 1953
Métricas para el reenvío de escritura 1954
Identificación de transacciones y consultas reenviadas 1959
Integración de Aurora MySQL con los servicios de AWS 1962
Autorización a Aurora MySQL a acceder a otros servicios de AWS 1963
Carga de datos desde archivos de texto en Amazon S3 1983
Almacenamiento de datos en archivos de texto en Amazon S3 1998
Invocación de una función Lambda desde Aurora MySQL 2010
Publicación de registros de Aurora MySQL en CloudWatch Logs 2022
Modo lab de Aurora MySQL 2028
Características del modo lab de Aurora 2028
Procedimientos recomendados con Aurora MySQL 2030
Determinar a qué instancia de base de datos está conectado 2031
Prácticas recomendadas para mejorar el rendimiento y la escalabilidad 2031
Prácticas recomendadas para tener un nivel alto de disponibilidad 2042
Recomendaciones para características de MySQL 2044
Evaluación de la utilización de instancias de base de datos para Aurora MySQL con métricas de Amazon CloudWatch 2052
Solución de problemas de rendimiento de Aurora MySQL 2054
Opciones de monitoreo de AWS 2054
Motivos más comunes de los problemas de rendimiento de la base de datos 2055
Solución de problemas de carga de trabajo 2056
Registro de Aurora MySQL 2087
Solución de problemas de conexión a la base de datos 2090
Solución de problemas de rendimiento de consultas 2107
Referencia de Aurora MySQL 2112
Parámetros de configuración 2112
Variables de estado globales 2173
Eventos de espera 2190
Estados de subprocesos 2197
Niveles de aislamiento 2202
Sugerencias 2209
Referencia de procedimientos almacenados 2213

tablas de information_schema 2263
Actualizaciones de Aurora MySQL 2272
Comprobación de los números de versión 2273
Versiones beta y de soporte a largo plazo 2275
Preparación para el final de la vida útil de la versión 2 de Aurora MySQL 2278
Preparación para el final de la vida útil de la versión 1 de Aurora MySQL 2283
Actualización de clústeres de base Amazon Aurora MySQL 2287
Actualizaciones y correcciones de motor de base de datos de Amazon Aurora MySQL 2420
Uso de Aurora PostgreSQL 2421
El entorno de vista previa de la base 2423
Tipos de clases de instancia de base de datos compatibles 2423
Características no admitidas en el entorno de vista previa 2424
Creación de un nuevo clúster de base de datos en el entorno de vista previa 2425
Versión 17 de PostgreSQL en el entorno de vista previa de bases de datos 2427
Seguridad con Aurora PostgreSQL 2428
Descripción de los roles y permisos de PostgreSQL 2430
Protección de los datos de Aurora PostgreSQL con SSL/TLS 2447
Enmascaramiento dinámico de Aurora PostgreSQL 2461
Actualización de aplicaciones para nuevos certificados SSL/TLS 2499
Determinación de si las aplicaciones se conectan a sus clústeres de base de datos de PostgreSQL de Aurora mediante SSL 2500
Determinación de si un cliente necesita una verificación de certificados para conectarse .. 2501
Actualización del almacén de confianza de su aplicación 2501
Uso de conexiones SSL/TLS para diferentes tipos de aplicaciones 2502
Uso de la autenticación Kerberos 2503
Disponibilidad en regiones y versiones 2504
Información general de la autenticación Kerberos 2505
Configuración 2506
Administración de un clúster de base de datos de Aurora PostgreSQL en un dominio de Active Directory 2522
Conexión con autenticación Kerberos 2523
Uso de grupos de seguridad de AD para el control de acceso de Aurora PostgreSQL 2527
Migración de datos a Aurora PostgreSQL 2539
Migración de una instancia de base de datos de RDS for PostgreSQL mediante una instantánea 2541

Migración de una instancia de base de datos de RDS for PostgreSQL mediante una réplica de lectura de Aurora ... 2549
Optimización del rendimiento de las consultas en Aurora PostgreSQL ... 2565
Mejora del rendimiento de las consultas con las lecturas optimizadas de Aurora ... 2565
Optimización de subconsultas correlacionadas en Aurora PostgreSQL ... 2573
Mejora del rendimiento de las consultas mediante la unión adaptativa ... 2580
Trabajo con tablas no registradas en Aurora PostgreSQL ... 2583
Creación de tablas no registradas ... 2583
Gestión de tablas no registradas durante la migración ... 2584
Conversión de tablas no registradas en tablas registradas ... 2584
Tablas no registradas y replicación lógica ... 2585
Trabajo con autovacuum de PostgreSQL ... 2585
Asignación de memoria para autovacuum ... 2586
Reducción de la probabilidad de reinicio del identificador de transacción ... 2588
Determinar si las tablas de una base de datos necesitan vacío ... 2589
Determinar qué tablas cumplen actualmente los requisitos de autovacuum ... 2591
Determinar si autovacuum se está ejecutando actualmente y durante cuánto tiempo ... 2592
Realización de una inmovilización de vacío manual ... 2593
Reindexar una tabla cuando autovacuum se está ejecutando ... 2596
Administración de autovacuum con índices de gran tamaño ... 2597
Otros parámetros que afectan a autovacuum ... 2600
Establecimiento de parámetros autovacuum de nivel de tabla ... 2601
Registro de actividades de autovacuum y vacuum ... 2602
Comportamiento de autovacuum con bases de datos no válidas ... 2603
Identificación de los bloqueadores de limpieza ... 2606
Administración de la contención de TOAST OID ... 2631
Descripción de las operaciones de TOAST ... 2631
Identificación de los desafíos de rendimiento ... 2632
Recomendaciones ... 2634
Monitorización ... 2635
Uso de Babelfish para Aurora PostgreSQL ... 2640
Limitaciones de Babelfish ... 2642
Descripción de la arquitectura y configuración de Babelfish ... 2643
Creación de un clúster de base de datos de Babelfish para Aurora PostgreSQL ... 2691
Migración de una base de datos SQL Server a Babelfish ... 2702
Autenticación de bases de datos con Babelfish para Aurora PostgreSQL ... 2714

Conexión a un clúster de base de datos de Babelfish ... 2734
Uso de Babelfish ... 2747
Solución de problemas de Babelfish ... 2831
Desactivación de Babelfish ... 2833
Administración de las actualizaciones de versiones de Babelfish ... 2834
Referencia de Babelfish ... 2857
Rendimiento y escalado para Aurora PostgreSQL ... 2916
Escalado de las instancias de base de datos Aurora PostgreSQL ... 2917
Número máximo de conexiones ... 2918
Límites de almacenamiento temporal ... 2919
Páginas enormes para Aurora PostgreSQL ... 2924
Pruebas de Amazon Aurora PostgreSQL mediante consultas de inserción de errores ... 2924
Visualización del estado del volumen para un clúster de bases de datos de Aurora ... 2930
Especificación del disco RAM para stats_temp_directory ... 2931
Administración de archivos temporales con PostgreSQL ... 2932
Ajuste con eventos de espera de Aurora PostgreSQL ... 2939
Conceptos esenciales para el ajuste de Aurora PostgreSQL ... 2940
Eventos de espera de Aurora PostgreSQL ... 2946
Client:ClientRead ... 2948
Client:ClientWrite ... 2952
CPU ... 2954
IO:BufFileRead y IO:BufFileWrite ... 2961
IO:DataFileRead ... 2970
IO:XactSync ... 2981
IPC:DamRecordTxAck ... 2984
Eventos de espera IPC:parallel ... 2985
IPC:ProcArrayGroupUpdate ... 2992
Lock:advisory ... 2994
Lock:extend ... 2997
Lock:Relation ... 3000
Lock:transactionid ... 3007
Lock:tuple ... 3010
LWLock:buffer_content (BufferContent) ... 3015
LWLock:buffer_mapping ... 3017
LWLock:BufferIO (IPC:BufferIO) ... 3020
LWLock:lock_manager ... 3022

LWLock:MultiXact ... 3026
LWLock:pg_stat_statements ... 3031
Timeout:PgSleep ... 3035
Ajuste de Aurora PostgreSQL con información proactiva de Amazon DevOps Guru ... 3036
La base de datos lleva mucho tiempo inactiva en la conexión de la transacción ... 3036
Prácticas recomendadas con Aurora PostgreSQL ... 3040
Prevención del rendimiento lento, el reinicio automático y la conmutación por error de las instancias de base de datos Aurora PostgreSQL ... 3041
Diagnóstico de sobrecarga de tablas e índices ... 3041
Administración de memoria mejorada en Aurora PostgreSQL ... 3045
Conmutación por error rápida ... 3048
Recuperación rápida después de una conmutación por error ... 3060
Administración de la pérdida de conexión ... 3067
Gestión de conexiones inactivas en PostgreSQL ... 3076
Configuración de los parámetros de memoria para Aurora PostgreSQL ... 3079
Analice el uso de recursos con métricas de CloudWatch ... 3088
Uso de la replicación lógica para realizar una actualización de la versión principal ... 3092
Solución de problemas de almacenamiento en Aurora PostgreSQL ... 3102
Replicación con Aurora PostgreSQL ... 3103
Réplicas de Aurora ... 3104
Mejora de la disponibilidad de lectura de las réplicas de Aurora ... 3105
Monitoreo de replicación de ... 3107
Información general sobre la replicación lógica ... 3108
Configuración de la replicación lógica ... 3109
Desactivación de la replicación lógica ... 3112
Supervisión de la memoria caché de escritura indirecta y de las ranuras lógicas ... 3112
Ejemplo: uso de la replicación lógica ... 3115
Ejemplo: replicación lógica mediante Aurora PostgreSQL y AWS DMS ... 3117
Configuración de la autenticación de IAM para conexiones de replicación lógica ... 3120
Reenvío de escritura local en Aurora PostgreSQL ... 3123
Limitaciones y consideraciones del reenvío de escritura local en Aurora PostgreSQL ... 3124
Configuración de Aurora PostgreSQL para el reenvío de escritura local ... 3126
Procedimiento del reenvío de escritura local en Aurora PostgreSQL ... 3129
Supervisión del reenvío de escritura local en Aurora PostgreSQL ... 3133
Uso de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock ... 3140
Requisitos previos ... 3140

Preparación de Aurora PostgreSQL como base de conocimientos 3141
Creación de una base de conocimiento en la consola de Bedrock 3144
Creación rápida de una base de conocimiento de Amazon Bedrock para Aurora PostgreSQL 3146
Integración de Aurora PostgreSQL con los servicios de AWS 3149
Importación de datos de Amazon S3 a RDS para Aurora PostgreSQL 3150
Exportación de datos de PostgreSQL a Amazon S3 3170
Invocación de una función Lambda desde Aurora PostgreSQL 3188
Publicación de registros de Aurora PostgreSQL en CloudWatch Logs 3204
Monitorización de planes de ejecución de consultas y máximo de memoria para Aurora PostgreSQL 3217
Acceso a los planes de ejecución de consultas y máximo de memoria mediante las funciones de Aurora 3218
Referencia de parámetros para los planes de ejecución de consultas de Aurora PostgreSQL 3219
Administración de planes de ejecución de consultas para Aurora PostgreSQL 3224
Descripción general de la administración de planes de consultas en Aurora PostgreSQL .. 3225
Prácticas recomendadas para la administración de planes de consultas de Aurora PostgreSQL 3234
Administración de planes de consultas 3237
Captura de planes de ejecución de Aurora PostgreSQL 3239
Uso de los planes administrados de Aurora PostgreSQL 3242
Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans 3247
Mejora de los planes 3248
Eliminación de planes 3252
Importación y exportación de planes administrados 3254
Referencia de parámetros 3256
Referencia de funciones 3263
Referencia de la vista apg_plan_mgmt.dba_plans 3275
Funciones avanzadas de la administración de planes de consultas 3280
Uso de extensiones y contenedores de datos externos 3295
Uso de la compatibilidad de extensiones delegadas de Amazon Aurora para PostgreSQL 3296
Administración de objetos grandes de forma más eficiente con el módulo lo 3310
Administración de datos espaciales con PostGIS 3314
Administración de las particiones con la extensión pg_partman 3324
Programación de mantenimiento con la extensión pg_cron 3330

Uso de pgAudit para registrar la actividad de la base de datos ... 3340
Uso de pglogical para sincronizar datos ... 3354
Contenedores de datos externos compatibles en Amazon Aurora PostgreSQL ... 3369
Uso de Extensiones de lenguaje de confianza para PostgreSQL ... 3385
Terminología ... 3386
Requisitos para usar Extensiones de lenguaje de confianza ... 3387
Configuración de Extensiones de lenguaje de confianza ... 3390
Información general de Extensiones de lenguaje de confianza ... 3394
Creación de extensiones TLE ... 3396
Eliminar las extensiones TLE de una base de datos ... 3401
Desinstalación de Extensiones de lenguaje de confianza ... 3402
Uso de enlaces de PostgreSQL con sus extensiones TLE ... 3403
Referencia de funciones para Extensiones de lenguaje de confianza ... 3410
Referencia de enlaces para Extensiones de lenguaje de confianza ... 3424
Referencia de Aurora PostgreSQL ... 3427
Intercalaciones de Aurora PostgreSQL para EBCDIC y otras migraciones de mainframe ... 3427
Intercalaciones admitidas en Aurora PostgreSQL ... 3429
Referencia de las funciones de Aurora PostgreSQL ... 3430
Parámetros de Aurora PostgreSQL ... 3489
Eventos de espera de Aurora PostgreSQL ... 3556
Actualizaciones de Aurora PostgreSQL ... 3587
Identificación de las versiones de Amazon Aurora PostgreSQL ... 3587
Versiones de Aurora PostgreSQL ... 3589
Versiones de extensión para Aurora PostgreSQL ... 3590
Actualización de clústeres de base de datos PostgreSQL de Amazon Aurora ... 3590
Uso de una versión de soporte a largo plazo (LTS) ... 3631
Uso de Base de datos ilimitada de Aurora PostgreSQL ... 3634
Arquitectura de Base de datos ilimitada ... 3635
Términos clave ... 3637
Tipos de tabla ... 3639
Facturación ... 3639
Introducción a Base de datos ilimitada ... 3640
Requisitos y consideraciones sobre Base de datos ilimitada de Aurora PostgreSQL ... 3641
Requisitos ... 3641
Consideraciones ... 3642
Características no admitidas ... 3645

Requisitos previos para usar Base de datos ilimitada 3646
Activación de las operaciones de grupos de particiones de base de datos 3646
Creación de un clúster de base de datos que utiliza Base de datos ilimitada 3648
Correlación de la capacidad máxima con los enrutadores y las particiones 3648
Creación de un clúster de base de datos 3652
Uso de los grupos de particiones de base de datos 3662
Conexión a su clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL 3663
Búsqueda del número de enrutadores y particiones en un grupo de particiones de base de datos 3664
Descripción de los grupos de particiones de base de datos 3664
Reinicio de un grupo de particiones de base de datos 3665
Modificación de la capacidad de un grupo de particiones de base de datos 3666
División de una partición 3668
Adición de un enrutador 3674
Eliminación de un grupo de particiones de base de datos 3678
Adición de un grupo de particiones de base de datos a un clúster de base de datos de Base de datos ilimitada existente 3679
Creación de tablas de Base de datos ilimitada 3685
Creación de tablas ilimitadas con variables 3686
Conversión de tablas estándar en tablas ilimitadas 3692
Esquemas de ejemplo 3697
Carga de datos en Base de datos ilimitada 3698
Uso del comando COPY con Base de datos ilimitada 3700
Uso de la utilidad de carga de datos de Base de datos ilimitada 3702
Consulta de Base de datos ilimitada 3732
Consultas de una sola partición 3733
Consultas distribuidas 3743
Seguimiento de consultas distribuidas 3744
Interbloqueos distribuidos 3753
Administración de Base de datos ilimitada 3756
Base de datos y consideraciones de tamaño de tabla 3756
Recuperación de espacio de almacenamiento mediante el vaciado 3757
Supervisión de Base de datos ilimitada 3762
Supervisión de Base de datos ilimitada con CloudWatch 3763
Supervisión de base de datos ilimitada con información de base de datos 3769

Supervisión de Base de datos ilimitada con Registros de CloudWatch 3782
Supervisión de Base de datos ilimitada con Monitorización mejorada 3783
Supervisión de Base de datos ilimitada con Información de rendimiento 3784
Supervisión de Base de datos ilimitada con GuardDuty para protección de RDS 3793
Funciones y vistas de Base de datos ilimitada 3794
Eventos de espera de Base de datos ilimitada 3820
Copia de seguridad y restauración de Base de datos ilimitada 3833
Copia de seguridad de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL. 3833
Restauración de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL 3836
No se admiten las utilidades de copia de seguridad y restauración de PostgreSQL 3839
Actualización de Base de datos ilimitada 3840
Actualización de los clústeres de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL 3840
Referencia sobre Base de datos ilimitada 3842
Comandos DDL 3842
Limitaciones e información sobre el DDL 3851
Comandos DML 3891
Limitaciones e información sobre el DML 3894
Variables 3898
Parámetros de clúster de base de datos 3899
Uso de una base de datos global de Aurora 3900
Información general sobre la base de datos global de Aurora 3901
Ventajas de la base de datos global de Amazon Aurora 3902
Disponibilidad en regiones y versiones 3903
Limitaciones de la base de datos global de Aurora 3903
Introducción a la base de datos global de Aurora 3906
Requisitos de configuración 3907
Creación de una base de datos global 3909
Adición de un clúster secundario 3923
Creación de un clúster secundario sin pantalla 3929
Creación de una base de datos global a partir de una instantánea 3932
Administración de una base de datos global de Aurora 3933
Modificación de una base de datos global de Aurora 3934
Modificación de parámetros de base de datos 3936

Eliminación de un clúster de una base de datos global de Aurora ... 3937
Eliminación de una base de datos global de Aurora ... 3940
Etiquetado para la base de datos global de Aurora ... 3942
Conexión a la base de datos global de Aurora ... 3947
Elección del punto de conexión que se adapte a las necesidades de su aplicación ... 3948
Visualización de los puntos de conexión de una base de datos global ... 3949
Consideraciones sobre el uso de los puntos de conexión del escritor global ... 3953
Uso del reenvío de escritura en una base de datos Aurora global ... 3954
Uso del reenvío de escritura en Aurora MySQL ... 3954
Uso del reenvío de escritura en Aurora PostgreSQL ... 3977
Transición o conmutación por error en la base de datos global de Aurora ... 3993
Planificación de la BCDR ... 3994
Realización de la transición ... 3995
Recuperación de una interrupción no planificada ... 4002
Administración de RPO (Aurora PostgreSQL) ... 4013
Resiliencia entre regiones para clústeres secundarios ... 4019
Monitoreo de una base de datos global de Aurora ... 4020
Supervisión de una base de datos Aurora global con la información sobre rendimiento ... 4022
Supervisión de bases de datos globales Aurora mediante las secuencias de actividad de base de datos ... 4022
Supervisión de las bases de datos globales basadas en Aurora MySQL ... 4023
Supervisión de bases de datos globales basadas en Aurora PostgreSQL ... 4026
Uso de las implementaciones azul/verde para base de datos global de Aurora ... 4030
Beneficios de usar implementaciones azul/verde con base de datos global de Aurora ... 4030
Como funcionan las implementaciones azul/verde con la base de datos global de Aurora ... 4031
Uso de las bases de datos globales de Aurora con otros servicios de AWS ... 4034
Actualización de una base de datos global de Amazon Aurora ... 4036
Actualizaciones de la versión principal ... 4036
Actualizaciones de la versión secundaria ... 4037
Amazon RDS Proxy ... 4041
Disponibilidad en regiones y versiones ... 4043
Cuotas y limitaciones ... 4043
Limitaciones de MySQL ... 4045
Limitaciones de PostgreSQL ... 4046
Planificación del lugar de uso de RDS Proxy ... 4047
Conceptos y terminología de RDS Proxy ... 4049

Información general de los conceptos de RDS Proxy 4050
Grupo de conexiones 4051
Seguridad 4051
Conmutación por error 4054
Transacciones 4056
Introducción al proxy de RDS 4056
Configuración de una red proxy 4057
Configuración de credenciales de base de datos 4062
Configuración de la autenticación de IAM 4066
Creación de un proxy 4073
Visualización de un proxy 4086
Conexión a través de RDS Proxy 4088
Administración de un RDS Proxy 4092
Modificación de un RDS Proxy 4092
Agregar un usuario de base de datos 4100
Cambio a autenticación integral de IAM 4102
Observaciones sobre la conexión de RDS Proxy 4105
Cómo evitar la fijación de RDS Proxy 4110
Eliminación de un RDS Proxy 4115
Trabajo con puntos de enlace del proxy de RDS 4116
Información general de los puntos de enlace de proxy 4116
Limitaciones para los puntos de conexión de proxy 4118
Usar puntos de enlace del lector con los clústeres de Aurora 4118
Acceso a las bases de datos de Aurora en todas las VPC 4123
Creación de un punto de enlace de proxy 4125
Visualización de puntos de enlace de proxy 4128
Modificación de un punto de enlace de proxy 4129
Eliminación de un punto de enlace de proxy 4131
Supervisión de RDS Proxy con CloudWatch 4132
Trabajo con eventos de RDS Proxy 4141
Eventos de RDS Proxy 4142
Ejemplos del RDS Proxy 4145
Solución de problemas de RDS Proxy 4148
Verificación de la conectividad para un proxy 4148
Problemas y soluciones comunes 4151
Solución de problemas de RDS para MySQL 4152

Solución de problemas de RDS para PostgreSQL 4154
Uso del proxy de RDS con AWS CloudFormation 4161
Uso de RDS Proxy con bases de datos globales de Aurora 4162
Limitaciones de RDS Proxy con bases de datos globales 4162
Cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales .. 4162
Integraciones sin ETL 4164
Ventajas 4166
Conceptos clave 4166
Limitaciones 4167
Limitaciones generales 4167
Limitaciones de Aurora MySQL 4168
Limitaciones de Aurora PostgreSQL 4169
Limitaciones de Amazon Redshift 4170
Amazon SageMaker AILimitaciones de Lakehouse 4170
Cuotas 4170
Regiones admitidas 4171
Introducción a las integraciones sin ETL 4171
Crear un grupo de parámetros de clúster de base de datos personalizado 4172
Paso 2: seleccionar o crear un clúster de base de datos de origen 4173
Paso 3a: crear un almacén de datos de destino 4174
Configuración de una integración mediante los AWS SDK 4175
Paso 3b: creación de un catálogo de AWS Glue para una integración sin ETL de Amazon SageMaker AI 4185
Siguientes pasos 4189
Creación de integraciones sin ETL con Amazon Redshift 4189
Requisitos previos 4189
Permisos necesarios 4189
Creación de integraciones sin ETL 4192
Cifrado de integraciones 4197
Siguientes pasos 4199
Creación de integraciones sin ETL con un Amazon SageMaker Lakehouse 4199
Requisitos previos 4199
Permisos necesarios 4200
Creación de integraciones sin ETL con un Amazon SageMaker Lakehouse 4203
Cifrado de integraciones 4208
Siguientes pasos 4210

Filtrado de datos para integraciones sin ETL ... 4210
Formato de un filtro de datos ... 4211
Lógica de filtros ... 4214
Prioridad del filtro ... 4215
Ejemplos de Aurora MySQL ... 4215
Ejemplos de Aurora PostgreSQL ... 4216
Adición de filtros de datos ... 4217
Eliminación de filtros de datos ... 4219
Añadir y consultar datos ... 4219
Creación de bases de datos de destino en Amazon Redshift ... 4220
Añadir datos al clúster de base de datos de origen ... 4220
Consulta de los datos de en Amazon Redshift ... 4222
Diferencias de tipos de datos ... 4223
Visualización y supervisión de integraciones sin ETL ... 4231
Visualización de las integraciones ... 4232
Monitorización mediante tablas del sistema ... 4233
Monitoreo con EventBridge ... 4234
Modificación de integraciones sin ETL ... 4234
Eliminación de las integraciones sin ETL ... 4236
Solución de problemas de integración sin ETL ... 4237
No puedo crear una integración sin ETL ... 4237
Mi integración está atascada en un estado de `Syncing` ... 4238
Mis tablas no se replican en Amazon Redshift ... 4238
Una o más de mis tablas de Amazon Redshift requieren una resincronización ... 4239
Uso de Aurora Serverless v2 ... 4243
Casos de uso de Aurora Serverless v2 ... 4243
Conversión de cargas de trabajo aprovisionadas ... 4246
Ventajas de Aurora Serverless v2 ... 4246
Cómo funciona Aurora Serverless v2 ... 4248
Descripción general ... 4248
Configuraciones de clúster ... 4250
Capacidad ... 4251
Escalado ... 4253
Alta disponibilidad ... 4256
Almacenamiento ... 4257
Parámetros de configuración ... 4257

Requisitos y limitaciones para Aurora Serverless v2 4258
Disponibilidad en regiones y versiones 4258
Los clústeres que utilizan Aurora Serverless v2 deben tener un rango de capacidad especificado 4259
Algunas características aprovisionadas no se admiten en Aurora Serverless v2 4260
Algunos aspectos de Aurora Serverless v2 son diferentes en Aurora Serverless v1 4260
Creación de un clúster de bases de datos de Aurora Serverless v2 4261
Configuración 4261
Creación de un clúster de bases de datos de Aurora Serverless v2 4263
Creación de un escritor Aurora Serverless v2 4268
Administrar Aurora Serverless v2 4269
Configuración del rango de capacidad de Aurora Serverless v2 para un clúster 4270
Comprobación del rango de capacidad de Aurora Serverless v2 4275
Adición de un lector Aurora Serverless v2 4277
Conversión de aprovisionadas a Aurora Serverless v2 4278
Conversión de Aurora Serverless v2 a aprovisionada 4279
Elegir el nivel de promoción para un lector Aurora Serverless v2 4280
Uso de TLS/SSL con Aurora Serverless v2 4282
Visualización de instancias de Aurora Serverless v2 de escritura y lectura 4283
Registros en Aurora Serverless v2 4285
Rendimiento y escalado para Aurora Serverless v2 4289
Elegir el rango de capacidad 4290
Trabajo con los grupos de parámetros para Aurora Serverless v2 4305
Evitar errores de memoria insuficiente 4311
Métricas importantes de CloudWatch 4312
Supervisión del rendimiento de Aurora Serverless v2 con la información de rendimiento ... 4317
Solución de problemas de capacidad de Aurora Serverless v2 4318
Escalado a 0 ACU con pausa automática y reanudación 4320
Descripción general de la pausa automática 4321
Requisitos previos y limitaciones 4322
Activación y desactivación de la pausa automática 4323
Funcionamiento de la pausa automática 4327
Configuraciones de los clústeres de Aurora y de la pausa automática 4332
Supervisión de la pausa automática 4336
Solución de problemas para la característica de pausa automática 4340
Diseño de la aplicación para la pausa automática 4342

Migración a Aurora Serverless v2 4343
Uso de un clúster de Aurora Serverless v2 existente 4344
Cambiar de un clúster aprovisionado 4348
Comparación de Aurora Serverless v2 y Aurora Serverless v1 4354
Actualización de Aurora Serverless v1 a Aurora Serverless v2 4366
Migración de una base de datos en las instalaciones a Aurora Serverless v2 4369
Uso Aurora Serverless v1 4371
Disponibilidad de regiones y versiones para Aurora Serverless v1 4372
Ventajas de Aurora Serverless v1 4372
Casos de uso de Aurora Serverless v1 4373
Limitaciones de Aurora Serverless v1 4373
Requisitos de configuración para Aurora Serverless v1 4376
Uso de TLS/SSL con Aurora Serverless v1 4377
Conjuntos de cifrado compatibles para conexiones a clústeres de base de datos de Aurora Serverless v1 4380
Cómo funciona Aurora Serverless v1 4380
Aurora Serverless v1 architecture 4381
Autoescalado 4383
Acción de tiempo de espera 4384
Pausar y reanudar 4386
Determinación de max_connections 4387
Grupos de parámetros 4389
Registro 4393
Mantenimiento 4397
Conmutación por error 4398
Instantáneas 4399
Creación de un clúster de bases de datos de Aurora Serverless v1 4399
Restauración de un clúster de bases de datos de Aurora Serverless v1 4403
Modificación de un clúster de bases de datos de Aurora Serverless v1 4407
Modificación de la configuración de escalado 4408
Actualización de la versión principal 4410
Conversión de Aurora Serverless v1 a aprovisionada 4413
Consideraciones a la hora de convertir un clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado 4414
Escalado manual de la capacidad del clúster de bases de datos de Aurora Serverless v1 4416
Visualización de los clústeres de base de datos de Aurora Serverless v1 4419

Monitoreo de los clústers de base de datos de Aurora Serverless v1 con CloudWatch 4423
Eliminación de un clúster de bases de datos de Aurora Serverless v1 4424
Aurora Serverless v1 y versiones del motor de base de datos de Aurora 4427
Aurora MySQL Serverless 4428
Aurora PostgreSQL Serverless 4429
Actualizaciones de versiones secundarias automáticas 4429
Uso de la API de datos de Amazon RDS 4430
Disponibilidad de regiones y versiones para la API de datos de Amazon RDS 4431
Compatibilidad con IPv6 4432
Compatibilidad de puntos de conexión de IPv6 4432
Uso de puntos de conexión de IPv6 4433
Uso de AWS PrivateLink con IPv6 4435
Consideraciones sobre la migración 4436
Solución de problemas de conectividad de IPv6 4436
Limitaciones 4437
API de datos con Aurora Serverless v2 en comparación con Aurora Serverless v1 4438
Número máximo de solicitudes por segundo 4438
Habilitación o desactivación de la API de datos de Amazon RDS en una base de datos existente 4438
Eventos de CloudTrail 4439
Compatibilidad con instrucciones múltiples 4439
Solicitudes simultáneas para el mismo ID de transacción 4440
Comportamiento de BatchExecuteStatement 4442
Comportamiento de ExecuteSQL 4442
Comportamiento de ExecuteStatement 4442
Comportamiento del parámetro de esquema 4443
Autenticación y autorización 4444
Autorización basada en etiquetas 4445
Almacenamiento de credenciales en un secreto de Secrets Manager 4447
Habilitación de la API de datos 4448
Habilitación de la API de datos de RDS al crear una base de datos 4449
Habilitación de la API de datos de RDS en una base de datos existente 4450
Creación de un punto de conexión de VPC de Amazon (PrivateLink) 4453
Llamadas a la API de datos 4457
Referencia de operaciones 4457
Llamadas desde AWS CLI 4462

Llamadas desde aplicaciones Python ... 4473
Llamadas desde aplicaciones Java ... 4477
Control del comportamiento del tiempo de espera ... 4482
Biblioteca de cliente de Java ... 4484
Descarga de la biblioteca de cliente de Java para la API de datos ... 4484
Ejemplos de la biblioteca de cliente de Java ... 4484
Procesamiento de resultados de consultas en JSON ... 4486
Recuperación de resultados de consultas en formato JSON ... 4487
Asignación de tipos de datos ... 4487
Solución de problemas ... 4488
Ejemplos ... 4489
Solución de problemas de la API de datos ... 4494
No se ha encontrado la transacción <transaction_ID> ... 4494
El paquete de la consulta es demasiado grande ... 4495
Límite de tamaño superado de respuesta de base de datos ... 4495
HttpEndpoint no está habilitado para el clúster <clúster_ID> ... 4496
DatabaseErrorException: Transaction is still running a query ... 4496
Excepción de resultado no compatible ... 4497
Las instrucciones múltiples no son compatibles ... 4497
El parámetro de esquema no es compatible ... 4497
Problemas de conectividad de IPv6 ... 4497
Registro de llamadas a la API de datos ... 4498
Trabajar con información de API de datos en CloudTrail ... 4499
Inclusión y exclusión de eventos de la API de datos de un seguimiento de CloudTrail ... 4500
Descripción de las entradas del archivo de registro de la API de datos ... 4502
Supervisión de consultas de la API de datos ... 4505
Cómo se representan las consultas de la API de datos de RDS en Información de rendimiento ... 4505
Uso del editor de consultas ... 4506
Disponibilidad del editor de consultas ... 4506
Autorización del acceso ... 4506
Ejecución de consultas ... 4508
Referencia de API de DBQMS ... 4512
CreateAvoriteQuery ... 4513
CreateQueryHistory ... 4513
CreateTab ... 4513

DeleteFavoritequeries ... 4513
DeletEqueryHistory ... 4513
Deletetab ... 4513
DescribeFavoritequeries ... 4514
DescribeQueryHistory ... 4514
DescribeABS ... 4514
GetQueryString ... 4514
UpdateFavoriteQuery ... 4514
UpdateQueryHistory ... 4514
UpdateTab ... 4514
Uso de machine learning de Aurora ... 4515
Uso de machine learning de Aurora con Aurora MySQL ... 4516
Requisitos para usar machine learning de Aurora ... 4517
Disponibilidad en regiones y versiones ... 4518
Características y limitaciones compatibles ... 4519
Configuración del clúster de Aurora para utilizar el machine learning de Aurora ... 4520
Uso de Amazon Bedrock con el clúster de base de datos de Aurora MySQL ... 4535
Uso de Amazon Comprehend con el clúster de base de datos de Aurora MySQL ... 4537
Uso de IA de SageMaker con el clúster de base de datos de Aurora MySQL ... 4540
Consideraciones sobre el rendimiento ... 4544
Monitorización ... 4546
Uso de machine learning de Aurora con Aurora PostgreSQL ... 4548
Requisitos para usar machine learning de Aurora ... 4548
Funciones y limitaciones compatibles ... 4549
Configuración del clúster de base de datos Aurora para utilizar el machine learning de Aurora ... 4550
Uso de Amazon Bedrock con el clúster de base de datos de Aurora PostgreSQL ... 4564
Uso de Amazon Comprehend con el clúster de base de datos de Aurora PostgreSQL ... 4566
Uso de IA de SageMaker con el clúster de base de datos de Aurora PostgreSQL ... 4568
Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado) ... 4573
Consideraciones sobre el rendimiento ... 4574
Supervisión ... 4580
Ejemplos de código ... 4582
Conceptos básicos ... 4593
Introducción a Aurora ... 4594

Conceptos básicos 4604
Acciones 4774
Escenarios 4980
Crear una biblioteca de préstamos de API de REST 4981
Crear un rastreador de elementos de trabajo de Aurora Serverless 4982
Prácticas recomendadas con Aurora 4987
Directrices operativas básicas de Amazon Aurora 4988
Recomendaciones de RAM de las instancias de base de datos 4988
Controladores de bases de datos de AWS 4989
Monitorización de Amazon Aurora 4990
Trabajo con los grupos de parámetros de base de datos y grupos de parámetros de clúster de base de datos 4990
Vídeo sobre las prácticas recomendadas de Amazon Aurora 4991
Ejecución de una prueba de concepto de Aurora 4992
Información general de una prueba de concepto de Aurora 4992
1. Identifique sus objetivos 4993
2. Conozca las características de su carga de trabajo 4994
3. Practique con la consola o la CLI 4995
Practique con la consola 4995
Practique con la AWS CLI 4996
4. Cree su clúster de Aurora 4997
5. Configure el esquema 4999
6. Importe los datos 5000
7. Transfiera su código SQL 5000
8. Especifique las opciones de configuración 5001
9. Conéctese a Aurora 5002
10. Ejecute la carga de trabajo 5004
11. Mida el rendimiento 5005
12. Pruebe la alta disponibilidad de Aurora 5008
13. Qué hacer a continuación 5010
Seguridad 5013
Autenticación de bases de datos 5015
Autenticación de contraseña 5016
Autenticación de bases de datos de IAM 5017
Autenticación Kerberos 5017
Administración de contraseñas con Aurora y Secrets Manager 5019

Disponibilidad en regiones y versiones ... 5019
Limitaciones ... 5019
Descripción general ... 5020
Ventajas ... 5021
Permisos ... 5021
Cumplimiento de la administración por parte de Aurora ... 5022
Administración de la contraseña para un clúster de base de datos ... 5023
Rotación de secretos para clústeres de bases de datos ... 5028
Visualización de los detalles del secreto para clústeres de base de datos ... 5030
Protección de los datos ... 5033
Cifrado de datos ... 5034
Privacidad del tráfico entre redes ... 5065
Identity and Access Management ... 5067
Público ... 5067
Autenticación con identidades ... 5068
Administración de acceso mediante políticas ... 5071
Cómo funciona Amazon Aurora con IAM ... 5073
Ejemplos de políticas basadas en identidades ... 5080
Políticas administradas por AWS ... 5103
Actualizaciones de políticas ... 5110
Prevención de la sustitución confusa entre servicios ... 5122
Autenticación de bases de datos de IAM ... 5124
Solución de problemas ... 5177
Registro y supervisión ... 5179
Validación de la conformidad ... 5182
Resiliencia ... 5183
Copia de seguridad y restauración ... 5183
Replicación ... 5184
Failover ... 5184
Seguridad de la infraestructura ... 5185
Grupos de seguridad ... 5185
Public accessibility (Accesibilidad pública) ... 5185
Puntos de enlace de la VPC (AWS PrivateLink) ... 5187
Consideraciones ... 1203
Disponibilidad ... 1203
Creación de un punto de enlace de la VPC de tipo interfaz ... 1204

Creación de una política de punto de enlace de la VPC ... 1204
Prácticas recomendadas de seguridad ... 5191
Control de acceso con grupos de seguridad ... 5192
Información general de los grupos de seguridad de VPC ... 5192
Escenario de grupos de seguridad ... 5194
Creación de un grupo de seguridad de VPC ... 5195
Asociación con un clúster de bases de datos ... 5196
Privilegios de la cuenta de usuario maestro ... 5196
Roles vinculados a servicios ... 5199
Permisos de roles vinculados a servicios de Amazon Aurora ... 5199
Permisos de roles vinculados a servicios para Amazon RDS Beta ... 5202
Rol vinculado a servicios para Amazon RDS Preview ... 5203
Uso de Amazon Aurora con Amazon VPC ... 5205
Uso de una clúster de base de datos en una VPC ... 5205
Escenarios de acceso a un clúster de base de datos en una VPC ... 5224
Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4) . 5231
Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble) ... 5239
Cuotas y restricciones ... 5251
Cuotas en Amazon Aurora ... 5251
Restricciones de la nomenclatura en Amazon Aurora ... 5258
Límites de tamaño de Amazon Aurora ... 5259
Solución de problemas ... 5261
No puede conectarse a la instancia de base de datos de ... 5261
Comprobar la conexión a la instancia de base de datos ... 5264
Solución de problemas de autenticación de conexión ... 5265
Problemas de seguridad ... 5265
Mensaje de error "No se pudieron recuperar los atributos de cuenta. Determinadas funciones de la consola pueden estar deterioradas". ... 5265
Restablecimiento de la contraseña del propietario de la instancia de base de datos ... 5265
Interrupción o reinicio de una instancia de base de datos ... 5266
Los cambios de parámetros no surten efecto ... 5267
Problemas de memoria que se puede liberar de Aurora ... 5267
Problemas de replicación de Aurora MySQL ... 5269
Diagnóstico y resolución de retardos entre réplicas de lectura ... 5269
Diagnóstico y solución de un error de replicación de lectura de MySQL ... 5271

Error de replicación detenida 5273
La replicación de réplicas de lectura no puede inicializar la estructura de metadatos 5274
Referencia de la API de Amazon RDS 5275
Uso de la API de consulta 5275
Parámetros de consulta 5275
Autenticación de solicitudes de consulta 5276
Solución de problemas de aplicaciones 5276
Recuperación de errores 5276
Consejos para la solución de problemas 5277
Historial de revisión 5278
AWSGlosario de 5396

¿Qué es Amazon Aurora?

Amazon Aurora (Aurora) es un motor de base de datos relacional completamente administrado compatible con MySQL y PostgreSQL. Ya sabe cómo MySQL y PostgreSQL combinan la velocidad y la fiabilidad de las bases de datos comerciales de gama alta con la sencillez y la rentabilidad de las bases de datos de código abierto. El código, las herramientas y las aplicaciones que se utilizan actualmente con las bases de datos MySQL y PostgreSQL se pueden usar con Aurora. Con algunas cargas de trabajo, Aurora puede proporcionar hasta cinco veces el rendimiento de MySQL y hasta tres veces el rendimiento de PostgreSQL sin requerir cambios en la mayoría de las aplicaciones existentes.

Aurora incluye un subsistema de almacenamiento de alto rendimiento. Sus motores de base de datos compatibles con MySQL y PostgreSQL están personalizados para aprovechar su almacenamiento de rápida distribución. El almacenamiento subyacente crece automáticamente en función de las necesidades. Un volumen de clúster de Aurora puede aumentar hasta un tamaño máximo de 128 tebibytes (TiB). Aurora también automatiza y estandariza la agrupación en clústeres y la reproducción de base de datos, que suelen ser algunos de los aspectos más problemáticos de la configuración y administración de las bases de datos.

Aurora forma parte del servicio de base de datos administrada de Amazon Relational Database Service (Amazon RDS). Amazon RDS facilita la configuración, el funcionamiento y el escalado de una base de datos relacional en la nube. Si no está familiarizado con Amazon RDS, consulte la guía del usuario de Amazon Relational Database Service. Para obtener más información sobre la variedad de opciones de bases de datos disponibles en Amazon Web Services, consulte Choosing the right database for your organization on AWS (Elegir la base de datos adecuada para su organización en AWS).

Temas

- Modelo de responsabilidad compartida de Amazon RDS
- Cómo funciona Amazon Aurora con Amazon RDS
- Clústeres de base de datos de Amazon Aurora
- Versiones de Amazon Aurora
- Regiones y zonas de disponibilidad
- Funciones admitidas en Amazon Aurora por Región de AWS y el motor de base de datos de Aurora
- Conexiones de puntos de conexión de Amazon Aurora

- Clases de instancia de base de datos de Amazon Aurora
- Almacenamiento de Amazon Aurora
- Fiabilidad de Amazon Aurora
- Seguridad de Amazon Aurora
- Alta disponibilidad para Amazon Aurora
- Replicación con Amazon Aurora
- Facturación de instancia de base de datos para Aurora

Modelo de responsabilidad compartida de Amazon RDS

Amazon RDS es responsable de alojar los componentes de software y la infraestructura de las instancias de base de datos y los clústeres de bases de datos. Usted es responsable del ajuste de las consultas, que es el proceso de optimización de las consultas SQL para mejorar el rendimiento. El rendimiento de las consultas depende en gran medida del diseño de la base de datos, el tamaño de los datos, la distribución de los datos, la carga de trabajo de la aplicación y los patrones de consulta, que pueden variar considerablemente. La supervisión y el ajuste son procesos enormemente individualizados que usted posee para sus bases de datos de RDS. Puede utilizar Información de rendimiento de Amazon RDS y otras herramientas para identificar consultas problemáticas.

Cómo funciona Amazon Aurora con Amazon RDS

En los siguientes puntos, se ilustra la relación de Amazon Aurora con los motores MySQL y PostgreSQL estándares disponibles en Amazon RDS:

- Puede elegir Aurora MySQL o Aurora PostgreSQL como opción del motor de base de datos cuando configura servidores de base de datos nuevos mediante Amazon RDS.
- Aurora aprovecha las características conocidas de Amazon Relational Database Service (Amazon RDS) para la gestión y administración. Aurora utiliza la interfaz de la Consola de administración de AWS de Amazon RDS, los comandos de AWS CLI y las operaciones de la API para gestionar las tareas de base de datos rutinarias, como el aprovisionamiento, la aplicación de parches, las copias de seguridad, la recuperación, la detección de errores y la reparación.
- Las operaciones de administración de Aurora normalmente afectan a clústeres completos de servidores de base de datos sincronizados mediante replicación, en lugar de instancias de base de datos individuales. La agrupación en clústeres, la replicación y la asignación de almacenamiento

automáticas simplifica y hace rentable configurar, usar y escalar las implementaciones de MySQL y PostgreSQL de mayor tamaño.

- Puede trasladar datos de Amazon RDS para MySQL y de Amazon RDS para PostgreSQL a Aurora creando y restaurando instantáneas, o bien configurando una replicación en un sentido. Puede usar herramientas de migración que le permiten convertir sus aplicaciones de RDS para MySQL y de RDS para PostgreSQL existentes a Aurora con un solo botón.

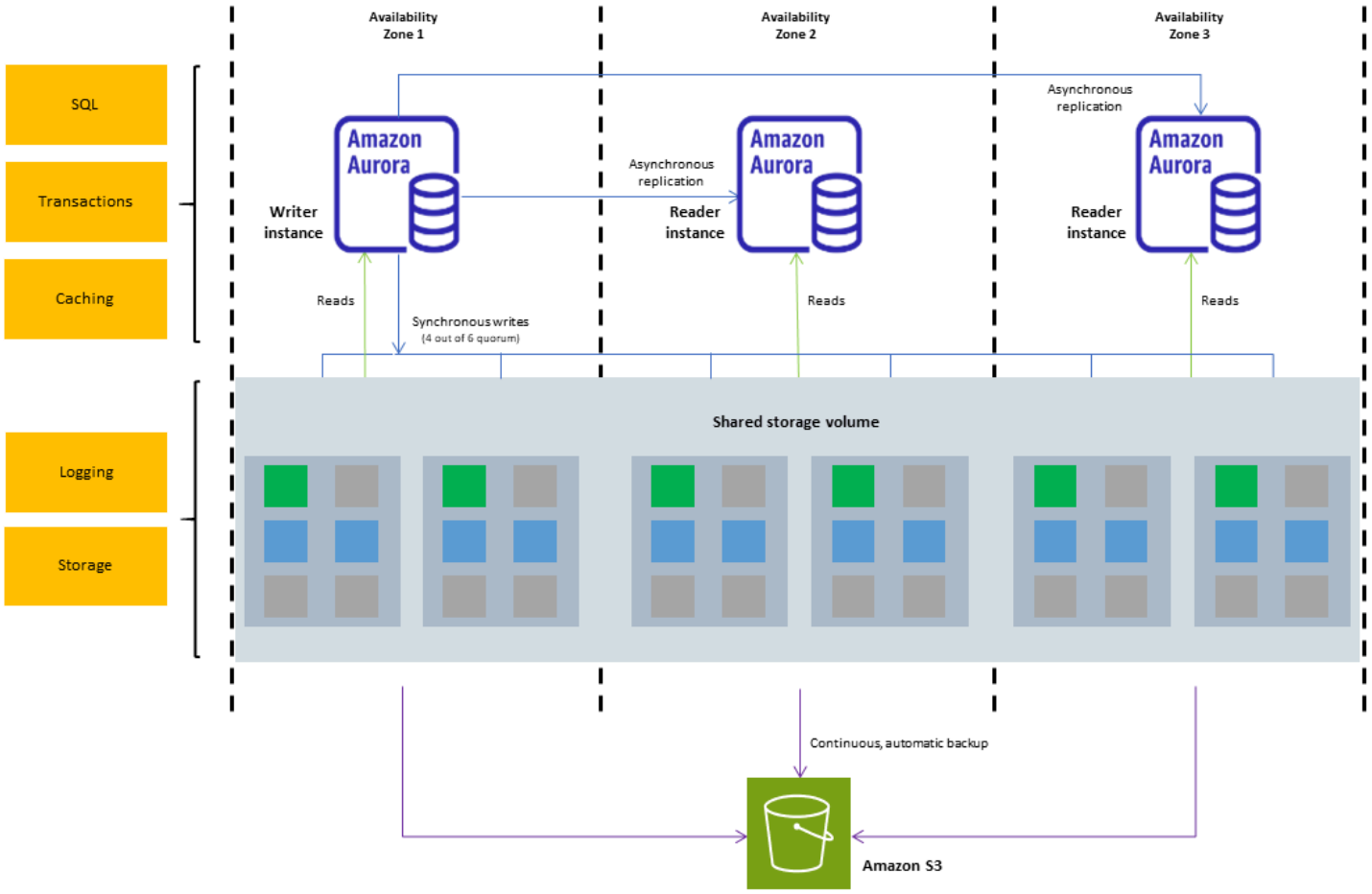
Antes de usar Amazon Aurora, complete los pasos que figuran en Configuración del entorno para Amazon Aurora y, después, revisar los conceptos y las características de Aurora que aparecen en Clústeres de base de datos de Amazon Aurora.

Clústeres de base de datos de Amazon Aurora

Un clúster de bases de datos de Amazon Aurora se compone de una o varias instancias de base de datos y de un volumen de clúster que administra los datos de esas instancias de base de datos. Un volumen de clúster de Aurora es un volumen de almacenamiento de base de datos virtual que abarca varias zonas de disponibilidad, de modo que una de esas zonas tiene una copia de los datos del clúster de bases de datos. Un clúster de bases de datos Aurora se compone de dos tipos de instancias de base de datos:

- Instancia de base de datos principal (escritor): admite operaciones de lectura y escritura y realiza todas las modificaciones de los datos en el volumen de clúster. Cada clúster de bases de datos Aurora tiene una instancia de base de datos principal.
- Réplica de Aurora (instancia de base de datos de lector): se conecta con el mismo volumen de almacenamiento que la instancia de base de datos principal, pero solo admite operaciones de lectura. Cada clúster de bases de datos Aurora puede tener hasta 15 réplicas de Aurora, además de la instancia de base de datos principal. Mantenga una alta disponibilidad localizando réplicas de Aurora en distintas zonas de disponibilidad. Aurora cambiará automáticamente a una réplica de Aurora en caso de que la instancia de base de datos principal deje de estar disponible. Puede especificar la prioridad de conmutación por error para réplicas de Aurora. Las réplicas de Aurora también pueden descargar las cargas de trabajo de lectura desde la instancia de base de datos principal.

El siguiente diagrama muestra la relación entre el volumen del clúster, la instancia de base de datos de escritor y las instancias de base de datos de lector en un clúster de bases de datos de Aurora.



> **Note**
>
> La información anterior se aplica a todos los clústeres de bases de datos de Aurora: aprovisionados, de consultas paralelas, de base de datos global de Aurora, de Aurora Serverless y los compatibles con Aurora MySQL y Aurora PostgreSQL.

El clúster de base de datos de Aurora muestra la separación entre la capacidad de computación y el almacenamiento. Por ejemplo, una configuración de Aurora con solo una instancia de base de datos sigue siendo un clúster, pero el volumen de almacenamiento subyacente implica que haya varios nodos de almacenamiento distribuidos en varias zonas de disponibilidad (AZ).

Las operaciones de entrada/salida (E/S) en los clústeres de base de datos de Aurora se cuentan de la misma manera, independientemente de si se encuentran en una instancia de base de datos de escritor o de lector. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.

Versiones de Amazon Aurora

Con Amazon Aurora, puede elegir el motor de base de datos relacional compatible que mejor se adapte a los requisitos de su aplicación y, al mismo tiempo, mantener la compatibilidad con los motores subyacentes. Aurora reutiliza el código del motor de base de datos de los motores compatibles. De este modo, puede aprovechar las habilidades, herramientas y bibliotecas existentes para esos motores. Al crear un clúster, debe especificar la versión del motor de bases de datos de Amazon Aurora que desea utilizar. La versión que elija determina la compatibilidad y las características disponibles.

En esta documentación se explican los puntos comunes y las diferencias entre Amazon Aurora y los motores de bases de datos correspondientes. Esta información le permite determinar la versión de software que debe seleccionar y cómo comprobar las características disponibles y las correcciones de errores en cada versión. También puede utilizar esta referencia para determinar la cadencia de actualización adecuada y planificar la actualización.

Motores de bases de datos compatibles con los clústeres de bases de datos de Amazon Aurora

Las siguientes bases de datos relacionales están disponibles en Amazon Aurora. Aurora reutiliza el código y mantiene la compatibilidad con los motores de base de datos subyacentes. Sin embargo, Aurora tiene sus propios números de versión, ciclos de lanzamiento y líneas de tiempo para la obsolescencia de la versión. Cada nueva versión de Aurora viene con notas de la versión que enumeran las nuevas características, correcciones, otras modificaciones y mejoras, que se aplican a cada versión.

base de datos de Aurora	Guía del usuario	Versiones disponibles	Notas de la versión
Amazon Aurora MySQL-Compatible Edition	Uso de Amazon Aurora MySQL	Actualizaciones del motor de base de datos de Amazon Aurora MySQL	Notas de la versión de Aurora MySQL
Edición de Amazon Aurora compatible con PostgreSQL	Uso de Amazon Aurora PostgreSQL	Actualizaciones del motor de base de	Notas de la versión de Aurora PostgreSQL

base de datos de Aurora	Guía del usuario	Versiones disponibles	Notas de la versión
		datos de Amazon Aurora PostgreSQL	

Especificar la versión de la base de datos de Amazon Aurora para su clúster de bases de datos

Al crear un nuevo clúster de bases de datos mediante la operación Crear base de datos en la Consola de administración de AWS, la AWS CLI o la operación de la API de `CreateDBCluster`, puede especificar cualquier versión de base de datos de Aurora disponible actualmente (principal o secundaria).

Para obtener más información sobre cómo crear clústeres de Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora. Para obtener más información sobre cómo cambiar la versión de un clúster de Aurora existente, consulte Modificación de un clúster de base de datos de Amazon Aurora.

> **Note**
>
> No todas las versiones de la base de datos de Aurora están disponibles en cada Región de AWS. Para obtener más información sobre las regiones y las versiones disponibles en cada Región de AWS, consulte Regiones y zonas de disponibilidad y Regiones y motores de base de datos admitidos para bases de datos globales Aurora.

Control de versiones de Amazon Aurora

Las versiones de Amazon Aurora son diferentes de las bases de datos comunitarias originales con las que son compatibles. Para ayudarlo a mantener la compatibilidad de las aplicaciones y aprovechar las características más recientes del motor de base de datos, en las siguientes secciones se explican las convenciones de control de versiones de Aurora y cómo se asignan las versiones de Aurora a sus respectivas bases de datos comunitarias.

Para obtener una lista de las bases de datos relacionales que están disponibles en Amazon Aurora, consulte Motores de bases de datos compatibles con los clústeres de bases de datos de Amazon Aurora.

Diferencias en los números de versión entre las bases de datos de la comunidad y Aurora

Cada versión de Amazon Aurora es compatible con una versión específica de su base de datos de la comunidad correspondiente. Puede encontrar la versión de la comunidad de la base de datos con la función `version` y la versión de Aurora con la función `aurora_version`.

En los ejemplos siguientes se muestra cómo encontrar la versión de comunidad de la base de datos para Aurora MySQL y Aurora PostgreSQL.

Aurora MySQL

La función `version` devuelve la versión comunitaria de la base de datos para Aurora MySQL.

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

Ejemplo de resultados:

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

Y la función `aurora_version` devuelve la versión de Aurora:

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

Ejemplo de resultados:

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

Aurora PostgreSQL

La función de `version` devuelve la versión comunitaria de la base de datos para Aurora PostgreSQL.

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

Ejemplo de resultados:

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

Y la función `aurora_version` devuelve la versión de Aurora:

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

Ejemplo de resultados:

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

Para obtener más información, consulte Comprobación de versiones de Aurora MySQL mediante SQL y Identificación de las versiones de Amazon Aurora PostgreSQL.

Versiones predeterminadas de Amazon Aurora

La versión predeterminada es la versión que Aurora elige automáticamente para la creación o actualización de la base de datos cuando no se especifica manualmente una versión de motor de destino. Por ejemplo, el siguiente comando muestra la versión de motor predeterminada para Aurora PostgreSQL (se incluye una salida de muestra).

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

16.4
```

Cada versión principal tiene la correspondiente versión secundaria predeterminada. Por lo tanto, la versión secundaria predeterminada es 16.n para Aurora PostgreSQL 16 y el número de versión n

cambia cuando Aurora lanza nuevas versiones secundarias predeterminadas. Normalmente, Aurora lanza dos versiones secundarias predeterminadas para cada versión principal por año. El siguiente script del intérprete de comandos bash muestra las versiones secundarias predeterminadas para un conjunto de versiones principales de Aurora PostgreSQL (se incluye una salida de muestra).

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

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

Si habilita las actualizaciones automáticas de versiones secundarias para el clúster de bases de datos de Aurora, Aurora utilizará la versión secundaria predeterminada o una versión secundaria más reciente para una versión principal determinada. Por ejemplo, si la versión secundaria predeterminada para Aurora PostgreSQL 15 es 15.8 y la versión más reciente 15.10 también está disponible, Aurora puede actualizarse automáticamente a 15.8 o 15.10.

Versiones principales de Amazon Aurora

Las versiones de Aurora utilizan el esquema *major*.*minor*.*patch*. Una version principal de Aurora se refiere a la versión principal de la comunidad de MySQL o PostgreSQL con la que Aurora es compatible. Las versiones principales de Aurora MySQL y Aurora PostgreSQL permanecen disponibles bajo el soporte estándar al menos hasta el final de la vida útil de la comunidad para la versión de la comunidad correspondiente. Puede seguir ejecutando una versión principal después de la fecha de finalización del soporte estándar de Aurora si paga una cuota. Para obtener más información, consulte Soporte extendido de Amazon RDS con Amazon Aurora y Precios de Amazon Aurora.

Para obtener más información sobre las versiones principales y el calendario de lanzamientos de Aurora MySQL y Aurora PostgreSQL, consulte las páginas siguientes de las notas de la versión correspondiente:

- Calendario de lanzamientos de las principales versiones de Aurora MySQL
- Calendario de lanzamientos de versiones principales de Aurora PostgreSQL

También puede ver información sobre las fechas de soporte de las versiones principales del motor ejecutando el comando describe-db-major-engine-versions de la AWS CLI o mediante la operación de la API de RDS DescribeDBMajorEngineVersions.

> **Note**
>
> El Soporte extendido de Amazon RDS para Aurora MySQL (versión 2) comienza el 1 de noviembre de 2024, pero no se cobrará hasta el 1 de diciembre de 2024. Entre el 1 y el 30 de noviembre de 2024, todos los clústeres de bases de datos de la versión 2 de Aurora MySQL están cubiertos por el Soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de Amazon RDS para determinadas versiones de Aurora.

Cuánto tiempo permanecen disponibles las versiones principales de Amazon Aurora

Las versiones principales de Amazon Aurora permanecen disponibles al menos hasta el final de la vida útil de la comunidad para la versión de la comunidad correspondiente. Puede utilizar las fechas de finalización del soporte estándar de Aurora para planificar sus ciclos de pruebas y actualizaciones. Estas fechas representan la fecha más temprana en la que podría requerirse una actualización a una versión más reciente. Para obtener más información sobre las fechas, consulte Versiones principales de Amazon Aurora.

Antes de que Aurora le pida que actualice a una versión principal más reciente y para ayudarlo a realizar la planificación, recibirá un recordatorio con al menos 12 meses de antelación. Los recordatorios comunican lo siguiente acerca del proceso de actualización.

- El momento de determinados hitos
- El impacto en los clústeres de base de datos
- Acciones recomendadas

Recomendamos que pruebe minuciosamente las aplicaciones con nuevas versiones de la base de datos antes de realizar una actualización del clúster a una nueva versión principal.

Cuando la versión principal alcance el final del soporte estándar de Aurora, cualquier clúster de bases de datos que aún ejecute la versión anterior se actualiza automáticamente a una versión de soporte extendido durante un periodo de mantenimiento programado. Podrían aplicarse cargos adicionales por el soporte extendido. Para obtener más información sobre el soporte extendido de Amazon RDS, consulte Uso del soporte extendido de Amazon RDS.

Versiones secundarias de Amazon Aurora

Las versiones de Aurora utilizan el esquema *major*.*minor*.*patch*. Una versión secundaria de Aurora proporciona mejoras progresivas de la comunidad y específicas de Aurora para el servicio, por ejemplo, nuevas características y correcciones.

Para obtener más información sobre las versiones secundarias y el calendario de lanzamientos de Aurora MySQL y Aurora PostgreSQL, consulte las páginas siguientes de las notas de la versión correspondiente:

- Calendario de lanzamientos de las versiones secundarias de Aurora MySQL
- Calendario de lanzamientos de las versiones secundarias de Aurora PostgreSQL

En las siguientes secciones se describen los detalles sobre la cadencia y la vida útil que puede esperar de las versiones secundarias de Aurora.

Temas

- Con qué frecuencia se publican las versiones secundarias de Amazon Aurora
- Cuánto tiempo permanecen disponibles las versiones secundarias de Amazon Aurora

Con qué frecuencia se publican las versiones secundarias de Amazon Aurora

En general, lanzamos versiones secundarias de Amazon Aurora trimestralmente. La programación de versiones puede variar para seleccionar características o correcciones adicionales.

Cuánto tiempo permanecen disponibles las versiones secundarias de Amazon Aurora

Normalmente, Amazon Aurora hace que cada versión secundaria de una determinada versión principal esté disponible durante al menos 12 meses. Al final de este periodo, Aurora podría actualizar automáticamente la base de datos a la versión secundaria predeterminada o a una

posterior. Aurora comienza la actualización durante el periodo de mantenimiento programado para cualquier clúster de bases de datos que esté ejecutando la versión secundaria anterior.

En algunos casos, Aurora podría reemplazar una versión secundaria de una determinada versión principal antes del periodo habitual de 12 meses. Entre los motivos se pueden incluir problemas de seguridad críticos o la fecha de finalización del soporte para una versión principal.

Antes de comenzar las actualizaciones automáticas de las versiones secundarias que se aproximan al final de la vida útil, generalmente Aurora proporciona un recordatorio con tres meses de antelación. Aurora detalla lo siguiente acerca del proceso de actualización.

- El momento de determinados hitos
- El impacto en los clústeres de base de datos
- Acciones recomendadas

Las notificaciones con menos de tres meses de antelación describen asuntos críticos, como problemas de seguridad, que requieren una acción más rápida.

Si la opción Actualización automática de versión secundaria está habilitada, recibirá un recordatorio, pero no una notificación de eventos de RDS. Aurora actualiza la base de datos dentro de un periodo de mantenimiento una vez transcurrido el plazo de actualización obligatorio.

Si la configuración Actualización automática de versión secundaria no está habilitada, recibirá un recordatorio y un evento de clúster de bases de datos de Amazon RDS con una categoría de `maintenance` y un ID de `RDS-EVENT-0156`. Aurora actualizará la base de datos en el siguiente periodo de mantenimiento.

Tenga en cuenta que, una vez que una versión secundaria alcanza el final de soporte estándar de Aurora, no se publicarán más versiones de revisión para esa versión secundaria. Para recibir correcciones de errores críticos o CVE, debe actualizar a una versión secundaria con soporte estándar.

Para obtener más información sobre las actualizaciones de versiones secundarias, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora.

Versiones de parches de Amazon Aurora

Las versiones de Aurora utilizan el esquema *`major`*`.`*`minor`*`.`*`patch`*. Una versión de la revisión de Aurora incluye importantes correcciones de errores añadidas a una versión secundaria después de

su lanzamiento inicial (por ejemplo, Aurora MySQL 3.04.0, 3.04.1, ..., 3.04.3). Mientras que cada nueva versión secundaria proporciona nuevas características de Aurora, las nuevas versiones de parches dentro de una versión secundaria específica se utilizan principalmente para resolver problemas importantes.

Para obtener más información sobre la aplicación de parches, consulte Mantenimiento de un clúster de base de datos de Amazon Aurora.

Actualización de clústeres de base de datos de Amazon Aurora

Con Amazon Aurora, puede controlar y probar las actualizaciones de los clústeres de bases de datos. Amazon Aurora ofrece opciones para la actualización automática de versiones secundarias, el control manual de las actualizaciones, las actualizaciones necesarias y las pruebas previas a la actualización. Puede mantener los clústeres actualizados con la versión secundaria más reciente, aplazar las actualizaciones no críticas, forzar las actualizaciones en caso de problemas graves y validar el comportamiento de las actualizaciones en entornos que no son de producción. En las siguientes secciones se detalla cómo administrar y probar las actualizaciones del clúster de base de datos de Aurora mediante estas capacidades.

Actualizaciones automáticas de versiones secundarias para Aurora

Las actualizaciones automáticas de las versiones secundarias actualizan periódicamente la base de datos a versiones recientes del motor de base de datos. Sin embargo, es posible que la actualización no siempre incluya la versión más reciente del motor de base de datos. Si necesita mantener las bases de datos en versiones específicas en momentos determinados, le recomendamos que las actualice manualmente a las versiones de base de datos que necesite según el calendario requerido. En caso de problemas de seguridad críticos o cuando una versión alcance su fecha de fin de soporte, Amazon Aurora podría aplicar una actualización de la versión secundaria aunque no haya habilitado la opción Actualización automática de versión secundaria. Para obtener más información, consulte la documentación de actualización para el motor de base de datos específico.

Consulte Actualización de la versión secundaria o el nivel de parche de un clúster de bases de datos Aurora MySQL y Actualización a una versión secundaria.

Puede mantenerse actualizado con las versiones menores de Aurora activando Auto minor version upgrade (Actualización automática de la versión secundaria) para cada instancia de base de datos en el clúster de Aurora. Aurora realiza la actualización automática solo si todas las instancias de base de datos del clúster tienen esta configuración activada.

Si Actualización automática de versión secundaria es Sí para el clúster de bases de datos, Aurora se actualiza automáticamente a la versión secundaria predeterminada o a una versión secundaria más reciente. Por ejemplo, si la versión secundaria predeterminada es 15.8 para Aurora PostgreSQL 15 y existe la versión 15.10, el objetivo de la actualización automática podría ser 15.8 o 15.10.

Aurora normalmente programa actualizaciones automáticas dos veces al año para clústeres de bases de datos que tienen habilitada la actualización automática de versiones secundarias. Estas actualizaciones se producen durante el periodo de mantenimiento que especifique para el clúster. Para obtener más información, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora.

Las actualizaciones automáticas de versiones secundarias se comunican de antemano a través de un evento de clúster de base de datos de Amazon RDS con una categoría `maintenance` e ID de `RDS-EVENT-0156`. Para obtener más información, consulte Categorías y mensajes de eventos de Amazon RDS para Aurora.

Control manual de las actualizaciones de los clústeres de bases de datos a nuevas versiones

Si tiene habilitada la configuración Actualización automática de versión secundaria, Aurora actualiza automáticamente el clúster de bases de datos a la versión secundaria predeterminada o a una versión secundaria más reciente. Por lo general, Aurora programa actualizaciones automáticas dos veces al año para clústeres de bases de datos que tienen la configuración Actualización automática de versión secundaria habilitada. Estas actualizaciones se inician durante las ventanas de mantenimiento especificadas por el cliente.

Para desactivar las actualizaciones automáticas de versiones secundarias, deshabilite Actualización automática de versión secundaria en cualquier instancia de base de datos de un clúster de Aurora. Aurora realiza una actualización automática de la versión secundaria solo si todas las instancias de base de datos del clúster tienen la configuración habilitada.

> **Note**
>
> En el caso de actualizaciones obligatorias, como el final de la vida útil de una versión secundaria, Aurora actualiza el clúster de bases de datos incluso si la opción Actualización automática de versión secundaria está deshabilitada. Recibirá un recordatorio, pero no una notificación de evento de RDS. Aurora actualiza el clúster dentro de un periodo de mantenimiento una vez transcurrido el plazo de actualización obligatorio.

Como las actualizaciones de versión pueden conllevar algunos riesgos de compatibilidad, no se producen automáticamente. Debe iniciarlas, excepto si hay obsolescencia de la versión principal. Recomendamos que pruebe minuciosamente las aplicaciones con nuevas versiones de la base de datos antes de realizar una actualización del clúster a una versión principal.

Para obtener más información acerca de cómo actualizar un clúster de bases de datos a una nueva versión principal de Aurora, consulte Actualización de clústeres de base Amazon Aurora MySQL y Actualización de clústeres de base de datos PostgreSQL de Amazon Aurora.

Actualizaciones necesarias de Amazon Aurora

Para determinadas correcciones críticas, Aurora podría llevar a cabo una actualización administrada a un nivel de revisión nueva dentro de la misma versión secundaria LTS. En este caso, Aurora actualiza el clúster incluso si Actualización automática de versión secundaria está desactivada. Antes, Aurora comunica el proceso detallado de actualización. Los detalles incluyen el tiempo de ciertos hitos, el impacto en los clústeres de bases de datos y las acciones recomendadas. Dichas actualizaciones administradas se producen automáticamente en el periodo de mantenimiento del clúster.

Probar su clúster de bases de datos con una nueva versión de Aurora antes de actualizar

Puede probar el proceso de actualización y cómo funciona la nueva versión con su aplicación y carga de trabajo. Utilice uno de los siguientes métodos:

- Clone su clúster mediante la característica de clonación rápida de base de datos de Amazon Aurora. Realice la actualización y cualquier prueba posterior a la actualización en el nuevo clúster.
- Restaure desde una instantánea de clúster para crear un nuevo clúster de Aurora. Puede crear una instantánea de clúster usted mismo de un clúster de Aurora existente. Aurora también crea automáticamente instantáneas periódicas para cada uno de sus clústeres. A continuación, puede iniciar una actualización de versión para el nuevo clúster. Puede experimentar con la copia actualizada del clúster antes de decidir si desea actualizar el clúster original.

Para obtener más información sobre estas formas de crear nuevos clústeres para pruebas, consulte Clonación de un volumen de clúster de base de datos de Amazon Aurora y Creación de una instantánea de clúster de base de datos.

Compatibilidad de versiones de Amazon Aurora

Si la base de datos de Amazon Aurora tiene dependencias complejas en función del comportamiento específico del motor de base de datos, le recomendamos que realice pruebas exhaustivas antes de actualizar a versiones más recientes del motor de base de datos. Existen opciones de compatibilidad a largo plazo para que pueda mantener los clústeres de bases de datos en determinadas versiones del motor de Aurora incluso después de que su reemplazo por versiones más recientes. En las siguientes secciones se explican las opciones de compatibilidad a largo plazo para los clústeres de base de datos de Aurora.

Compatibilidad a largo plazo para versiones secundarias de Amazon Aurora seleccionadas

Para cada versión principal de Aurora, determinadas versiones secundarias se designan como versiones de compatibilidad a largo plazo (LTS) y se ponen a disposición durante al menos tres años. Es decir, al menos una versión secundaria por versión principal está disponible durante más tiempo que los 12 meses habituales. Normalmente, Aurora le enviará un recordatorio seis meses antes de que finalice este periodo. Aurora comunica lo siguiente acerca del proceso de actualización.

- El momento de determinados hitos
- El impacto en los clústeres de base de datos
- Acciones recomendadas

Las notificaciones con menos de seis meses de antelación comunican asuntos críticos, como problemas de seguridad, que requieren una actuación más rápida.

Las versiones secundarias de LTS solo incluyen correcciones críticas (a través de versiones de revisiones). Una versión LTS no incluye nuevas características publicadas después de su introducción. Una vez al año, los clústeres de base de datos que se ejecutan en una versión secundaria LTS, se parchean a la última versión del parche de la versión LTS. Aurora aplica revisiones a los clústeres para que usted se beneficie de correcciones de seguridad y estabilidad acumulativas. Si hay correcciones críticas, es posible que Aurora aplique la revisión de una versión secundaria LTS con más frecuencia, por ejemplo, para la seguridad, que deben aplicarse.

> **Note**
>
> Si desea permanecer en una versión secundaria de LTS mientras dure su ciclo de vida, desactive la actualización automática de versiones secundarias para sus instancias de base

de datos. Para evitar actualizar automáticamente el clúster de base de datos desde la versión secundaria de LTS, desmarque la casilla Habilitar actualización automática de versiones secundarias en cualquier instancia de base de datos de su clúster de Aurora.

Para conocer los números de versión de todas las versiones de Aurora LTS, consulte Versiones de soporte a largo plazo (LTS) de Aurora MySQL y Uso de una versión de compatibilidad a largo plazo (LTS) de Aurora PostgreSQL.

Soporte extendido de Amazon RDS para determinadas versiones de Aurora

Con la compatibilidad extendida de Amazon RDS, puede seguir ejecutando la base de datos en una versión principal del motor después de la fecha de finalización de la compatibilidad estándar de Aurora por un precio adicional. Durante el periodo del Soporte extendido de RDS, Amazon RDS suministrará parches para las CVE cruciales y altas, según se definen en las puntuaciones de gravedad de CVSS de la National Vulnerability Database (NVD). Para obtener más información, consulte Soporte extendido de Amazon RDS con Amazon Aurora.

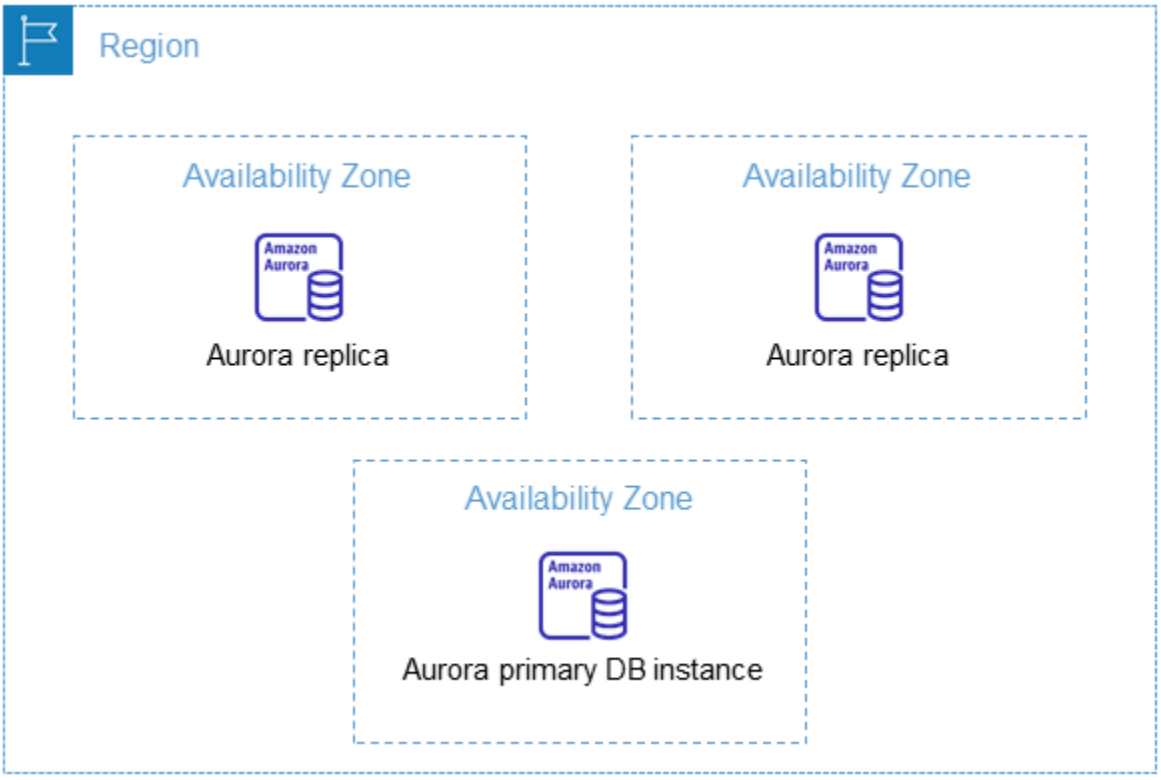
El Soporte extendido de RDS solo está disponible en determinadas versiones de Aurora. Para obtener más información, consulte Versiones principales de Amazon Aurora.

Regiones y zonas de disponibilidad

Los recursos de informática en la nube de Amazon están alojados en varias ubicaciones de todo el mundo. Dichas ubicaciones se componen de regiones de AWS y zonas de disponibilidad. Cada región de AWS es un área geográfica independiente. Cada región de AWS tiene varias ubicaciones aisladas conocidas como zonas de disponibilidad.

> **Note**
>
> Para obtener información acerca de cómo encontrar las zonas de disponibilidad de una región de AWS, consulte el tema donde se describen las zonas de disponibilidad en la documentación de Amazon EC2.

Amazon opera centros de datos de alta disponibilidad con tecnología de vanguardia. Aunque es infrecuente, puede suceder que se produzcan errores que afecten a la disponibilidad de las instancias de bases de datos que están en la misma ubicación. Si aloja todas sus instancias de base de datos en una ubicación que se ve afectada por dicho error, ninguna de sus instancias de base de datos estará disponible.



Es importante recordar que cada región de AWS es completamente independiente. Cualquier actividad de Amazon RDS; que se inicie (por ejemplo, la creación de instancias de base de datos o la enumeración de las instancias de base de datos disponibles) se ejecuta solo en la región de AWS predeterminada actual. La Región de AWS por defecto se puede cambiar en la consola, o estableciendo la variable de entorno `AWS_DEFAULT_REGION`. O bien, se puede anular utilizando el parámetro `--region` con la AWS Command Line Interface (AWS CLI). Para obtener más información, consulte Configurar la AWS Command Line Interface, específicamente las secciones relativas a las variables de entorno y las opciones de la línea de comandos.

Amazon RDS admite las regiones especiales de AWS denominadas AWS GovCloud (US). Estas se han diseñado para permitir a los clientes y los organismos del Gobierno de Estados Unidos que traspasen cargas de trabajo más confidenciales a la nube. Mediante las regiones AWS GovCloud (US) se atienden las necesidades reglamentarias y de cumplimiento propias del Gobierno de Estados Unidos. Para obtener más información, consulte ¿Qué es AWS GovCloud (US)?

Para crear una instancia de base de datos de Amazon RDS o trabajar con ella en una región concreta de AWS, use el punto de enlace de servicio regional correspondiente.

> **Note**
>
> Aurora no admite zonas locales.

AWSRegiones de

Cada región de AWS se ha diseñado para que esté totalmente aislada de las demás regiones de AWS. Este diseño logra la mayor tolerancia a errores y estabilidad posibles.

Cuando se consultan los recursos, solo se ven los recursos vinculados a la región de AWS especificada. Esto se debe a que las regiones de AWSestán aisladas entre sí y no replicamos automáticamente los recursos en distintas regiones de AWS.

Disponibilidad por región

Cuando trabaje con un clúster de base de datos de Aurora usando la interfaz de línea de comandos o las operaciones de la API, asegúrese de especificar su punto de enlace regional.

Temas

- Disponibilidad por región de Aurora MySQL

- Disponibilidad por región de Aurora PostgreSQL

Disponibilidad por región de Aurora MySQL

En la tabla siguiente, se muestran las regiones de AWS donde Aurora MySQL está disponible actualmente y el punto de enlace de cada región.

Nombre de la región	Región	Punto de conexión	Protocolo	
Este de EE. UU. (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
Este de EE. UU. (Norte de Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Oeste de EE. UU. (Norte de California)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
Oeste de EE. UU. (Oregón)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
África (Ciudad del Cabo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asia-Pacífico	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo
(Hong Kong)			
Asia-Pacífico (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
Asia-Pacífico (Yakarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS
Asia-Pacífico (Malasia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Asia-Pacífico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS
Asia-Pacífico (Bombay)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
Asia-Pacífico (Nueva Zelanda)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS
Asia-Pacífico (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo	
Asia-Pacífico (Seúl)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	
Asia-Pacífico (Singapur)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Asia-Pacífico (Sídney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asia-Pacífico (Taipéi)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asia-Pacífico (Tailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asia-Pacífico (Tokio)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canadá (centro)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Oeste de Canadá (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo
Europa (Fráncfort)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
Europa (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
Europa (Milán)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS
Europa (París)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
Europa (España)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
Europa (Estocolmo)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
Europa (Zúrich)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS
Israel (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
México (central)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Medio Oriente (Baréin)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo	
Medio Oriente (EAU)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS	
América del Sur (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Este de EE. UU.)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Oeste de EE.UU.)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Disponibilidad por región de Aurora PostgreSQL

En la tabla siguiente, se muestran las regiones de AWS donde Aurora PostgreSQL está disponible actualmente y el punto de enlace de cada región.

Nombre de la región	Región	Punto de conexión	Protocolo	
Este de EE. UU. (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo	
Este de EE. UU. (Norte de Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Oeste de EE. UU. (Norte de California)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
Oeste de EE. UU. (Oregón)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
África (Ciudad del Cabo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asia-Pacífico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Asia-Pacífico (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	
Asia-Pacífico (Yakarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo	
Asia-Pacífico (Malasia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Asia-Pacífico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Asia-Pacífico (Bombay)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Asia-Pacífico (Nueva Zelanda)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
Asia-Pacífico (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Asia-Pacífico (Seúl)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	
Asia-Pacífico (Singapur)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Asia-Pacífico (Sídney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo	
Asia-Pacífico (Taipéi)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asia-Pacífico (Tailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asia-Pacífico (Tokio)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canadá (centro)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Oeste de Canadá (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Fráncfort)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europa (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	
Europa (Milán)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	
Europa (París)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo	
Europa (España)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS	
Europa (Estocolmo)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS	
Europa (Zúrich)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS	
Israel (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS	
México (central)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS	
Medio Oriente (Baréin)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS	
Medio Oriente (EAU)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS	
América del Sur (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Este de EE. UU.)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo	
AWS GovCloud (Oeste de EE.UU.)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Zonas de disponibilidad

Una zona de disponibilidad es una ubicación aislada en una Región de AWS determinada. Cada región tiene varias zonas de disponibilidad (AZ) diseñadas para proporcionar alta disponibilidad para la región. Una AZ se identifica mediante el código de la región de AWS seguido de un identificador de letra (por ejemplo, `us-east-1a`). Si crea la VPC y las subredes en lugar de utilizar la VPC predeterminada, defina cada subred en una AZ específica. Cuando crea un clúster de base de datos de Aurora, Aurora crea la instancia principal en una de las subredes del grupo de subredes de base de datos de la VPC. Por lo tanto, asocia esa instancia con una AZ específica elegida por Aurora.

Cada clúster de base de datos Aurora aloja copias del almacenamiento en tres zonas de disponibilidad separadas seleccionadas automáticamente por Aurora entre las zonas de disponibilidad del grupo de subredes de base de datos. Cada instancia de base de datos en el clúster debe estar en una de estas tres zonas de disponibilidad.

Cuando crea una instancia de base de datos en el clúster, Aurora elige automáticamente una zona de disponibilidad adecuada para esa instancia si no especifica una zona de disponibilidad.

Utilice el comando describe-availability-zones de Amazon EC2, tal y como se indica a continuación, para describir las zonas de disponibilidad dentro de la región especificada que están habilitadas para su cuenta.

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

Por ejemplo, para describir las zonas de disponibilidad de la región Este de EE. UU. (Norte de Virginia) (us-east-1) que están habilitadas para su cuenta, ejecute el siguiente comando:

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

Para obtener información sobre cómo especificar la AZ cuando crea un clúster o agrega instancias, consulte Configurar la red para el clúster de base de datos.

Zona horaria local para los clústeres de base de datos de Amazon Aurora

De manera predeterminada, la zona horaria del clúster de base de datos Amazon Aurora es el horario universal coordinado (UTC). En su lugar, puede definir la zona horaria de las instancias del clúster de base de datos en la zona horaria local de su aplicación.

Para definir la zona horaria local de un clúster de base de datos, configure el parámetro de zona horaria en uno de los valores admitidos. Este parámetro se establece en el grupo de parámetros del clúster para su clúster de base de datos.

- Para Aurora MySQL, el nombre de este parámetro es `time_zone`. Para obtener información sobre las mejores prácticas para configurar el parámetro `time_zone`, consulte Optimización de las operaciones de marca temporal.
- Para Aurora PostgreSQL, el nombre de este parámetro es `timezone`.

Cuando se define el parámetro de zona horaria de un clúster de base de datos, todas las instancias de ese clúster de base de datos cambian para usar la nueva zona horaria local. En algunos casos, es posible que otros clústeres de base de datos de Aurora utilicen el mismo grupo de parámetros de clúster. Si es así, todas las instancias de esos clústeres de base de datos cambian para usar también la nueva zona horaria local. Para obtener más información acerca de los parámetros de nivel de clúster, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos.

Después de definir la zona horaria local, todas las conexiones nuevas con la base de datos reflejarán el cambio. En algunos casos, es posible que tenga una conexión con la base de datos abierta al cambiar la zona horaria local. Si es así, no verá la actualización de la zona horaria local hasta que cierre la conexión y abra una nueva conexión.

Si va a replicar entre varias regiones de AWS, el clúster de base de datos original de la replicación y la réplica usarán distintos grupos de parámetros. Los grupos de parámetros son exclusivos de una región de AWS. Si desea usar la misma zona horaria local para cada instancia, asegúrese de configurar el parámetro de zona horaria de los grupos de parámetros tanto del elemento original de replicación como de la réplica.

Cuando se restaura un clúster de base de datos desde una instantánea de clúster de base de datos, la zona horaria local se define como UTC. Podrá actualizar la zona horaria a su zona horaria local una vez que se haya completado la restauración. En algunos casos, es posible que desee restaurar un clúster de base de datos en un punto en el tiempo. Si lo hace, la zona horaria local del clúster restaurado será el ajuste de zona horaria del grupo de parámetros del clúster de base de datos restaurado.

Puede definir su zona horaria local en uno de los valores que se muestran en la siguiente tabla. Para enumerar todas las zonas horarias disponibles, puede utilizar las siguientes consultas SQL:

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

> **Note**
>
> Para algunas zonas horarias, los valores de horas de determinados rangos de fechas pueden indicarse de un modo incorrecto, como se muestra en la tabla. Para algunas zonas horarias de Australia, la abreviatura de zona devuelta es un valor obsoleto, como se muestra en la tabla.

Time zone (Zona horarioa)	Notas
`Africa/Harare`	Este ajuste de zona horaria puede devolver valores incorrectos del 28 de febrero de 1903 a las 21:49:40 GMT hasta el 28 de febrero de 1903 a las 21:55:48 GMT.
`Africa/Monrovia`	
`Africa/Nairobi`	Este ajuste de zona horaria puede devolver valores incorrectos del 31 de diciembre de 1939 a las 21:30:00 GMT hasta el 31 de diciembre de 1959 a las 21:15:15 GMT.
`Africa/Windhoek`	

Time zone (Zona horarioa)	Notas
`America/Bogota`	Este ajuste de zona horaria puede devolver valores incorrectos del 23 de noviembre de 1914 a las 04:56:16 GMT hasta el 23 de noviembre de 1914 a las 04:56:20 GMT.
`America/Caracas`	
`America/Chihuahua`	
`America/Cuiaba`	
`America/Denver`	
`America/Fortaleza`	En algunos casos, para un clúster de base de datos en la región América del Sur (São Paulo), la hora no se muestra correctamente para una zona horaria de Brasil recientemente cambiada. Si es así, restablezca el parámetro de zona horaria del clúster de base de datos en `America/Fortaleza` .
`America/Guatemala`	
`America/Halifax`	Este ajuste de zona horaria puede devolver valores incorrectos del 27 de octubre de 1918 a las 05:00:00 GMT hasta el 31 de octubre de 1918 a las 05:00:00 GMT.
`America/Manaus`	Si su clúster de base de datos se encuentra en la zona horaria de América del Sur (Cuiabá) y la hora esperada no se muestra correctamente para la zona horaria de Brasil recientemente cambiada, restablezca el parámetro de zona horaria del clúster de base de datos en `America/Manaus` .
`America/Matamoros`	

Time zone (Zona horarioa)	Notas
America/Monterrey	
America/Montevideo	
America/Noronha	
America/Phoenix	
America/Tijuana	
Asia/Ashgabat	
Asia/Baghdad	
Asia/Baku	
Asia/Bangkok	
Asia/Beirut	
Asia/Calcutta	
Asia/Kabul	
Asia/Karachi	
Asia/Kathmandu	
Asia/Muscat	Este ajuste de zona horaria puede devolver valores incorrectos del 31 de diciembre de 1919 a las 20:05:36 GMT hasta el 31 de diciembre de 1919 a las 20:05:40 GMT.
Asia/Riyadh	Este ajuste de zona horaria puede devolver valores incorrectos del 13 de marzo de 1947 a las 20:53:08 GMT hasta el 31 de diciembre de 1949 a las 20:53:08 GMT.

Time zone (Zona horarioa)	Notas
`Asia/Seoul`	Este ajuste de zona horaria puede devolver valores incorrectos del 30 de noviembre de 1904 a las 15:30:00 GMT hasta el 7 de septiembre de 1945 a las 15:00:00 GMT.
`Asia/Shanghai`	Este ajuste de zona horaria puede devolver valores incorrectos del 31 de diciembre de 1927 a las 15:54:08 GMT hasta el 2 de junio de 1940 a las 16:00:00 GMT.
`Asia/Singapore`	
`Asia/Taipei`	Este ajuste de zona horaria puede devolver valores incorrectos del 30 de septiembre de 1937 a las 16:00:00 GMT hasta el 29 de septiembre de 1979 a las 15:00:00 GMT.
`Asia/Tehran`	
`Asia/Tokyo`	Este ajuste de zona horaria puede devolver valores incorrectos del 30 de septiembre de 1937 a las 15:00:00 GMT hasta el 31 de diciembre de 1937 a las 15:00:00 GMT.
`Asia/Ulaanbaatar`	
`Atlantic/Azores`	Este ajuste de zona horaria puede devolver valores incorrectos del 24 de mayo de 1911 a las 01:54:32 GMT hasta el 1 de enero de 1912 a las 01:54:32 GMT.
`Australia/Adelaide`	La abreviatura de esta zona horaria aparece como CST en lugar de ACDT/ACST.
`Australia/Brisbane`	La abreviatura de esta zona horaria aparece como EST en lugar de AEDT/AEST.
`Australia/Darwin`	La abreviatura de esta zona horaria aparece como CST en lugar de ACDT/ACST.

Time zone (Zona horarioa)	Notas
Australia/Hobart	La abreviatura de esta zona horaria aparece como EST en lugar de AEDT/AEST.
Australia/Perth	La abreviatura de esta zona horaria aparece como WST en lugar de AWDT/AWST.
Australia/Sydney	La abreviatura de esta zona horaria aparece como EST en lugar de AEDT/AEST.
Brazil/East	
Canada/Saskatchewan	Este ajuste de zona horaria puede devolver valores incorrectos del 27 de octubre de 1918 a las 08:00:00 GMT hasta el 31 de octubre de 1918 a las 08:00:00 GMT.
Europe/Amsterdam	
Europe/Athens	
Europe/Dublin	
Europe/Helsinki	Este ajuste de zona horaria puede devolver valores incorrectos del 30 de abril de 1921 a las 22:20:08 GMT hasta el 30 de abril de 1921 a las 22:20:11 GMT.
Europe/Paris	
Europe/Prague	
Europe/Sarajevo	
Pacific/Auckland	
Pacific/Guam	

Time zone (Zona horarioa)	Notas
Pacific/Honolulu	Este ajuste de zona horaria puede devolver valores incorrectos del 21 de mayo de 1933 a las 11:30:00 GMT hasta el 30 de septiembre de 1945 a las 11:30:00 GMT.
Pacific/Samoa	Este ajuste de zona horaria puede devolver valores incorrectos del 1 de enero de 1911 a las 11:22:48 GMT hasta el 1 de enero de 1950 a las 11:30:00 GMT.
US/Alaska	
US/Central	
US/Eastern	
US/East-Indiana	
US/Pacific	
UTC	

Funciones admitidas en Amazon Aurora por Región de AWS y el motor de base de datos de Aurora

Aurora Los motores de base de datos compatibles con MySQL y PostgreSQL admiten varias Amazon Aurora y Amazon RDS características y opciones. La compatibilidad varía según las versiones específicas de cada motor de base de datos y entre Regiones de AWS. Para identificar el soporte de la versión del motor de la base de datos de Aurora y la disponibilidad de una función en un Región de AWS determinado, puede utilizar las siguientes secciones.

Algunas de estas características son solo capacidades de Aurora. Por ejemplo, Aurora Serverless, las bases de datos de Aurora globales y la compatibilidad con la integración con los servicios de Machine Learning de AWS no son compatibles con Amazon RDS. Otras características, como el proxy de Amazon RDS, son compatibles con Amazon Aurora y Amazon RDS.

Regiones y motores de base de datos admitidos

- Convenciones de tabla
- Regiones y motores de base de datos de Aurora admitidos para implementaciones azul/verde
- Regiones y motores de base de datos Aurora admitidos para configuraciones de almacenamiento en clúster
- Regiones y motores de base de datos Aurora admitidos para los flujos de actividad de bases de datos
- Regiones y motores de base de datos Aurora admitidos para exportar datos del clúster a Amazon S3
- Regiones y motores de base de datos Aurora admitidos para exportar datos de instantáneas a Amazon S3
- Regiones y motores de base de datos admitidos para bases de datos globales Aurora
- Regiones y motores de base de datos Aurora admitidos para autenticación de bases de datos IAM
- Regiones y motores de base de datos Aurora admitidos para autenticación de Kerberos
- Regiones y motores de bases de datos admitidos para machine learning de Aurora
- Regiones y motores de base de datos Aurora para Información sobre rendimiento
- Regiones y motores de base de datos de Aurora admitidos para integraciones sin ETL
- Regiones y motores de base de datos Aurora para Amazon RDS Proxy
- Regiones y motores de bases de datos Aurora admitidos para la integración de Secrets Manager
- Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2

- Aurora Serverless v1
- Regiones y motores de base de datos Aurora admitidos para API de datos de RDS
- Regiones y motores de base de datos Aurora admitidos para aplicación de parches sin tiempo de inactividad (ZDP)
- Aurora PostgreSQL Limitless Database でサポートされているリージョン
- Regiones y motores de base de datos admitidos para características nativas del motor Aurora

Convenciones de tabla

Las tablas en las secciones de funciones utilizan los siguientes patrones para especificar los números de versión y el nivel de soporte:

- Versión x.y – Solo se admite la versión específica.
- Versión x.y y posteriores: se admiten la versión especificada y todas las versiones secundarias posteriores. Por ejemplo, “versión 10.11 y posteriores” significa que son compatibles las versiones 10.11, 10.11.1 y 10.12.

Regiones y motores de base de datos de Aurora admitidos para implementaciones azul/verde

Una implementación azul/verde copia un entorno de base de datos de producción en un entorno de almacenamiento provisional sincronizado e independiente. Con las implementaciones azul/verde de Amazon RDS, puede realizar cambios en la base de datos en el entorno de almacenamiento provisional sin que eso afecte al entorno de producción. Por ejemplo, puede actualizar la versión principal o secundaria del motor de base de datos, cambiar los parámetros de la base de datos o realizar cambios de esquema en el entorno de almacenamiento provisional. Cuando esté listo, puede promocionar el entorno de almacenamiento provisional para que sea el nuevo entorno de base de datos de producción. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Implementaciones azules/verdes con Aurora MySQL

La característica de implementaciones azul/verde está disponible para todas las versiones de Aurora MySQL en todas las Regiones de AWS, incluidos los clústeres de Aurora MySQL configurados como base de datos global de Aurora.

Implementaciones azules/verdes con Aurora PostgreSQL

Las siguientes regiones y versiones de motor están disponibles para implementaciones azul/verde con Aurora PostgreSQL, incluidos los clústeres de Aurora PostgreSQL configurados como base de datos global de Aurora.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Todas las Regiones de AWS	Versión 17.x y posteriores	Versión 16.x y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posteriores	Versión 13.6 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y posteriores

Regiones y motores de base de datos Aurora admitidos para configuraciones de almacenamiento en clúster

Amazon Aurora tiene dos configuraciones de almacenamiento en clústeres de base de datos: Aurora I/O-Optimized y Aurora Standard. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.

Aurora I/O-Optimized

Aurora I/O-Optimized está disponible en todas las Regiones de AWS para las siguientes versiones de Amazon Aurora:

- Aurora MySQL versión 3.03.1 y posteriores
- Aurora PostgreSQL versiones 16.1 y posteriores, 15.2 y posteriores, 14.7 y posteriores y 13.10 y posteriores

Aurora Standard

Aurora Standard está disponible en todas las Regiones de AWS y para todas las versiones de Aurora MySQL y Aurora PostgreSQL.

Regiones y motores de base de datos Aurora admitidos para los flujos de actividad de bases de datos

Utilizando las secuencias de actividad de la base de datos en Aurora, puede supervisar y establecer alarmas para auditar la actividad en su base de datos de Aurora. Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.

Las secuencias de actividades de base de datos no se admiten en las siguientes características:

- Aurora Serverless v1
- Aurora Serverless v2
- Babelfish para Aurora PostgreSQL

Temas

- Secuencias de actividades de la base de datos con Aurora MySQL
- Secuencias de actividades de la base de datos con Aurora PostgreSQL

Secuencias de actividades de la base de datos con Aurora MySQL

A continuación se detallan las regiones y las versiones de motores que están disponibles para secuencias de actividades de base de datos con Aurora MySQL.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Este de EE. UU. (Ohio)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Oeste de EE. UU. (Norte de California)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Oeste de EE. UU. (Oregón)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
África (Ciudad del Cabo)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Hong Kong)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Hyderabad)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Yakarta)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Malasia)	No disponible	No disponible
Asia-Pacífico (Melbourne)	No disponible	No disponible
Asia-Pacífico (Mumbai)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible
Asia-Pacífico (Osaka)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Seúl)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Singapur)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Sídney)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Tokio)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Canadá (centro)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Oeste de Canadá (Calgary)	No disponible	No disponible
China (Pekín)	No disponible	No disponible
China (Ningxia)	No disponible	No disponible
Europa (Fráncfort)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (Irlanda)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (Londres)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (Milán)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (París)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (España)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (Estocolmo)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
Europa (Zúrich)	No disponible	No disponible
Israel (Tel Aviv)	No disponible	No disponible

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Medio Oriente (Baréin)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
México (central)	No disponible	No disponible
Medio Oriente (EAU)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores
América del Sur (São Paulo)	Todas las versiones disponibles	Aurora versión 2.11 y posteriores

Secuencias de actividades de la base de datos con Aurora PostgreSQL

A continuación se detallan las regiones y las versiones de motores que están disponibles para secuencias de actividades de base de datos con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Este de EE. UU. (Ohio)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Oeste de EE. UU.	Versión 17. y	Versión 16. y	Versión 15.2 y	Todas las versiones	Todas las versiones	Todas las versiones	Versión 11.9 y

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
(Norte de California)	posteriores	posteriores	posteriores	disponibles	disponibles	disponibles	versión 11.13 y posteriores
Oeste de EE. UU. (Oregón)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
África (Ciudad del Cabo)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Hong Kong)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Hyderabad)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Yakarta)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Malasia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	
Asia-Pacífico (Melbourne)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Mumbai)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Seúl)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Singapur)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Sídney)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Tokio)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Canadá (centro)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Oeste de Canadá (Calgary)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
China (Pekín)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
China (Ningxia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Europa (Fráncfort)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Irlanda)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Europa (Londres)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Europa (Milán)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Europa (París)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (España)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Europa (Estocolmo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
Europa (Zúrich)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Israel (Tel Aviv)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (Baréin)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Medio Oriente (EAU)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores
América del Sur (São Paulo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Todas las versiones disponibles	Todas las versiones disponibles	Todas las versiones disponibles	Versión 11.9 y versión 11.13 y posteriores

Regiones y motores de base de datos Aurora admitidos para exportar datos del clúster a Amazon S3

Puede exportar datos de clústeres de bases de datos de Autora a un bucket de Amazon S3. Después de exportar los datos, puede analizar los datos exportados directamente con herramientas como Amazon Athena o Amazon Redshift Spectrum. Para obtener más información, consulte Exportación de datos del clúster de base de datos a Amazon S3.

La exportación de datos de clústeres a S3 está disponible en las siguientes Regiones de AWS:

- Este de EE. UU. (Norte de Virginia)
- Este de EE. UU. (Ohio)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Hyderabad)
- Asia-Pacífico (Yakarta)

- Asia-Pacífico (Melbourne)
- Asia-Pacífico (Mumbai)
- Asia-Pacífico (Nueva Zelanda)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Oeste de Canadá (Calgary)
- China (Ningxia)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (París)
- Europa (España)
- Europa (Estocolmo)
- Europa (Zúrich)
- Israel (Tel Aviv)
- Oriente Medio (EAU)
- América del Sur (São Paulo)

Temas

- Exportación de datos de clústeres a S3 con Aurora MySQL
- Exportación de datos de clústeres a S3 con Aurora PostgreSQL

Exportación de datos de clústeres a S3 con Aurora MySQL

Todas las versiones del motor de Aurora PostgreSQL disponibles en la actualidad admiten la exportación de datos de clústeres de bases de datos a Amazon S3. Para obtener más información sobre las versiones, consulte las Notas de la versión de Aurora MySQL.

Exportación de datos de clústeres a S3 con Aurora PostgreSQL

Todas las versiones del motor de Aurora PostgreSQL disponibles en la actualidad admiten la exportación de datos de clústeres de bases de datos a Amazon S3. Para obtener más información sobre las versiones, consulte las notas de la versión de Aurora PostgreSQL.

Regiones y motores de base de datos Aurora admitidos para exportar datos de instantáneas a Amazon S3

Puede exportar datos de instantáneas de clústeres de bases de datos de Aurora a un bucket de Amazon S3. Puede exportar instantáneas manuales e instantáneas del sistema automatizadas. Después de exportar los datos, puede analizar los datos exportados directamente con herramientas como Amazon Athena o Amazon Redshift Spectrum. Para obtener más información, consulte Exportación de datos de instantánea del clúster de bases de datos a Amazon S3.

La exportación de instantáneas a S3 está disponible en todas las Regiones de AWS excepto en las siguientes:

- Asia-Pacífico (Malasia)
- Asia-Pacífico (Nueva Zelanda)
- Asia-Pacífico (Taipéi)
- Asia-Pacífico (Tailandia)
- México (central)
- AWS GovCloud (Este de EE. UU.)
- AWS GovCloud (Oeste de EE. UU.)

Temas

- Exportación de datos de instantánas a S3 con Aurora MySQL
- Exportación de datos de instantáneas a S3 con Aurora PostgreSQL

Exportación de datos de instantánas a S3 con Aurora MySQL

Todas las versiones del motor de Aurora PostgreSQL disponibles en la actualidad admiten la exportación de datos de instantáneas de clústeres de bases de datos a Amazon S3. Para obtener más información sobre las versiones, consulte las Notas de la versión de Aurora MySQL.

Exportación de datos de instantáneas a S3 con Aurora PostgreSQL

Todas las versiones del motor de Aurora PostgreSQL disponibles en la actualidad admiten la exportación de datos de instantáneas de clústeres de base de datos a Amazon S3. Para obtener más información sobre las versiones, consulte las notas de la versión de Aurora PostgreSQL.

Regiones y motores de base de datos admitidos para bases de datos globales Aurora

Una base de datos global de Aurora es una sola base de datos que abarca varias Regiones de AWS, lo que permite lecturas globales de baja latencia y la recuperación de desastres provocados por las interrupciones de suministro de energía eléctrica que afectan regiones enteras. Proporciona tolerancia a fallos integrada para la implementación porque la instancia de base de datos no depende de una sola Región de AWS, sino de varias regiones y zonas de disponibilidad diferentes. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

Temas

- Bases de datos globales de Aurora con Aurora MySQL
- Bases de datos globales de Aurora con Aurora PostgreSQL

Bases de datos globales de Aurora con Aurora MySQL

A continuación, se detallan las regiones y las versiones del motor que están disponibles para las bases de datos globales de Aurora con Aurora MySQL.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Este de EE. UU. (Ohio)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Oeste de EE. UU. (Norte de California)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Oeste de EE. UU. (Oregón)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
África (Ciudad del Cabo)	Versión 3.01.0 y posterior	Versión 2.07.1 y posterior

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Hong Kong)	Versión 3.01.0 y posterior	Versión 2.07.1 y posterior
Asia-Pacífico (Hyderabad)	Versión 3.02.0 y posterior	Versión 2.11.2 y posteriores
Asia-Pacífico (Yakarta)	Versión 3.01.0 y posterior	Versión 2.07.6 y posteriores
Asia-Pacífico (Malasia)	Versión 3.04.0 y posteriores	Versión 2.07.6 y posteriores
Asia-Pacífico (Melbourne)	Versión 3.03.0 y posteriores	Versión 2.07.6 y posteriores
Asia-Pacífico (Mumbai)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 3.01.0 y posterior	Versión 2.07.3 y posterior
Asia-Pacífico (Seúl)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Asia-Pacífico (Singapur)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Asia-Pacífico (Sídney)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Asia-Pacífico (Taipéi)	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible
Asia-Pacífico (Tokio)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Canadá (centro)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Oeste de Canadá (Calgary)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
China (Pekín)	Versión 3.01.0 y posterior	Versión 2.07.2 y posterior
China (Ningxia)	Versión 3.01.0 y posterior	Versión 2.07.2 y posterior
Europa (Fráncfort)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Europa (Irlanda)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Europa (Londres)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Europa (Milán)	Versión 3.01.0 y posterior	Versión 2.07.1 y posterior
Europa (París)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Europa (España)	Versión 3.02.0 y posterior	Versión 2.07.6 y posteriores
Europa (Estocolmo)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
Europa (Zúrich)	Versión 3.02.0 y posterior	Versión 2.07.6 y posteriores
Israel (Tel Aviv)	Versión 3.02.0 y posterior	Versión 2.07.6 y posteriores
México (central)	No disponible	No disponible
Medio Oriente (Baréin)	Versión 3.01.0 y posterior	Versión 2.07.1 y posterior
Medio Oriente (EAU)	Versión 3.02.0 y posterior	Versión 2.07.6 y posteriores
América del Sur (São Paulo)	Versión 3.01.0 y posterior	Versión 2.07.1 y posterior
AWS GovCloud (Este de EE. UU.)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior
AWS GovCloud (Oeste de EE. UU.)	Versión 3.01.0 y posterior	Versión 2.07.0 y posterior

Bases de datos globales de Aurora con Aurora PostgreSQL

A continuación, se detallan las regiones y las versiones del motor que están disponibles para las bases de datos globales de Aurora con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Este de EE. UU. (Ohio)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Oeste de EE. UU. (Oregón)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
África (Ciudad del Cabo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
							posteriores
Asia-Pacífico (Hong Kong)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Hyderabad)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Yakarta)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Malasia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Melbourne)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Mumbai)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Seúl)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Singapur)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Sídney)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Sídney)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tokio)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canadá (centro)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Oeste de Canadá (Calgary)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
China (Pekín)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
China (Ningxia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Fráncfort)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Irlanda)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Londres)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Milán)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (París)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (España)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Estocolmo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Zúrich)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Israel (Tel Aviv)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (Baréin)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Medio Oriente (EAU)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
América del Sur (São Paulo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
AWS GovCloud (Este de EE. UU.)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Oeste de EE. UU.)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Regiones y motores de base de datos Aurora admitidos para autenticación de bases de datos IAM

Con la autenticación de base de datos IAM en Aurora, puede autenticarse en su clúster de base de datos utilizando la autenticación de base de datos de Identity and Access Management (IAM). Con este método de autenticación, no es necesario usar una contraseña al conectarse a un clúster de bases de datos. En su lugar, puede usar un token de autenticación. Para obtener más información, consulte Autenticación de bases de datos de IAM .

Temas

- Autenticación de base de datos de IAM con Aurora MySQL.
- Autenticación de bases de datos de IAM con Aurora PostgreSQL

Autenticación de base de datos de IAM con Aurora MySQL.

La autenticación de bases de datos de IAM con Aurora MySQL está disponible en todas las regiones para las siguientes versiones:

- Aurora MySQL 3: todas las versiones disponibles
- Aurora MySQL 2: todas las versiones disponibles

Autenticación de bases de datos de IAM con Aurora PostgreSQL

La autenticación de bases de datos de IAM con Aurora PostgreSQL está disponible en todas las regiones para las siguientes versiones del motor:

- Aurora PostgreSQL 16: todas las versiones disponibles
- Aurora PostgreSQL 15: todas las versiones disponibles
- Aurora PostgreSQL 14: todas las versiones disponibles
- Aurora PostgreSQL 13: todas las versiones disponibles
- Aurora PostgreSQL 12: todas las versiones disponibles
- Aurora PostgreSQL 11: todas las versiones disponibles

Regiones y motores de base de datos Aurora admitidos para autenticación de Kerberos

Con la autenticación Kerberos con Aurora, puede admitir la autenticación externa de usuarios de bases de datos mediante Kerberos y Microsoft Active Directory. El uso de Kerberos y Microsoft Active Directory permite usar el inicio de sesión único y la autenticación centralizada de usuarios de bases de datos. Kerberos y Active Directory están disponibles con AWS Directory Service for Microsoft Active Directory, una función de Directory Service. Para obtener más información, consulte Autenticación Kerberos.

Temas

- Autenticación Kerberos con Aurora MySQL
- Autenticación Kerberos con con Aurora PostgreSQL
- Grupos de seguridad de Active Directory (AD) con Aurora PostgreSQL

Autenticación Kerberos con Aurora MySQL

A continuación se detallan las regiones y las versiones de motores que están disponibles para la autenticación Kerberos con Aurora MySQL.

Región	Aurora MySQL versión 3
Este de EE. UU. (Norte de Virginia)	Versión 3.03.0 y posteriores
Este de EE. UU. (Ohio)	Versión 3.03.0 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 3.03.0 y posteriores

Región	Aurora MySQL versión 3
Oeste de EE. UU. (Oregón)	Versión 3.03.0 y posteriores
África (Ciudad del Cabo)	No disponible
Asia-Pacífico (Hong Kong)	No disponible
Asia-Pacífico (Yakarta)	No disponible
Asia-Pacífico (Malasia)	No disponible
Asia-Pacífico (Melbourne)	No disponible
Asia-Pacífico (Mumbai)	Versión 3.03.0 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible
Asia-Pacífico (Osaka)	No disponible
Asia-Pacífico (Seúl)	Versión 3.03.0 y posteriores
Asia-Pacífico (Singapur)	Versión 3.03.0 y posteriores
Asia-Pacífico (Sídney)	Versión 3.03.0 y posteriores
Asia-Pacífico (Taipéi)	No disponible
Asia-Pacífico (Tailandia)	No disponible
Asia-Pacífico (Tokio)	Versión 3.03.0 y posteriores
Canadá (centro)	Versión 3.03.0 y posteriores
Oeste de Canadá (Calgary)	No disponible
China (Pekín)	Versión 3.03.0 y posteriores
China (Ningxia)	Versión 3.03.0 y posteriores
Europa (Fráncfort)	Versión 3.03.0 y posteriores

Región	Aurora MySQL versión 3
Europa (Irlanda)	Versión 3.03.0 y posteriores
Europa (Londres)	Versión 3.03.0 y posteriores
Europa (Milán)	No disponible
Europa (París)	Versión 3.03.0 y posteriores
Europa (España)	No disponible
Europa (Estocolmo)	Versión 3.03.0 y posteriores
Europa (Zúrich)	No disponible
Israel (Tel Aviv)	No disponible
México (central)	No disponible
Medio Oriente (Baréin)	No disponible
Medio Oriente (EAU)	No disponible
América del Sur (São Paulo)	Versión 3.03.0 y posteriores
AWS GovCloud (Este de EE. UU.)	Versión 3.03.0 y posteriores
AWS GovCloud (Oeste de EE. UU.)	Versión 3.03.0 y posteriores

Autenticación Kerberos con con Aurora PostgreSQL

A continuación se detallan las regiones y las versiones de motores que están disponibles para la autenticación Kerberos con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Este de EE. UU. (Ohio)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Norte de California)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Oregón)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
África (Ciudad del Cabo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Hong Kong)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Hyderabad)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Yakarta)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Malasia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Melbourne)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Mumbai)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Seúl)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Singapur)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Sídney)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tokio)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Canadá (centro)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de Canadá (Calgary)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
China (Pekín)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
China (Ningxia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Fráncfort)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Irlanda)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Londres)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Milán)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (París)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (España)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Estocolmo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Zúrich)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Israel (Tel Aviv)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (Baréin)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Medio Oriente (EAU)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
América del Sur (São Paulo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
AWS GovCloud (Este de EE. UU.)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
AWS GovCloud (Oeste de EE. UU.)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Grupos de seguridad de Active Directory (AD) con Aurora PostgreSQL

A continuación, se detallan las regiones y las versiones de motores que están disponibles para ActiveDirectory con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Ohio)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Norte de California)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Oregón)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
África (Ciudad del Cabo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Hong Kong)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Hyderabad)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Yakarta)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Malasia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Melbourne)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Mumbai)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Seúl)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Singapur)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Sídney)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tokio)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Canadá (centro)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de Canadá (Calgary)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
China (Pekín)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
China (Ningxia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Fráncfort)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Irlanda)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Londres)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Milán)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (París)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (España)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Estocolmo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Zúrich)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Israel (Tel Aviv)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (Baréin)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Medio Oriente (EAU)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
América del Sur (São Paulo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Este de EE. UU.)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
AWS GovCloud (Oeste de EE. UU.)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Regiones y motores de bases de datos admitidos para machine learning de Aurora

Mediante el machine learning de Amazon Aurora, puede integrar su clúster de base de datos de Aurora con uno de los siguientes servicios de machine learning de AWS, según sus necesidades. Cada uno admite casos de uso de machine learning específicos.

Amazon Bedrock es un servicio totalmente administrado que ofrece los principales modelos básicos de las empresas de IA a través de una API, junto con herramientas para desarrolladores que permiten crear y escalar aplicaciones de IA generativa.

Amazon Comprehend es un servicio de procesamiento de lenguaje natural (NLP) que se utiliza para extraer información de los documentos. Al utilizar el machine learning de Aurora con Amazon Comprehend, puede determinar la opinión del texto en las tablas de la base de datos.

IA de SageMaker es un servicio de machine learning completamente administrado. Los científicos de datos utilizan IA de Amazon SageMaker para crear, entrenar y probar modelos de machine learning para una variedad de tareas de inferencia, como la detección de fraudes. Al utilizar el machine learning de Aurora con IA de SageMaker, los desarrolladores de bases de datos pueden invocar la funcionalidad de IA de SageMaker en código SQL.

No todas las Regiones de AWS son compatibles con todos los servicios de machine learning. Solo algunas Regiones de AWS admiten el machine learning de Aurora y, por lo tanto, proporcionan

acceso a estos servicios desde un clúster de base de datos de Aurora. El proceso de integración del machine learning de Aurora también difiere según el motor de base de datos. Para obtener más información, consulte Uso de machine learning de Amazon Aurora.

Temas

- Machine Learning de Aurora con Aurora MySQL
- Machine learning de Aurora con Aurora PostgreSQL

Machine Learning de Aurora con Aurora MySQL

Amazon Bedrock solo es compatible con Aurora MySQL versión 3.06 y posteriores. Para obtener información sobre la disponibilidad en regiones de Amazon Bedrock, consulte Model support by Región de AWS en la Guía del usuario de Amazon Bedrock.

El machine learning de Aurora con Amazon Comprehend e IA de Amazon SageMaker es compatible con Aurora MySQL en las Regiones de AWS incluidas en la tabla. Además de tener disponible su versión de Aurora MySQL, también las Región de AWS deben admitir el servicio que desee utilizar. Para obtener una lista de las Regiones de AWS en las que está disponible IA de Amazon SageMaker, consulte Amazon SageMaker AI endpoints and quotas en la Referencia general de Amazon Web Services. Para obtener una lista de las Regiones de AWS en las que Amazon Comprehend está disponible, consulte Amazon Comprehend endpoints and quotas, en la Referencia general de Amazon Web Services.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Este de EE. UU. (Ohio)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Oeste de EE. UU. (Norte de California)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Oeste de EE. UU. (Oregón)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
África (Ciudad del Cabo)	No disponible	No disponible
Asia-Pacífico (Hong Kong)	Versión 3.01.0 y posterior	Versión 2.07 y posterior

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Hyderabad)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Yakarta)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Malasia)	Versión 3.04.0 y posteriores	No disponible
Asia-Pacífico (Melbourne)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Mumbai)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 3.01.0 y posterior	Versión 2.07.3 y posterior
Asia-Pacífico (Seúl)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Singapur)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Sídney)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Asia-Pacífico (Taipéi)	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible
Asia-Pacífico (Tokio)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Canadá (centro)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Oeste de Canadá (Calgary)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
China (Pekín)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
China (Ningxia)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Europa (Fráncfort)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Europa (Irlanda)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Europa (Londres)	Versión 3.01.0 y posterior	Versión 2.07 y posterior

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Europa (Milán)	No disponible	No disponible
Europa (París)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Europa (España)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Europa (Estocolmo)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Europa (Zúrich)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Israel (Tel Aviv)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
México (central)	No disponible	No disponible
Medio Oriente (Baréin)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
Medio Oriente (EAU)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
América del Sur (São Paulo)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
AWS GovCloud (Este de EE. UU.)	Versión 3.01.0 y posterior	Versión 2.07 y posterior
AWS GovCloud (Oeste de EE. UU.)	Versión 3.01.0 y posterior	Versión 2.07 y posterior

Machine learning de Aurora con Aurora PostgreSQL

Para obtener información sobre la compatibilidad de las versiones para Amazon Bedrock en Aurora PostgreSQL, consulte Uso de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock.

El machine learning de Aurora con Amazon Comprehend e IA de Amazon SageMaker es compatible con Aurora PostgreSQL en las Regiones de AWS incluidas en la tabla. Además de tener disponible su versión de Aurora PostgreSQL, también las Región de AWS deben admitir el servicio que desee utilizar. Para obtener una lista de las Regiones de AWS en las que está disponible IA de Amazon SageMaker, consulte Amazon SageMaker AI endpoints and quotas en la Referencia general de Amazon Web Services. Para obtener una lista de las Regiones de AWS en las que Amazon

Comprehend está disponible, consulte Amazon Comprehend endpoints and quotas, en la Referencia general de Amazon Web Services.

A continuación se detallan las regiones y las versiones de motores que están disponibles para machine learning de Aurora con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Este de EE. UU. (Ohio)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Oeste de EE. UU. (Oregón)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
África (Ciudad del Cabo)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Hong Kong)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Hyderabad)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Yakarta)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Malasia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Melbourne)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Mumbai)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Nueva Zelanda)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Osaka)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Seúl)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Singapur)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Sídney)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Taipéi)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Tailandia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Asia-Pacífico (Tokio)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Canadá (centro)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Oeste de Canadá (Calgary)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
China (Pekín)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
China (Ningxia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Europa (Fráncfort)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Europa (Irlanda)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Londres)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Europa (Milán)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Europa (París)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Europa (España)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Europa (Estocolmo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Europa (Zúrich)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Israel (Tel Aviv)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (Baréin)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
Medio Oriente (EAU)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
América del Sur (São Paulo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores
AWS GovCloud (Este de EE. UU.)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Oeste de EE. UU.)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.3 y posterior	Versión 12.4 y posteriores	Versión 11.9, 11.12 y posteriores

Regiones y motores de base de datos Aurora para Información sobre rendimiento

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.

Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.

Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.

Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte

Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Performance Insights amplía las características de supervisión de Amazon RDS existentes para informarle y ayudarle a analizar el rendimiento de su base de datos. En el panel de Performance Insights puede visualizar la carga de la base de datos en su carga de instancia de base de datos de Amazon RDS y filtrarla por esperas, instrucciones SQL, hosts o usuarios. Para obtener más información, consulte Uso de Performance Insights en Amazon Aurora.

Para obtener información sobre la compatibilidad de las características de la Información de rendimiento por región, motor de base de datos y clase de instancia, consulte Compatibilidad del motor de la base de datos, la región y la clase de instancia de Amazon Aurora con características de Información de rendimiento.

Temas

- Performance Insights con Aurora MySQL
- Performance Insights con Aurora PostgreSQL
- Performance Insights con Aurora Serverless

Performance Insights con Aurora MySQL

> **Note**
>
> La compatibilidad con la versión del motor es diferente para Performance Insights con Aurora MySQL si tiene la consulta en paralelo activada. Para obtener más información sobre la consulta en paralelo, consulte Consulta paralela para Amazon Aurora MySQL.

Temas

- Performance Insights con Aurora MySQL y consulta en paralelo desactivadas
- Performance Insights con Aurora MySQL y consulta en paralelo activadas

Performance Insights con Aurora MySQL y consulta en paralelo desactivadas

A continuación, se detallan las regiones y versiones del motor que están disponibles para Información sobre rendimiento con Aurora MySQL y la consulta en paralelo desactivada.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Todas las versiones	Todas las versiones
Este de EE. UU. (Ohio)	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Norte de California)	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Oregón)	Todas las versiones	Todas las versiones
África (Ciudad del Cabo)	Todas las versiones	Todas las versiones
Asia-Pacífico (Hong Kong)	Todas las versiones	Todas las versiones
Asia-Pacífico (Hyderabad)	Todas las versiones	Todas las versiones
Asia-Pacífico (Yakarta)	Todas las versiones	Todas las versiones
Asia-Pacífico (Malasia)	Todas las versiones	Todas las versiones
Asia-Pacífico (Melbourne)	Todas las versiones	Todas las versiones
Asia-Pacífico (Mumbai)	Todas las versiones	Todas las versiones
Asia-Pacífico (Nueva Zelanda)	Todas las versiones	Todas las versiones
Asia-Pacífico (Osaka)	Todas las versiones	Todas las versiones
Asia-Pacífico (Seúl)	Todas las versiones	Todas las versiones
Asia-Pacífico (Singapur)	Todas las versiones	Todas las versiones
Asia-Pacífico (Sídney)	Todas las versiones	Todas las versiones
Asia-Pacífico (Taipéi)	Todas las versiones	Todas las versiones

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Tailandia)	Todas las versiones	Todas las versiones
Asia-Pacífico (Tokio)	Todas las versiones	Todas las versiones
Canadá (centro)	Todas las versiones	Todas las versiones
Oeste de Canadá (Calgary)	Todas las versiones	Todas las versiones
China (Pekín)	Todas las versiones	Todas las versiones
China (Ningxia)	Todas las versiones	Todas las versiones
Europa (Fráncfort)	Todas las versiones	Todas las versiones
Europa (Irlanda)	Todas las versiones	Todas las versiones
Europa (Londres)	Todas las versiones	Todas las versiones
Europa (Milán)	Todas las versiones	Todas las versiones
Europa (París)	Todas las versiones	Todas las versiones
Europa (España)	Todas las versiones	Todas las versiones
Europa (Estocolmo)	Todas las versiones	Todas las versiones
Europa (Zúrich)	Todas las versiones	Todas las versiones
Israel (Tel Aviv)	Todas las versiones	Todas las versiones
México (central)	Todas las versiones	Todas las versiones
Medio Oriente (Baréin)	Todas las versiones	Todas las versiones
Medio Oriente (EAU)	Todas las versiones	Todas las versiones
América del Sur (São Paulo)	Todas las versiones	Todas las versiones
AWS GovCloud (Este de EE. UU.)	Todas las versiones	Todas las versiones

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
AWS GovCloud (Oeste de EE. UU.)	Todas las versiones	Todas las versiones

Performance Insights con Aurora MySQL y consulta en paralelo activadas

A continuación, se detallan las regiones y versiones del motor que están disponibles para Información sobre rendimiento con Aurora MySQL y la consulta en paralelo activada.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	No disponible	Versión 2.09.0 y posterior
Este de EE. UU. (Ohio)	No disponible	Versión 2.09.0 y posterior
Oeste de EE. UU. (Norte de California)	No disponible	Versión 2.09.0 y posterior
Oeste de EE. UU. (Oregón)	No disponible	Versión 2.09.0 y posterior
África (Ciudad del Cabo)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Hong Kong)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Hyderabad)	No disponible	Todas las versiones
Asia-Pacífico (Yakarta)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Malasia)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Melbourne)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Mumbai)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Nueva Zelanda)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Osaka)	No disponible	Versión 2.09.0 y posterior

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Seúl)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Singapur)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Sídney)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Taipéi)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Tailandia)	No disponible	Versión 2.09.0 y posterior
Asia-Pacífico (Tokio)	No disponible	Versión 2.09.0 y posterior
Canadá (centro)	No disponible	Versión 2.09.0 y posterior
Oeste de Canadá (Calgary)	No disponible	Versión 2.09.0 y posterior
China (Pekín)	No disponible	Versión 2.09.0 y posterior
China (Ningxia)	No disponible	Versión 2.09.0 y posterior
Europa (Fráncfort)	No disponible	Versión 2.09.0 y posterior
Europa (Irlanda)	No disponible	Versión 2.09.0 y posterior
Europa (Londres)	No disponible	Versión 2.09.0 y posterior
Europa (Milán)	No disponible	Versión 2.09.0 y posterior
Europa (París)	No disponible	Versión 2.09.0 y posterior
Europa (España)	No disponible	Versión 2.09.0 y posterior
Europa (Estocolmo)	No disponible	Versión 2.09.0 y posterior
Europa (Zúrich)	No disponible	Versión 2.09.0 y posterior
Israel (Tel Aviv)	No disponible	Versión 2.09.0 y posterior
México (central)	No disponible	Versión 2.09.0 y posterior

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Medio Oriente (Baréin)	No disponible	Versión 2.09.0 y posterior
Medio Oriente (EAU)	No disponible	Versión 2.09.0 y posterior
América del Sur (São Paulo)	No disponible	Versión 2.09.0 y posterior
AWS GovCloud (Este de EE. UU.)	No disponible	Versión 2.09.0 y posterior
AWS GovCloud (Oeste de EE. UU.)	No disponible	Versión 2.09.0 y posterior

Performance Insights con Aurora PostgreSQL

A continuación, se detallan las regiones y las versiones del motor que están disponibles para Información sobre rendimiento con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Este de EE. UU. (Norte de Virginia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Este de EE. UU. (Ohio)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Norte de	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
California)								
Oeste de EE. UU. (Oregón)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
África (Ciudad del Cabo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Hong Kong)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Hyderabad)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Yakarta)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Malasia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Melbourne)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asia-Pacífico (Mumbai)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Nueva Zelanda)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Osaka)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Seúl)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Singapur)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Sídney)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Taipéi)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Asia-Pacífico (Tailandia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asia-Pacífico (Tokio)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Canadá (centro)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Oeste de Canadá (Calgary)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
China (Pekín)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
China (Ningxia)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Fráncfort)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Irlanda)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Londres)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Europa (Milán)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (París)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (España)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Estocolmo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Europa (Zúrich)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Israel (Tel Aviv)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
México (central)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Medio Oriente (Baréin)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
Medio Oriente (EAU)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
América del Sur (São Paulo)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
AWS GovCloud (Este de EE. UU.)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones
AWS GovCloud (Oeste de EE. UU.)	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones	Todas las versiones

Performance Insights con Aurora Serverless

Aurora Serverless v2 admite Performance Insights para todas las versiones compatibles con MySQL y PostgreSQL. Se recomienda establecer la capacidad mínima en al menos 2 unidades de capacidad (ACU) de Aurora.

Aurora Serverless v1no es compatible con Performance Insights.

Regiones y motores de base de datos de Aurora admitidos para integraciones sin ETL

Las integraciones sin ETL de Amazon Aurora son una solución totalmente administrada que permite que los datos transaccionales estén disponibles en Amazon Redshift o Amazon SageMaker después de escribirlos en un clúster de Aurora. Para obtener más información, consulte Integraciones sin ETL.

Temas

- Integraciones sin ETL de Aurora MySQL

- Integraciones sin ETL de Aurora PostgreSQL

Integraciones sin ETL de Aurora MySQL

Las siguientes regiones y versiones de motores están disponibles para las integraciones sin ETL de Aurora MySQL con Amazon Redshift y Amazon SageMaker.

Región	Integración sin ETL con Amazon Redshift para Aurora MySQL versión 3	Integración sin ETL con Amazon SageMaker para Aurora MySQL versión 3
Este de EE. UU. (Norte de Virginia)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Este de EE. UU. (Ohio)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 3.05.2 y posteriores	No disponible
Oeste de EE. UU. (Oregón)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
África (Ciudad del Cabo)	Versión 3.05.2 y posteriores	No disponible
Asia-Pacífico (Hong Kong)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Asia-Pacífico (Hyderabad)	Versión 3.05.2 y posteriores	No disponible
Asia-Pacífico (Yakarta)	Versión 3.05.2 y posteriores	No disponible
Asia-Pacífico (Malasia)	No disponible	No disponible
Asia-Pacífico (Melbourne)	Versión 3.05.2 y posteriores	No disponible
Asia-Pacífico (Mumbai)	Versión 3.05.2 y posteriores	No disponible
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 3.05.2 y posteriores	No disponible
Asia-Pacífico (Seúl)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores

Región	Integración sin ETL con Amazon Redshift para Aurora MySQL versión 3	Integración sin ETL con Amazon SageMaker para Aurora MySQL versión 3
Asia-Pacífico (Singapur)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Asia-Pacífico (Sídney)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible
Asia-Pacífico (Tailandia)	No disponible	No disponible
Asia-Pacífico (Tokio)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Canadá (centro)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Oeste de Canadá (Calgary)	Versión 3.05.2 y posteriores	No disponible
China (Pekín)	Versión 3.05.2 y posteriores	No disponible
China (Ningxia)	Versión 3.05.2 y posteriores	No disponible
Europa (Fráncfort)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Europa (Irlanda)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Europa (Londres)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Europa (Milán)	Versión 3.05.2 y posteriores	No disponible
Europa (París)	Versión 3.05.2 y posteriores	No disponible
Europa (España)	Versión 3.05.2 y posteriores	No disponible
Europa (Estocolmo)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
Europa (Zúrich)	Versión 3.05.2 y posteriores	No disponible
Israel (Tel Aviv)	Versión 3.05.2 y posteriores	No disponible
México (central)	No disponible	No disponible

Región	Integración sin ETL con Amazon Redshift para Aurora MySQL versión 3	Integración sin ETL con Amazon SageMaker para Aurora MySQL versión 3
Medio Oriente (Baréin)	Versión 3.05.2 y posteriores	No disponible
Medio Oriente (EAU)	Versión 3.05.2 y posteriores	No disponible
América del Sur (São Paulo)	Versión 3.05.2 y posteriores	Versión 3.05.2 y posteriores
AWS GovCloud (Este de EE. UU.)	No disponible	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible	No disponible

Integraciones sin ETL de Aurora PostgreSQL

Las siguientes regiones y versiones de motores están disponibles para las integraciones sin ETL de Aurora PostgreSQL con Amazon Redshift y Amazon Sagemaker.

Región	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 16	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 16
Este de EE. UU. (Ohio)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Este de EE. UU. (Norte de Virginia)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible

Región	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 16	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 16
Oeste de EE. UU. (Oregón)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
África (Ciudad del Cabo)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Asia-Pacífico (Hong Kong)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Asia-Pacífico (Hyderabad)	Versión 17.4 y posteriores	Versión 16.4 y posteriores	No disponible	No disponible
Asia-Pacífico (Yakarta)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Asia-Pacífico (Melbourne)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Asia-Pacífico (Malasia)	No disponible	No disponible		
Asia-Pacífico (Malasia)	No disponible	No disponible		
Asia-Pacífico (Mumbai)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible		

Región	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 16	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 16
Asia-Pacífico (Osaka)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Asia-Pacífico (Seúl)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Asia-Pacífico (Singapur)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Asia-Pacífico (Sídney)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible		
Asia-Pacífico (Tailandia)	No disponible	No disponible		
Asia-Pacífico (Taipéi)	No disponible	No disponible		
Asia-Pacífico (Tailandia)	No disponible	No disponible		
Asia-Pacífico (Tokio)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Canadá (centro)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores

Región	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 16	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 16
Oeste de Canadá (Calgary)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
China (Pekín)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
China (Ningxia)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Europa (Fráncfort)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Europa (Irlanda)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Europa (Londres)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
Europa (Milán)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Europa (París)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Europa (España)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Europa (Estocolmo)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores

Región	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 17	Integración sin ETL con Amazon Redshift para Aurora PostgreSQL versión 16	Integración sin ETL con Amazon SageMaker para Aurora PostgreSQL versión 16
Europa (Zúrich)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Israel (Tel Aviv)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
México (central)	No disponible	No disponible		
México (central)	No disponible	No disponible		
Medio Oriente (Baréin)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
Medio Oriente (EAU)	Versión 17.4 y posteriores	No disponible	Versión 16.4 y posteriores	No disponible
América del Sur (São Paulo)	Versión 17.4 y posteriores	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 16.4 y posteriores
AWS GovCloud (Este de EE. UU.)	No disponible	No disponible	No disponible	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible	No disponible	No disponible	No disponible

Regiones y motores de base de datos Aurora para Amazon RDS Proxy

El proxy de Amazon RDS es un proxy de base de datos totalmente administrado y de alta disponibilidad que hace que las aplicaciones sean más escalables al agrupar y compartir conexiones

de base de datos establecidas. Para obtener más información sobre RDS Proxy, consulte Amazon RDS Proxy para Aurora.

Temas

- Proxy de Amazon RDS con Aurora MySQL
- Proxy de Amazon RDS con Aurora PostgreSQL

Proxy de Amazon RDS con Aurora MySQL

A continuación se detallan las regiones y las versiones del motor que están disponibles para el proxy de RDS con Aurora MySQL.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Este de EE. UU. (Ohio)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Oeste de EE. UU. (Oregón)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
África (Ciudad del Cabo)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Hong Kong)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Hyderabad)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Yakarta)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Malasia)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Melbourne)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Mumbai)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Seúl)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Singapur)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Sídney)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible
Asia-Pacífico (Tailandia)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Asia-Pacífico (Tokio)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Canadá (centro)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Oeste de Canadá (Calgary)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
China (Pekín)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
China (Ningxia)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (Fráncfort)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (Irlanda)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (Londres)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (Milán)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (París)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (España)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (Estocolmo)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Europa (Zúrich)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Israel (Tel Aviv)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
México (central)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
Medio Oriente (Baréin)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Medio Oriente (EAU)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
América del Sur (São Paulo)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
AWS GovCloud (Este de EE. UU.)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores
AWS GovCloud (Oeste de EE. UU.)	Versión 3.01.0 y posterior	Versión 2.07 y versión 2.11 y posteriores

Proxy de Amazon RDS con Aurora PostgreSQL

A continuación, se detallan las regiones y las versiones del motor que están disponibles para el proxy de RDS con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Este de EE. UU. (Ohio)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Oeste de EE. UU. (Norte de California)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Oeste de EE. UU. (Oregón)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
África (Ciudad del Cabo)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Hong Kong)	Versión 17. y posteriores	Versión 16. y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Hyderabad)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Yakarta)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Malasia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Melbourne)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Mumbai)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Seúl)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Singapur)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia-Pacífico (Sídney)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tailandia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Asia-Pacífico (Tokio)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Canadá (centro)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Oeste de Canadá (Calgary)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
China (Pekín)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
China (Ningxia)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Fráncfort)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Irlanda)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Londres)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Milán)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (París)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (España)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Estocolmo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Europa (Zúrich)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Israel (Tel Aviv)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
México (central)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Medio Oriente (Baréin)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
Medio Oriente (EAU)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
América del Sur (São Paulo)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Este de EE. UU.)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores
AWS GovCloud (Oeste de EE. UU.)	Versión 17.y posteriores	Versión 16.y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.4 y posteriores	Versión 12.8 y posteriores	Versión 11.9 y versión 11.13 y posteriores

Regiones y motores de bases de datos Aurora admitidos para la integración de Secrets Manager

Con AWS Secrets Manager, puede reemplazar las credenciales con codificación rígida (incluidas las contraseñas de bases de datos), con una llamada a la API de Secrets Manager para recuperar el secreto mediante programación. Para obtener más información acerca de Secrets Manager, consulte la Guía del usuario de AWS Secrets Manager.

Puede especificar que Amazon Aurora administre la contraseña de usuario maestra en Secrets Manager para un clúster de base de datos de Aurora. Aurora genera la contraseña, la almacena en Secrets Manager y la rota periódicamente. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.

La integración de Secrets Manager está disponible en todas las Regiones de AWS.

Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2

Aurora Serverless v2 es una característica de Auto Scaling en diferido diseñada para ser un enfoque rentable para ejecutar cargas de trabajo intermitentes o impredecibles en Amazon Aurora. Escala la capacidad vertical y horizontalmente de forma automática en función de las necesidades de las aplicaciones. El escalado es más rápido y granular que con Aurora Serverless v1. Con Aurora Serverless v2, cada clúster puede contener una instancia de base de datos de escritura y varias instancias de base de datos de lector Puede combinar instancias de base de datos aprovisionadas Aurora Serverless v2 y tradicionales dentro del mismo clúster. Para obtener más información, consulte Uso de Aurora Serverless v2.

Temas

- Aurora Serverless v2 con Aurora MySQL
- Aurora Serverless v2 con Aurora PostgreSQL

Aurora Serverless v2 con Aurora MySQL

A continuación, se detallan las regiones y las versiones del motor que están disponibles para Aurora Serverless v2 con Aurora MySQL.

Región	Aurora MySQL versión 3
Este de EE. UU. (Norte de Virginia)	Versión 3.02.0 y posterior
Este de EE. UU. (Ohio)	Versión 3.02.0 y posterior
Oeste de EE. UU. (Norte de California)	Versión 3.02.0 y posterior
Oeste de EE. UU. (Oregón)	Versión 3.02.0 y posterior
África (Ciudad del Cabo)	Versión 3.02.0 y posterior
Asia-Pacífico (Hong Kong)	Versión 3.02.0 y posterior
Asia-Pacífico (Hyderabad)	Versión 3.02.3 y posteriores
Asia-Pacífico (Yakarta)	Versión 3.02.0 y posterior

Región	Aurora MySQL versión 3
Asia-Pacífico (Malasia)	Versiones 3.04.3, 3.05.2, 3.06.1, 3.07.1 y posteriores
Asia-Pacífico (Melbourne)	Versión 3.02.3 y posteriores
Asia-Pacífico (Mumbai)	Versión 3.02.0 y posterior
Asia-Pacífico (Nueva Zelanda)	No disponible
Asia-Pacífico (Osaka)	Versión 3.02.0 y posterior
Asia-Pacífico (Seúl)	Versión 3.02.0 y posterior
Asia-Pacífico (Singapur)	Versión 3.02.0 y posterior
Asia-Pacífico (Sídney)	Versión 3.02.0 y posterior
Asia-Pacífico (Taipéi)	No disponible
Asia-Pacífico (Tailandia)	Versiones 3.04.3 y posteriores, 3.08.0 y posteriores
Asia-Pacífico (Tokio)	Versión 3.02.0 y posterior
Canadá (centro)	Versión 3.02.0 y posterior
Oeste de Canadá (Calgary)	Versión 3.04.0 y posteriores
China (Pekín)	Versión 3.02.2 y posteriores
China (Ningxia)	Versión 3.02.2 y posteriores
Europa (Fráncfort)	Versión 3.02.0 y posterior
Europa (Irlanda)	Versión 3.02.0 y posterior
Europa (Londres)	Versión 3.02.0 y posterior
Europa (Milán)	Versión 3.02.0 y posterior

Región	Aurora MySQL versión 3
Europa (París)	Versión 3.02.0 y posterior
Europa (España)	Versión 3.02.3 y posteriores
Europa (Estocolmo)	Versión 3.02.0 y posterior
Europa (Zúrich)	Versión 3.02.3 y posteriores
Israel (Tel Aviv)	Versiones 3.02.3 y posteriores, 3.03.1 y posteriores
México (central)	No disponible
Medio Oriente (Baréin)	Versión 3.02.0 y posterior
Medio Oriente (EAU)	Versión 3.02.3 y posteriores
América del Sur (São Paulo)	Versión 3.02.0 y posterior
AWS GovCloud (Este de EE. UU.)	Versión 3.02.2 y posteriores
AWS GovCloud (Oeste de EE. UU.)	Versión 3.02.2 y posteriores

Los límites superior e inferior de la ACU para la capacidad de Aurora Serverless v2 pueden variar según la versión del motor. Para obtener más información, consulte Capacidad de Aurora Serverless v2.

Aurora Serverless v2 con Aurora PostgreSQL

A continuación, se detallan las regiones y las versiones de motores que están disponibles para Aurora Serverless v2 con Aurora PostgreSQL.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Este de EE. UU.	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
(Norte de Virginia)					
Este de EE. UU. (Ohio)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Oeste de EE. UU. (Oregón)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
África (Ciudad del Cabo)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Hong Kong)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Hyderabad)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6 y posteriores	Versión 13.9 y posteriores
Asia-Pacífico (Yakarta)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Malasia)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6, 14.9 y posteriores	Versión 13.9, 13.12 y posteriores
Asia-Pacífico (Melbourne)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6 y posteriores	Versión 13.9 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia-Pacífico (Mumbai)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Seúl)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Singapur)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Sídney)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Asia-Pacífico (Tailandia)	Versión 17.4 y posteriores	Versión 16.4 y posteriores	Versión 15.8 y posteriores	Versión 14.13 y posteriores	No disponible
Asia-Pacífico (Tokio)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Canadá (centro)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Oeste de Canadá (Calgary)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.6, 14.8 y posteriores	Versión 13.9, 13.11 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
China (Pekín)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
China (Ningxia)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (Fráncfort)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (Irlanda)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (Londres)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (Milán)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (París)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (España)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6 y posteriores	Versión 13.9 y posteriores
Europa (Estocolmo)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Europa (Zúrich)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6 y posteriores	Versión 13.9 y posteriores
Israel (Tel Aviv)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6 y posteriores	Versión 13.9 y posteriores
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Medio Oriente (Baréin)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
Medio Oriente (EAU)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.6 y posteriores	Versión 13.9 y posteriores
América del Sur (São Paulo)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
AWS GovCloud (Este de EE. UU.)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores
AWS GovCloud (Oeste de EE. UU.)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.2 y posteriores	Versión 14.3 y posterior	Versión 13.6 y posteriores

Los límites superior e inferior de la ACU para la capacidad de Aurora Serverless v2 pueden variar según la versión del motor. Para obtener información, consulte Capacidad de Aurora Serverless v2.

Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1 que será el 31 de marzo de 2025. Recomendamos encarecidamente actualizar todos los clústeres de bases de datos de Aurora Serverless v1 a Aurora Serverless v2 antes de dicha fecha. Esta actualización puede implicar un cambio en el número de versión principal del motor de base de datos. Por lo tanto, es importante planificar, probar e implementar esta transición

antes de la fecha de fin de la vida útil. A partir del 8 de enero de 2025, los clientes ya no podrán crear nuevos clústeres o instancias de Aurora Serverless v1 con la Consola de administración de AWS o la CLI. Para obtener información sobre el proceso de migración, consulte Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2. Aurora Serverless v2 escala de forma más rápida y detallada. Aurora Serverless v2 también es más compatible con otras características de Aurora, como las instancias de base de datos de lector. Puede obtener más información sobre Aurora Serverless v2 en Uso de Aurora Serverless v2.

Aurora Serverless v1 es una característica de Auto Scaling en diferido diseñada para ser un enfoque rentable para ejecutar cargas de trabajo intermitentes o impredecibles en Amazon Aurora. Se inicia, se apaga y aumenta o reduce su capacidad de forma automática en función de las necesidades de las ap`licaciones utilizando una sola instancia de base de datos en cada clúster. Para obtener más información, consulte Uso de Amazon Aurora Serverless v1.

Temas

- Aurora Serverless v1 con Aurora MySQL
- Aurora Serverless v1 con Aurora PostgreSQL

Aurora Serverless v1 con Aurora MySQL

A continuación, se detallan las regiones y las versiones del motor que están disponibles para Aurora Serverless v1 con Aurora MySQL.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	No disponible	Versión 2.11.4
Este de EE. UU. (Ohio)	No disponible	Versión 2.11.4
Oeste de EE. UU. (Norte de California)	No disponible	Versión 2.11.4
Oeste de EE. UU. (Oregón)	No disponible	Versión 2.11.4

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
África (Ciudad del Cabo)	No disponible	No disponible
Asia-Pacífico (Hong Kong)	No disponible	No disponible
Asia-Pacífico (Hyderabad)	No disponible	No disponible
Asia-Pacífico (Yakarta)	No disponible	No disponible
Asia-Pacífico (Malasia)	No disponible	No disponible
Asia-Pacífico (Melbourne)	No disponible	No disponible
Asia-Pacífico (Mumbai)	No disponible	Versión 2.11.4
Asia-Pacífico (Osaka)	No disponible	No disponible
Asia-Pacífico (Seúl)	No disponible	Versión 2.11.4
Asia-Pacífico (Singapur)	No disponible	Versión 2.11.4
Asia-Pacífico (Sídney)	No disponible	Versión 2.11.4
Asia-Pacífico (Tailandia)	No disponible	No disponible
Asia-Pacífico (Tokio)	No disponible	Versión 2.11.4
Canadá (centro)	No disponible	Versión 2.11.4
Oeste de Canadá (Calgary)	No disponible	No disponible
China (Pekín)	No disponible	No disponible
China (Ningxia)	No disponible	Versión 2.11.4
Europa (Fráncfort)	No disponible	Versión 2.11.4
Europa (Irlanda)	No disponible	Versión 2.11.4
Europa (Londres)	No disponible	Versión 2.11.4

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Europa (Milán)	No disponible	No disponible
Europa (París)	No disponible	Versión 2.11.4
Europa (España)	No disponible	No disponible
Europa (Estocolmo)	No disponible	No disponible
Europa (Zúrich)	No disponible	No disponible
Israel (Tel Aviv)	No disponible	No disponible
Medio Oriente (Baréin)	No disponible	No disponible
Medio Oriente (EAU)	No disponible	No disponible
América del Sur (São Paulo)	No disponible	No disponible
AWS GovCloud (Este de EE. UU.)	No disponible	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible	No disponible

Aurora Serverless v1 con Aurora PostgreSQL

A continuación, se detallan las regiones y las versiones de motores que están disponibles para Aurora Serverless v1 con Aurora PostgreSQL.

Región	Aurora PostgreSQL 13
Este de EE. UU. (Norte de Virginia)	Versión 13.12
Este de EE. UU. (Ohio)	Versión 13.12
Oeste de EE. UU. (Norte de California)	Versión 13.12
Oeste de EE. UU. (Oregón)	Versión 13.12

Región	Aurora PostgreSQL 13
África (Ciudad del Cabo)	No disponible
Asia-Pacífico (Hong Kong)	No disponible
Asia-Pacífico (Hyderabad)	No disponible
Asia-Pacífico (Yakarta)	No disponible
Asia-Pacífico (Malasia)	No disponible
Asia-Pacífico (Melbourne)	No disponible
Asia-Pacífico (Mumbai)	Versión 13.12
Asia-Pacífico (Osaka)	No disponible
Asia-Pacífico (Seúl)	Versión 13.12
Asia-Pacífico (Singapur)	Versión 13.12
Asia-Pacífico (Sídney)	Versión 13.12
Asia-Pacífico (Tailandia)	No disponible
Asia-Pacífico (Tokio)	Versión 13.12
Canadá (centro)	Versión 13.12
Oeste de Canadá (Calgary)	No disponible
China (Pekín)	No disponible
China (Ningxia)	No disponible
Europa (Fráncfort)	Versión 13.12
Europa (Irlanda)	Versión 13.12
Europa (Londres)	Versión 13.12

Región	Aurora PostgreSQL 13
Europa (Milán)	No disponible
Europa (París)	Versión 13.12
Europa (España)	No disponible
Europa (Estocolmo)	No disponible
Europa (Zúrich)	No disponible
Israel (Tel Aviv)	No disponible
Medio Oriente (Baréin)	No disponible
Medio Oriente (EAU)	No disponible
América del Sur (São Paulo)	No disponible
AWS GovCloud (Este de EE. UU.)	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible

Regiones y motores de base de datos Aurora admitidos para API de datos de RDS

La API de datos de RDS (API de datos) proporciona una interfaz de servicios web para un clúster de base de datos de Amazon Aurora. En vez de administrar conexiones de base de datos desde aplicaciones cliente, puede ejecutar comandos SQL en un punto de enlace HTTPS. Para obtener más información, consulte Uso de la API de datos de Amazon RDS.

Temas

- API de datos con Aurora PostgreSQL sin servidor v2 y aprovisionada
- API de datos con Aurora MySQL sin servidor v2 y aprovisionada
- API de datos con Aurora PostgreSQL sin servidor v1
- API de datos con Aurora MySQL sin servidor v1

API de datos con Aurora PostgreSQL sin servidor v2 y aprovisionada

A continuación se detallan las regiones y las versiones de motores que están disponibles para la API de datos con Aurora PostgreSQL sin servidor v2 y clústeres de base de datos aprovisionados.

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Este de EE. UU. (Norte de Virginia)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Este de EE. UU. (Ohio)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Oeste de EE. UU. (Norte de California)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Oeste de EE. UU. (Oregón)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
África (Ciudad del Cabo)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Hong Kong)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Hyderabad)	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Yakarta)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia-Pacífico (Malasia)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Melbourne)	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Mumbai)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Osaka)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Seúl)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Singapur)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Sídney)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Taipéi)	No disponible	No disponible	No disponible	No disponible	No disponible
Asia-Pacífico (Tailandia)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Asia-Pacífico (Tokio)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Canadá (centro)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Oeste de Canadá (Calgary)	No disponible	No disponible	No disponible	No disponible	No disponible
China (Pekín)	No disponible	No disponible	No disponible	No disponible	No disponible
China (Ningxia)	No disponible	No disponible	No disponible	No disponible	No disponible
Europa (Fráncfort)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (Irlanda)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (Londres)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (Milán)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (París)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (España)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (Estocolmo)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
Europa (Zúrich)	No disponible	No disponible	No disponible	No disponible	No disponible

Región	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Israel (Tel Aviv)	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (Baréin)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
México (central)	No disponible	No disponible	No disponible	No disponible	No disponible
Medio Oriente (EAU)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
América del Sur (São Paulo)	Versión 17.4 y posteriores	Versión 16.1 y posteriores	Versión 15.3 y posteriores	Versión 14.8 y posterior	Versión 13.11 y posteriores
AWS GovCloud (Este de EE. UU.)	No disponible	No disponible	No disponible	No disponible	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible	No disponible	No disponible	No disponible	No disponible

API de datos con Aurora MySQL sin servidor v2 y aprovisionada

A continuación se detallan las regiones y las versiones de motores que están disponibles para la API de datos con Aurora MySQL sin servidor v2 y clústeres de base de datos aprovisionados.

Región	Aurora MySQL versión 3
Este de EE. UU. (Ohio)	Versión 3.07 y versiones posteriores
Este de EE. UU. (Norte de Virginia)	Versión 3.07 y versiones posteriores
Oeste de EE. UU. (Norte de California)	Versión 3.07 y versiones posteriores
Oeste de EE. UU. (Oregón)	Versión 3.07 y versiones posteriores
África (Ciudad del Cabo)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Hong Kong)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Hyderabad)	No disponible
Asia-Pacífico (Yakarta)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Malasia)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Melbourne)	No disponible
Asia-Pacífico (Mumbai)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Nueva Zelanda)	No disponible
Asia-Pacífico (Osaka)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Seúl)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Singapur)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Sídney)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Taipéi)	No disponible
Asia-Pacífico (Tailandia)	Versión 3.07 y versiones posteriores
Asia-Pacífico (Tokio)	Versión 3.07 y versiones posteriores
Canadá (centro)	Versión 3.07 y versiones posteriores

Región	Aurora MySQL versión 3
Oeste de Canadá (Calgary)	No disponible
China (Pekín)	No disponible
China (Ningxia)	No disponible
Europa (Fráncfort)	Versión 3.07 y versiones posteriores
Europa (Irlanda)	Versión 3.07 y versiones posteriores
Europa (Londres)	Versión 3.07 y versiones posteriores
Europa (Milán)	Versión 3.07 y versiones posteriores
Europa (París)	Versión 3.07 y versiones posteriores
Europa (España)	Versión 3.07 y versiones posteriores
Europa (Estocolmo)	Versión 3.07 y versiones posteriores
Europa (Zúrich)	No disponible
Israel (Tel Aviv)	No disponible
México (central)	No disponible
Medio Oriente (Baréin)	Versión 3.07 y versiones posteriores
Medio Oriente (EAU)	Versión 3.07 y versiones posteriores
América del Sur (São Paulo)	Versión 3.07 y versiones posteriores
AWS GovCloud (Este de EE. UU.)	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible

API de datos con Aurora PostgreSQL sin servidor v1

A continuación se detallan las regiones y las versiones de motores que están disponibles para la API de datos con Aurora PostgreSQL sin servidor v1.

Región	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Este de EE. UU. (Norte de Virginia)	Versión 13.9	Versión 11.18
Este de EE. UU. (Ohio)	Versión 13.9	Versión 11.18
Oeste de EE. UU. (Norte de California)	Versión 13.9	Versión 11.18
Oeste de EE. UU. (Oregón)	Versión 13.9	Versión 11.18
África (Ciudad del Cabo)	No disponible	No disponible
Asia-Pacífico (Hong Kong)	No disponible	No disponible
Asia-Pacífico (Hyderabad)	No disponible	No disponible
Asia-Pacífico (Yakarta)	No disponible	No disponible
Asia-Pacífico (Malasia)	No disponible	No disponible
Asia-Pacífico (Melbourne)	No disponible	No disponible
Asia-Pacífico (Mumbai)	Versión 13.9	Versión 11.18
Asia-Pacífico (Osaka)	No disponible	No disponible
Asia-Pacífico (Seúl)	Versión 13.9	Versión 11.18
Asia-Pacífico (Singapur)	Versión 13.9	Versión 11.18
Asia-Pacífico (Sídney)	Versión 13.9	Versión 11.18
Asia-Pacífico (Tailandia)	No disponible	No disponible

Región	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Asia-Pacífico (Tokio)	Versión 13.9	Versión 11.18
Canadá (centro)	Versión 13.9	Versión 11.18
China (Pekín)	No disponible	No disponible
China (Ningxia)	No disponible	No disponible
Europa (Fráncfort)	Versión 13.9	Versión 11.18
Europa (Irlanda)	Versión 13.9	Versión 11.18
Europa (Londres)	Versión 13.9	Versión 11.18
Europa (Milán)	No disponible	No disponible
Europa (París)	Versión 13.9	Versión 11.18
Europa (España)	Versión 13.9	Versión 11.18
Europa (Estocolmo)	No disponible	No disponible
Europa (Zúrich)	No disponible	No disponible
Israel (Tel Aviv)	No disponible	No disponible
Medio Oriente (Baréin)	No disponible	No disponible
Medio Oriente (EAU)	No disponible	No disponible
América del Sur (São Paulo)	No disponible	No disponible
AWS GovCloud (Este de EE. UU.)	No disponible	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible	No disponible

API de datos con Aurora MySQL sin servidor v1

A continuación se detallan las regiones y las versiones de motores que están disponibles para la API de datos con Aurora MySQL sin servidor v1.

Región	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Versión 2.11.3
Este de EE. UU. (Ohio)	Versión 2.11.3
Oeste de EE. UU. (Norte de California)	Versión 2.11.3
Oeste de EE. UU. (Oregón)	Versión 2.11.3
África (Ciudad del Cabo)	No disponible
Asia-Pacífico (Hong Kong)	No disponible
Asia-Pacífico (Hyderabad)	No disponible
Asia-Pacífico (Yakarta)	No disponible
Asia-Pacífico (Malasia)	No disponible
Asia-Pacífico (Melbourne)	No disponible
Asia-Pacífico (Mumbai)	Versión 2.11.3
Asia-Pacífico (Osaka)	No disponible
Asia-Pacífico (Seúl)	Versión 2.11.3
Asia-Pacífico (Singapur)	Versión 2.11.3
Asia-Pacífico (Sídney)	Versión 2.11.3
Asia-Pacífico (Tailandia)	No disponible
Asia-Pacífico (Tokio)	Versión 2.11.3
Canadá (centro)	Versión 2.11.3

Región	Aurora MySQL versión 2
Oeste de Canadá (Calgary)	No disponible
China (Pekín)	No disponible
China (Ningxia)	Versión 2.11.3
Europa (Fráncfort)	Versión 2.11.3
Europa (Irlanda)	Versión 2.11.3
Europa (Londres)	Versión 2.11.3
Europa (Milán)	No disponible
Europa (París)	Versión 2.11.3
Europa (España)	Versión 2.11.3
Europa (Estocolmo)	No disponible
Europa (Zúrich)	No disponible
Israel (Tel Aviv)	No disponible
Medio Oriente (Baréin)	No disponible
Medio Oriente (EAU)	No disponible
América del Sur (São Paulo)	No disponible
AWS GovCloud (Este de EE. UU.)	No disponible
AWS GovCloud (Oeste de EE. UU.)	No disponible

Regiones y motores de base de datos Aurora admitidos para aplicación de parches sin tiempo de inactividad (ZDP)

Realizar actualizaciones para clústeres de base de datos de Aurora implica la posibilidad de una interrupción cuando se cierra la base de datos y mientras se actualiza. De forma predeterminada, si comienza la actualización cuando la base de datos está ocupada, perderá todas las conexiones y transacciones que el clúster de bases de datos tiene en proceso. Si espera hasta que la base de datos esté inactiva para realizar la actualización, es posible que tenga que esperar mucho tiempo.

La característica de aplicación de parches sin tiempo de inactividad (ZDP) intenta, en la medida de lo posible, conservar las conexiones de cliente a través de una actualización de Aurora. Si la ZDP se completa correctamente, las sesiones de aplicación se conservan y el motor de base de datos se reinicia a la vez que la actualización está en curso. El reinicio del motor de base de datos puede provocar una disminución del rendimiento de unos segundos a aproximadamente un minuto.

Para obtener información detallada sobre las condiciones y las versiones del motor en las que la ZDP está disponible para las actualizaciones de Aurora MySQL, consulte Uso de parches sin tiempo de inactividad.

Para obtener información detallada sobre las condiciones y las versiones del motor en las que la ZDP está disponible para las actualizaciones de Aurora PostgreSQL, consulte Actualizaciones de versión secundarias y aplicación de revisiones sin tiempo de inactividad.

Aurora PostgreSQL Limitless Database でサポートされているリージョン

Amazon Aurora PostgreSQL Limitless Database は、1 秒あたり数百万件の書き込みトランザクション処理を可能にする自動水平スケーリング機能を提供し、単一のデータベース内で操作するようなシンプルさはそのままに、ペタバイト単位のデータを管理できます。Aurora PostgreSQL Limitless Database を使用すると、ワークロードをサポートするために複数の DB インスタンスにまたがってデータをスケールする複雑なソリューションを構築および維持する必要がないため、大規模なアプリケーションの構築に集中できます。

詳細については、「Uso de Base de datos ilimitada de Amazon Aurora PostgreSQL」を参照してください。

Aurora PostgreSQL Limitless Database は、アジアパシフィック (台北) を除くすべての Regiones de AWSで利用できます。

Regiones y motores de base de datos admitidos para características nativas del motor Aurora

Los motores de base de datos de Aurora también admiten funciones y funciones adicionales específicas para Aurora. Algunas funciones nativas del motor pueden disponer de una compatibilidad limitada o de privilegios restringidos para un motor, una versión o una región de bases de datos de Aurora en particular.

Temas

- Funciones nativas del motor para Aurora MySQL
- Funciones nativas del motor para Aurora PostgreSQL

Funciones nativas del motor para Aurora MySQL

A continuación se incluyen las funciones nativas del motor para Aurora MySQL.

- Auditoría avanzada
- Backtrack
- Consultas sobre inyección de errores
- Reenvío de escritura en el clúster
- Consulta paralela

Funciones nativas del motor para Aurora PostgreSQL

A continuación se incluyen las funciones nativas del motor para Aurora PostgreSQL.

- Babelfish
- Consultas sobre inyección de errores
- Administración de planes de consultas

Conexiones de puntos de conexión de Amazon Aurora

Amazon Aurora suele implicar un clúster de instancias de base de datos en lugar de una sola instancia. Una instancia de base de datos específica gestiona cada conexión. Al conectarse a

un clúster de Aurora, el nombre de anfitrión y el puerto especificados apuntan a un controlador intermedio denominado punto de enlace. Aurora utiliza el mecanismo de punto de enlace para abstraer estas conexiones. Por lo tanto, no tiene que codificar todos los nombres de host o escribir su propia lógica para el equilibrio y la redirección de conexiones cuando algunas instancias de base de datos no están disponibles.

En determinadas tareas de Aurora, las diversas instancias o grupos de instancias desempeñan diferentes roles. Por ejemplo, la instancia principal gestiona todas las instrucciones de lenguaje de definición de datos (DDL) y de lenguaje de manipulación de datos (DML). Hasta 15 réplicas de Aurora gestionan el tráfico de consultas de solo lectura.

Temas

- Tipos de puntos de enlace de Aurora
- Visualización de los puntos de enlace para un clúster de Aurora
- Cómo funcionan los puntos de enlace de Aurora con la alta disponibilidad
- Puntos de conexión de clúster para Amazon Aurora
- Puntos de conexión de lectura para Amazon Aurora
- Puntos de conexión de instancia para Amazon Aurora
- Puntos de conexión personalizados para Amazon Aurora

Tipos de puntos de enlace de Aurora

Al usar puntos de enlace puede asignar cada conexión a la instancia o grupo de instancias adecuados en función de su caso de uso. Por ejemplo, para realizar instrucciones DDL puede conectarse a la instancia que sea la instancia principal. Para realizar consultas, puede conectarse al punto de conexión del lector y Aurora lleva a cabo de forma automática el equilibrio de conexión entre todas las réplicas de Aurora. En el caso de clústeres con instancias de base de datos de diferentes capacidades o configuraciones, puede conectarse a puntos de enlace personalizados asociados a diversos subconjuntos de instancias de base de datos. En el caso de diagnóstico o ajuste, puede conectarse a un punto de enlace de instancia específico para examinar los detalles de una instancia de base de datos específica.

Un punto de enlace se representa como URL específica de Aurora que contiene una dirección de host y un puerto. Los siguientes tipos de puntos de enlace están disponibles en un clúster de bases de datos Aurora.

Punto de enlace de clúster

Conéctese a la instancia principal de su clúster para desarrollar o probar aplicaciones y para realizar transformaciones, como instrucciones `INSERT` y operaciones DDL, DML y ETL. Encuentre la ubicación del punto de conexión del clúster mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS, como se describe en Visualización de los puntos de enlace para un clúster de Aurora.

Para obtener más información sobre los puntos de conexión del clúster, consulte Puntos de conexión de clúster para Amazon Aurora.

Punto de conexión del lector

Realice consultas. Aurora equilibra automáticamente la conexión entre todas las réplicas de Aurora. Encuentre la ubicación del punto de conexión del lector mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS, como se describe en Visualización de los puntos de enlace para un clúster de Aurora.

Para obtener más información acerca de los puntos de conexión del lector, consulte Puntos de conexión de lectura para Amazon Aurora.

Punto de conexión de instancia

Examine los detalles de una instancia de base de datos específica para el diagnóstico o el ajuste. Solo puede encontrar la ubicación del punto de conexión de cada una de sus instancias en la Consola de administración de AWS, en la página de detalles de la instancia.

Para obtener más información acerca de los puntos de conexión de las instancias, consulte Puntos de conexión de instancia para Amazon Aurora.

Punto de enlace personalizado

Conéctese a diferentes subconjuntos de instancias de base de datos en el clúster de base de datos. Esto resulta útil cuando tiene distintas capacidades y configuraciones de instancias en el clúster de base de datos. Encuentre las ubicaciones del punto de conexión personalizado mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS, como se describe en Visualización de los puntos de enlace para un clúster de Aurora.

Para obtener más información acerca de los puntos de conexión personalizados, consulte Puntos de conexión personalizados para Amazon Aurora.

Punto de conexión del escritor de bases de datos global de Aurora

La base de datos global de Aurora tiene un tipo de punto de conexión especial que cumple el mismo propósito que el punto de conexión del clúster de un clúster de Aurora independiente. Gestiona las solicitudes de escritura y lectura. Cuando un clúster secundario se convierte en el nuevo clúster principal debido a una transición o una conmutación por error, Aurora cambia automáticamente este punto de conexión para que apunte al punto de conexión del clúster del nuevo clúster principal, en la otra Región de AWS. De esta forma, no tendrá que codificar la región de AWS en la cadena de conexión de su aplicación ni tendrá que cambiar la cadena de conexión cuando cambie el diseño de la base de datos global. Aurora crea este punto de conexión al configurar una base de datos global de Aurora, por ejemplo, al elegir Agregar región para un clúster de Aurora en la Consola de administración de AWS.

Para obtener información sobre cómo utilizar este tipo de punto de conexión con la base de datos global de Aurora, consulte Conexión a la base de datos global de Amazon Aurora.

Visualización de los puntos de enlace para un clúster de Aurora

Si bien solo puede encontrar la ubicación del punto de conexión de la instancia en la página de detalles de la instancia, en la Consola de administración de AWS, puede usar la consola, la AWS CLI o la API de Amazon RDS para buscar las ubicaciones del clúster, el lector y los puntos de conexión personalizados.

Console

En la Consola de administración de AWS, encontrará el punto de conexión del clúster, el punto de conexión del lector y cualquier punto de conexión personalizado en la página de detalles de instancia de su clúster. Verá el punto de enlace de instancia en la página de detalles de cada instancia. Al conectarse, añada el número de puerto asociado, tras el signo de dos puntos, al nombre de punto de conexión que aparece en la página de detalles.

AWS CLI

Con la AWS CLI, verá el escritor, el lector y cualquier punto de conexión personalizado en la salida del comando describe-db-clústers. Por ejemplo, el siguiente comando muestra los atributos de punto de conexión de todos los clústeres en la región de AWS actual.

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

Amazon RDS API

Con la API de Amazon RDS, recuperará los puntos de conexión llamando a la operación DescribeDBclústerEndpoints.

Cómo funcionan los puntos de enlace de Aurora con la alta disponibilidad

Para clústeres en los que la alta disponibilidad es importante, utilice el punto de conexión de clúster para conexiones de lectura y escritura, o con fines generales, así como el punto de conexión del lector para conexiones de solo lectura. Los puntos de enlace del escritor y del lector administran la conmutación por error de instancias de base de datos mejor que los puntos de enlace de instancia. A diferencia de los puntos de enlace de instancia, los puntos de enlace del escritor y del lector cambian automáticamente a qué instancia de base de datos se conectan si una instancia de base de datos del clúster deja de estar disponible. Para obtener más información acerca de los puntos de conexión del lector y el clúster, consulte Puntos de conexión de clúster para Amazon Aurora y Puntos de conexión de lectura para Amazon Aurora.

Si se produce un error en la instancia de base de datos principal de un clúster de bases de datos, Aurora conmuta por error automáticamente a una nueva instancia de base de datos principal. Lo hace promoviendo una réplica de Aurora existente a una nueva instancia de base de datos principal o creando una instancia de base de datos principal. Si se produce una conmutación por error, puede usar el punto de conexión del clúster para volver a conectarse a la instancia de base de datos recién promocionada o creada, o puede usar el punto de conexión del lector para volver a conectarse a una de las réplicas de Aurora en el clúster de base de datos. Durante una conmutación por error, el punto de enlace del lector podría dirigir las conexiones a la nueva instancia de base de datos principal de un clúster de bases de datos durante un breve periodo tras convertirse una réplica de Aurora en la nueva instancia de base de datos principal.

Si diseña su propia lógica de aplicación para administrar conexiones a puntos de enlace de instancia, puede, manualmente o mediante programación, encontrar el conjunto resultante de instancias de base de datos disponibles en el clúster de bases de datos. Utilice el comando describe-db-clusters de la AWS CLI o la operación DescribeDBClusters de la API de RDS para encontrar el clúster de base de datos y los puntos de conexión de lector, las instancias de base de datos, ya sean lectoras, y sus niveles de promoción. Luego puede confirmar sus clases de instancia tras la conmutación por error y conectarse a un punto de enlace de instancia adecuado.

Para obtener más información acerca de las conmutaciones por error, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

Para obtener más información sobre la alta disponibilidad en Amazon Aurora, consulte Alta disponibilidad para Amazon Aurora.

Puntos de conexión de clúster para Amazon Aurora

El punto de enlace de clúster o (punto de enlace del escritor) para un clúster de bases de datos Aurora se conecta a la instancia de base de datos principal actual de ese clúster de bases de datos. Este punto de enlace es el único que puede realizar operaciones de escritura como instrucciones DDL. Por este motivo, el punto de enlace del clúster es el punto al que se conecta la primera vez que configura un clúster o bien si su clúster solo contiene una única instancia de base de datos.

Cada clúster de bases de datos de Aurora tiene un punto de enlace de clúster y una instancia de base de datos principal.

Utilice el punto de enlace del clúster para todas las operaciones de escritura en el clúster de la base de datos, incluidos inserciones, actualizaciones, eliminaciones y cambios de DDL. También puede usar el punto de enlace del clúster para operaciones de lectura, como por ejemplo consultas.

El punto de enlace del clúster proporciona soporte de conmutación por error para conexiones de lectura/escritura al clúster de bases de datos. Si se produce un error en la instancia de base de datos principal actual de un clúster de bases de datos, Aurora conmuta por error automáticamente a una nueva instancia de base de datos principal. Durante una conmutación por error, el clúster de bases de datos continúa atendiendo solicitudes de conexión al punto de enlace del clúster de la nueva instancia de base de datos principal, con una interrupción del servicio mínima.

En el siguiente ejemplo se ilustra un punto de enlace del clúster de un clúster de bases de datos Aurora MySQL.

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

Cada clúster de Aurora tiene un solo punto de conexión de clúster integrado, cuyo nombre y atributos están administrados por Aurora. No puede crear, eliminar o modificar este tipo de punto de conexión.

Use el punto de enlace del clúster al administrar su clúster, realizar la extracción, transformar, cargar operaciones (ETL) o desarrollar y probar aplicaciones. El punto de enlace de clúster se conecta a la instancia principal del clúster. La instancia principal es la única instancia de base de datos donde puede crear tablas e índices, ejecutar instrucciones `INSERT` y realizar otras operaciones de DDL y DML.

La dirección IP física a la que apunta el punto de enlace del clúster cambia cuando el mecanismo de conmutación por error promueve una nueva instancia de base de datos para que sea la instancia principal de lectura/escritura del clúster. Si usa cualquier forma de grupos de conexiones u otros multiplexados, prepárese para vaciar o reducir el tiempo de vida de cualquier información de DNS almacenada en caché. De esta forma se garantiza que no intente establecer una conexión de lectura/escritura en una instancia de base de datos que deje de estar disponible o sea ahora de solo lectura tras una conmutación por error.

Puntos de conexión de lectura para Amazon Aurora

El punto de conexión del lector para un clúster de bases de datos Aurora proporciona soporte de equilibrio de conexión para conexiones de solo lectura al clúster de bases de datos. Utilice el punto de enlace del lector para operaciones de lectura, como por ejemplo consultas. Al procesar esas instrucciones en las réplicas de Aurora de solo lectura, este punto de enlace reduce la sobrecarga de la instancia principal. También ayuda al clúster a escalar la capacidad para gestionar consultas `SELECT` simultáneas proporcionalmente al número de réplicas de Aurora en el clúster. Cada clúster de bases de datos Aurora tiene un punto de enlace del lector.

Si el clúster contiene una o más réplicas de Aurora, el punto de conexión del lector equilibra cada solicitud de conexión entre las réplicas de Aurora. En ese caso, solo se pueden ejecutar instrucciones de solo lectura como `SELECT` en esa sesión. Si el clúster solo contiene una instancia principal y no hay réplicas de Aurora, el punto de enlace del lector se conecta a la instancia principal. En ese caso, se pueden ejecutar operaciones de escritura a través del punto de enlace.

En el siguiente ejemplo se ilustra un punto de enlace del lector de un clúster de bases de datos Aurora MySQL.

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

Use el punto de enlace del lector para conexiones de solo lectura para su clúster de Aurora. Este punto de conexión usa un mecanismo de equilibrio de conexión para ayudar a su clúster a gestionar una carga de trabajo con uso intensivo de consultas. El punto de conexión del lector es el punto de conexión que proporciona a las aplicaciones que realizan informes u otras operaciones de solo lectura en el clúster.

El punto de conexión del lector equilibra las conexiones a las réplicas de Aurora disponibles en un clúster de bases de datos de Aurora. No equilibra consultas individuales. Si desea equilibrar cada consulta y distribuir la carga de trabajo de lectura de un clúster de bases de datos, abra una nueva conexión al punto de conexión del lector para cada consulta.

Cada clúster de Aurora tiene un solo punto de enlace del lector integrado cuyo nombre y otros atributos se administran mediante Aurora. No puede crear, eliminar o modificar este tipo de punto de enlace.

Si el clúster contiene solo un objetivo principal (instancia o grupo de partición de base de datos) y no hay réplicas de Aurora, el punto de conexión del lector se conecta a la instancia principal. En ese caso, podrá realizar operaciones de escritura a través de este punto de enlace.

> **Tip**
>
> A través del proxy de RDS, se pueden crear puntos de enlace adicionales de sólo lectura para un clúster de Aurora. Estos puntos de conexión realizan el mismo tipo de equilibrado de conexión que el punto de conexión del lector Aurora. Las aplicaciones pueden volver a conectarse más rápidamente a los puntos de enlace del proxy que el punto de enlace del lector Aurora si las instancias no están disponibles. Los puntos de enlace del proxy también pueden aprovechar otras características del proxy, como la multiplexación. Para obtener más información, consulte Usar puntos de enlace del lector con los clústeres de Aurora.

Puntos de conexión de instancia para Amazon Aurora

Un punto de enlace de una instancia se conecta a una instancia de base de datos específica de un clúster de Aurora. Cada instancia de base de datos de un clúster de bases de datos tiene su propio punto de enlace de instancia único. Así que hay un punto de enlace de instancia para la actual instancia de base de datos principal del clúster de bases de datos y un punto de enlace de instancia para cada una de las réplicas de Aurora en el clúster de la base de datos.

El punto de enlace de la instancia proporciona un control directo sobre las conexiones al clúster de bases de datos, en los casos en los que el uso del punto de enlace del clúster o del lector puede no ser adecuado. Por ejemplo, su aplicación cliente podría necesitar un equilibrado de conexión más detallado en función del tipo de carga de trabajo. En este caso, puede configurar varios clientes para que se conecten a distintas réplicas de Aurora de un clúster de bases de datos con el fin de distribuir las cargas de trabajo de lectura. Si quiere ver un ejemplo donde se usen puntos de enlace de la instancia para mejorar la velocidad de conexión tras una conmutación por error de Aurora PostgreSQL, consulte Conmutación por error rápida con Amazon Aurora PostgreSQL. Si quiere ver un ejemplo donde se usen puntos de conexión de la instancia para mejorar la velocidad de conexión tras una conmutación por error de Aurora MySQL, consulte MariaDB Connector/J failover support – case Amazon Aurora.

En el siguiente ejemplo se ilustra un punto de enlace de la instancia de una instancia de base de datos de un clúster de bases de datos Aurora MySQL.

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

Cada instancia de base de datos en un clúster de Aurora tiene su propio punto de enlace de instancia integrado, cuyo nombre y otros atributos administra Aurora. No puede crear, eliminar o modificar este tipo de punto de enlace. Es posible que esté familiarizado con los puntos de enlace de instancia si usa Amazon RDS. Sin embargo, con Aurora normalmente se utilizan los puntos de enlace del escritor y del lector con más frecuencia que los puntos de enlace de instancia.

En las operaciones de Aurora diarias, la forma principal de uso de los puntos de enlace de instancia consiste en diagnosticar los problemas de rendimiento o capacidad que afectan a una instancia específica de un clúster de Aurora. Mientras se conecta a una instancia específica, puede examinar sus variables de estado, métricas, etc. Hacer esto puede ayudarle a determinar qué sucede con esa instancia que es distinto de lo que ocurre con otras instancias del clúster.

En los casos de uso avanzados, podría configurar algunas instancias de base de datos de manera distinta a otras. En este caso, use el punto de enlace de instancia para conectarse directamente a una instancia que sea más pequeña, más grande o que, de otro modo, tenga características distintas del resto. Asimismo, configure la prioridad de conmutación por error para que esta instancia de base de datos especial sea la última opción para hacerse cargo como instancia principal. Recomendamos que use puntos de enlace personalizados en lugar del punto de enlace de instancia en estos casos. Al hacerlo se simplifica la administración de la conexión y la alta disponibilidad a medida que añade más instancias de base de datos a su clúster.

Puntos de conexión personalizados para Amazon Aurora

Un punto de enlace personalizado de un clúster de Aurora representa un conjunto de instancias de base de datos que ha elegido. Al conectarse al punto de conexión, Aurora realiza el equilibrio de conexión y elige una de las instancias del grupo para gestionar la conexión. Defina las instancias a las que hace referencia este punto de enlace y decida el objetivo de este.

Un clúster de bases de datos Aurora no tiene puntos de enlace personalizados hasta que crea uno. Puede crear hasta cinco puntos de conexión personalizados para cada clúster de Aurora aprovisionado o de Aurora Serverless v2. No puede usar puntos de enlace personalizados para clústeres de Aurora Serverless v1.

El punto de conexión personalizado proporciona conexiones de base de datos con equilibrio en función de otros criterios, aparte de la capacidad de solo lectura o de lectura/escritura de las instancias de base de datos. Por ejemplo, podría definir un punto de enlace personalizado para conectarse a instancias que usan una clase de instancia de AWS particular o un grupo de parámetros de base de datos particular. A continuación, podría informar a grupos de usuarios particulares acerca de este punto de enlace personalizado. Por ejemplo, podría dirigir a los usuarios internos a instancias de baja capacidad para la generación de informes o de consultas ad hoc (de una vez). También podría dirigir el tráfico de producción a instancias de alta capacidad.

Dado que la conexión puede ir a cualquier instancia de base de datos que se asocie al punto de enlace personalizado, recomendamos que se asegure de que todas las instancias de base de datos de ese grupo comparten alguna característica similar. De esta forma se garantiza que el rendimiento, la capacidad de memoria, etc. sean coherentes para todo aquel que se conecte a ese punto de enlace.

Esta característica está destinada a usuarios avanzados con tipos de cargas de trabajo especializados donde no resulta práctico que todas las réplicas de Aurora del clúster sean idénticas. Con los puntos de enlace personalizados, puede predecir la capacidad de la instancia de base de datos usada para cada conexión. Al usar puntos de enlace personalizados, normalmente no usa el punto de enlace del lector de ese clúster.

En el siguiente ejemplo se ilustra un punto de enlace personalizado de una instancia de base de datos de un clúster de bases de datos de Aurora MySQL.

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

Puede usar puntos de enlace personalizados para simplificar la administración de la conexión si su clúster contiene instancias de base de datos con diferentes capacidades y ajustes de configuración.

Anteriormente, podría haber usado el mecanismo CNAMES para configurar alias del servicio de nombres de dominio (DNS) desde su propio dominio para lograr resultados similares. Al usar puntos de enlace personalizados, puede evitar actualizar registros CNAME al aumentar o disminuir su clúster. Los puntos de enlace personalizados también significan que puede usar conexiones Transport Layer Security/de capa de conexión segura (TLS/SSL).

En lugar de usar una instancia de base de datos para cada propósito especializado y conectarse a su punto de enlace de instancia, puede tener varios grupos de instancias de base de datos especializadas. En este caso, cada grupo tiene su propio punto de enlace personalizado. De esta forma, Aurora puede realizar el equilibrio de conexión entre todas las instancias dedicadas

a tareas como la creación de informes o la gestión de consultas de producción o internas. Los puntos de conexión personalizados distribuyen las conexiones entre las instancias de forma pasiva y utilizan DNS para devolver la dirección IP de una de las instancias de forma aleatoria. Si una de las instancias de base de datos dentro de un grupo deja de estar disponible, Aurora dirige conexiones de punto de enlace personalizado posteriores a una de las otras instancias de base de datos asociadas al mismo punto de enlace.

Temas

- Observaciones sobre los puntos de conexión personalizados en Amazon Aurora
- Creación de un punto de enlace personalizado
- Visualización de puntos de enlace personalizados
- Edición de un punto de enlace personalizado
- Eliminación de un punto de enlace personalizado
- Ejemplos de AWS CLI sobre puntos de conexión personalizados para Amazon Aurora

Observaciones sobre los puntos de conexión personalizados en Amazon Aurora

Utilice las siguientes secciones para administrar, especificar propiedades y usar las reglas de afiliación para puntos de conexión personalizados.

Temas

- Administración de puntos de enlace personalizados
- Especificación de propiedades para puntos de enlace personalizados
- Reglas de afiliación para puntos de enlace personalizados

Administración de puntos de enlace personalizados

Dado que los clústeres de Aurora recién creados no tienen ningún punto de enlace personalizado, debe crear y administrar estos objetos usted mismo. Para ello, utilice la Consola de administración de AWS, AWS CLI o la API de Amazon RDS.

> **Note**
>
> También debe crear y administrar puntos de enlace personalizados para clústeres de Aurora restaurados a partir de instantáneas. Los puntos de enlace personalizados no se incluyen

en la instantánea. Vuélvalos a crear tras la restauración y elija nuevos nombres de punto de enlace si el clúster restaurado está en la misma región que el original.

Para trabajar con puntos de enlace personalizados desde la Consola de administración de AWS, vaya a la página de detalles para su clúster de Aurora y utilice los controles de la sección de puntos de enlace personalizados.

Para trabajar con puntos de enlace personalizados desde la AWS CLI, puede usar estas operaciones:

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

Para trabajar con puntos de enlace personalizados a través de la API de Amazon RDS, puede usar las siguientes funciones:

- CreateDBclústerEndpoint
- DescribeDBclústerEndpoints
- ModifyDBclústerEndpoint
- DeleteDBclústerEndpoint

Especificación de propiedades para puntos de enlace personalizados

La longitud máxima de un nombre de punto de enlace personalizado es 63 caracteres. El formato del nombre es este:

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

No puede volver a usar el mismo nombre de punto de conexión personalizado para más de un clúster en la misma Región de AWS. El identificador DNS personalizado es un identificador único asociado a la Cuenta de AWS en una Región de AWS particular.

Cada punto de enlace personalizado tiene un tipo asociado que determina qué instancias de base de datos cumplen los requisitos para asociarse a ese punto de enlace. En la actualidad, el tipo

puede ser READER o ANY. Las siguientes consideraciones se aplican a los tipos de punto de enlace personalizado:

- No puede seleccionar el tipo de punto de conexión personalizado en la Consola de administración de AWS. Todos los puntos de enlace personalizados que crea a través de la Consola de administración de AWS tienen un tipo de ANY.

  Puede configurar y modificar el tipo de punto de conexión personalizado mediante la AWS CLI o la API de Amazon RDS.
- Solo las instancias de base de datos del lector pueden formar parte de un punto de conexión personalizado READER.
- Tanto las instancias de base de datos del lector como del escritor pueden formar parte de un punto de conexión personalizado ANY. Aurora dirige las conexiones a los puntos de enlace de clúster con ANY tipo a cualquier instancia de base de datos asociada con la misma probabilidad. El tipo ANY se aplica a los clústeres que usan cualquier topología de replicación.
- Si intenta crear un punto de enlace personalizado con un tipo que no es adecuado en función de la configuración de replicación para un clúster, Aurora devuelve un error.

Reglas de afiliación para puntos de enlace personalizados

Al añadir una instancia de base de datos a un punto de enlace personalizado o quitarla de un punto de enlace personalizado, cualquier conexión existente a esa instancia de base de datos permanece activa.

Puede definir una lista de instancias de base de datos que se incluirán en un punto de enlace personalizado o se excluirán de este. Nos referimos a estas listas como listas estáticas y de exclusión, respectivamente. Puede usar el mecanismo de inclusión/exclusión para seguir subdividiendo los grupos de instancias de base de datos y asegurarse de que el conjunto de puntos de enlace personalizados cubre todas las instancias de base de datos en el clúster. Cada punto de enlace personalizado puede contener solo uno de estos tipos de lista.

En:Consola de administración de AWS

- La opción está representada por la casilla Attach future instances added to this clúster (Asociar futuras instancias añadidas a este clúster). Al no seleccionar la casilla de verificación, el punto de conexión personalizado usa una lista estática que contiene solo las instancias de base de datos especificadas en la página. Al seleccionar la casilla de verificación, el punto de enlace personalizado usa una lista de exclusión. En este caso, el punto de conexión personalizado

representa todas las instancias de base de datos del clúster (incluidas las añadidas en el futuro), excepto las que no se hayan seleccionado en la página.

- La consola no permite especificar el tipo de punto de conexión. Cualquier punto de conexión personalizado creado con la consola es de tipo `ANY`.

  Por lo tanto, Aurora no cambia la pertenencia al punto de conexión personalizado cuando las instancias de base de datos cambian de rol entre escritor y lector debido a una conmutación por error o a una promoción.

En la AWS CLI y la API de Amazon RDS:

- Puede especificar el tipo de punto de conexión. Por lo tanto, cuando el tipo de punto de conexión se establece en `READER`, la pertenencia al punto de conexión se ajusta automáticamente durante las conmutaciones por error y las promociones.

  Por ejemplo, un punto de conexión personalizado con el tipo `READER` incluye una réplica de Aurora que luego se promociona para ser una instancia de base de datos de escritura. La nueva instancia de escritor ya no forma parte del punto de conexión personalizado.
- Puede añadir miembros individuales a las listas y eliminarlos de ellas después de que cambien sus roles. Utilice el comando de la AWS CLI modify-db-clúster-endpoint o la operación de API ModifyDBclústerEndpoint.

Puede asociar una instancia de base de datos a más de un punto de enlace personalizado. Por ejemplo, supongamos que añade una nueva instancia de base de datos a un clúster, o que Aurora añade una instancia de base de datos automáticamente a través del mecanismo de Auto Scaling. En estos casos, la instancia de base de datos se añade a todos los puntos de enlace personalizados para los que cumple los requisitos. A qué puntos de conexión se agrega la instancia de base de datos se basa en el tipo de punto de conexión personalizado de `READER` o `ANY`, más cualquier lista estática o de exclusión definida para cada punto de conexión. Por ejemplo, si el punto de enlace incluye una lista estática de instancias de base de datos, las réplicas de Aurora recién añadidas no se añaden a ese punto de enlace. Por el contrario, si el punto de enlace tiene una lista de exclusión, las réplicas de Aurora recién añadidas se añaden al punto de enlace, si no se menciona en la lista de exclusión y sus roles coinciden con el tipo del punto de enlace personalizado.

Si una réplica de Aurora deja de estar disponible, permanece asociada a cualquier punto de enlace personalizado. Por ejemplo, sigue formando parte del punto de enlace personalizado si está en mal

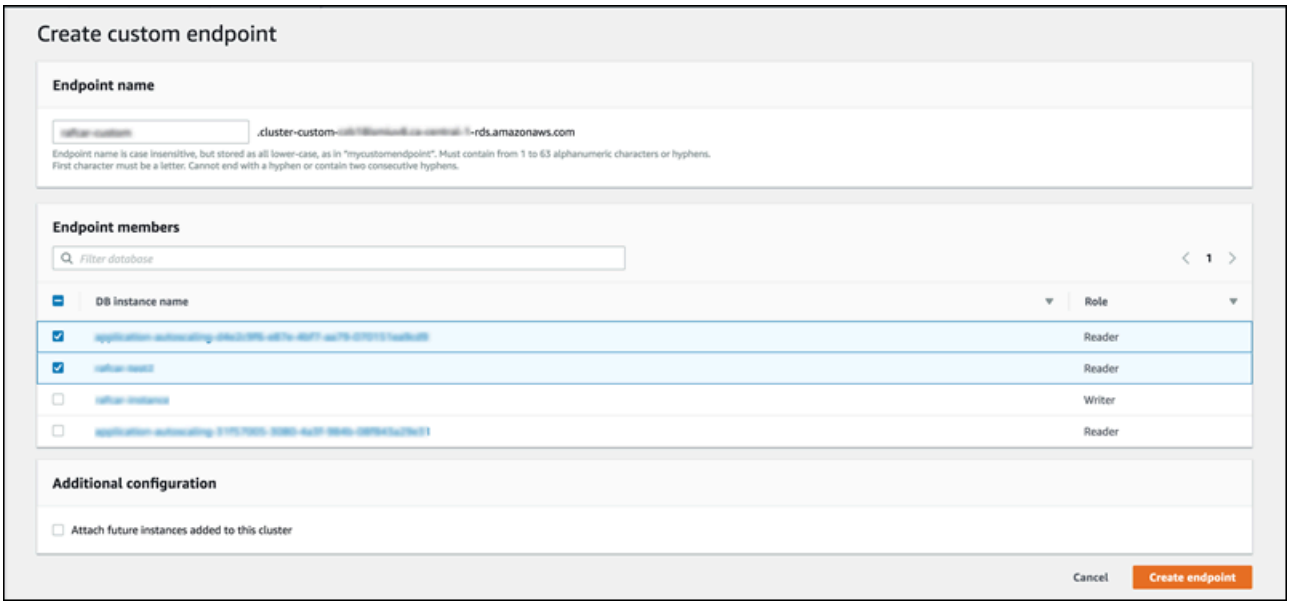
estado, se detiene, se reinicia, etc. Sin embargo, no podrá conectarse a ella a través de esos puntos de enlace hasta que vuelva a estar disponible.

Creación de un punto de enlace personalizado

Para crear un punto de conexión personalizado, utilice la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

Consola

Para crear un punto de enlace personalizado con la Consola de administración de AWS, vaya a la página de detalles del clúster y elija la acción `Create custom endpoint` en la sección Endpoints (Puntos de enlace). Elija un nombre para el punto de enlace personalizado, único para su ID de usuario y región. Para elegir una lista de instancias de base de datos que se conserve igual incluso a medida que se amplía el clúster, deje sin seleccionar la casilla de verificación Attach future instances added to this clúster (Asociar futuras instancias añadidas a este clúster). Al seleccionar esa casilla de verificación, el punto de enlace personalizado añade de forma dinámica cualquier nueva instancia a medida que se añaden al clúster.



No puede seleccionar el tipo de punto de conexión personalizado en la Consola de administración de AWS. Todos los puntos de conexión personalizados que cree a través de la Consola de administración de AWS tienen el tipo `ANY`.

AWS CLI

Para crear un punto de enlace personalizado con la AWS CLI, ejecute el comando create-db-clúster-endpoint.

El siguiente comando crea un punto de enlace personalizado asociado a un clúster específico. En un primer momento, el punto de enlace se asocia a todas las instancias de réplica de Aurora del clúster. Un comando posterior lo asocia a un conjunto específico de instancias de base de datos del clúster.

Para Linux, macOS o Unix:

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

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

Para Windows:

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

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

API de RDS

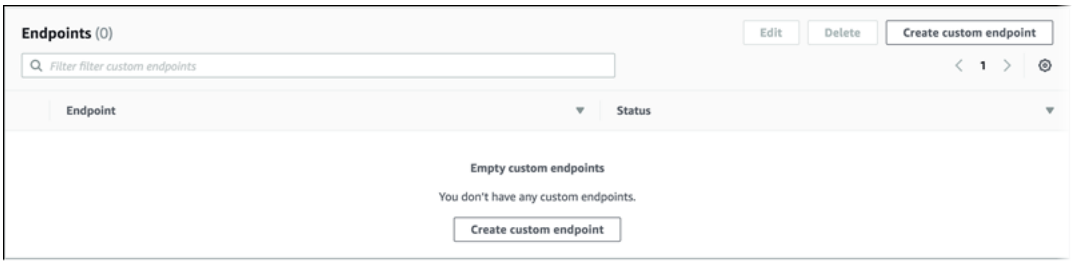
Para crear un punto de enlace personalizado con la API de RDS, ejecute la operación CreateDBclústerEndpoint.

Visualización de puntos de enlace personalizados

Consola

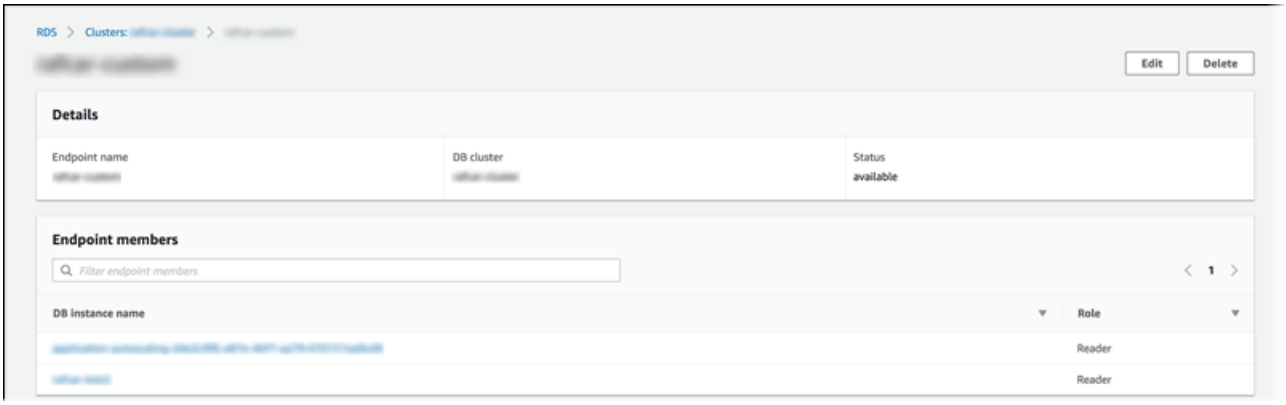
Para ver puntos de enlace personalizados con la Consola de administración de AWS, vaya a la página de detalles del clúster y busque en la sección Endpoints (Puntos de enlace). Esta sección solo contiene información sobre puntos de enlace personalizados. Los detalles de los puntos de enlace integrados aparecen en la sección Details (Detalles) principal. Para ver los detalles de un punto de enlace personalizado específico, seleccione su nombre para abrir la página de detalles para ese punto de enlace.

En la siguiente captura de pantalla se muestra cómo la lista de puntos de enlace personalizados para un clúster de Aurora está inicialmente vacía.



Tras crear algunos puntos de enlace personalizados para ese clúster, se muestran en la sección Endpoints (Puntos de enlace).

Al hacer clic en la página de detalles se muestran las instancias de base de datos a las que se asocia el punto de enlace actualmente.



Para ver el detalle adicional de si las nuevas instancias de base de datos añadidas al clúster se añaden también automáticamente al punto de conexión, abra el cuadro de diálogo Edit (Editar) del punto de enlace.

AWS CLI

Para ver puntos de enlace personalizados con la AWS CLI, ejecute el comando describe-db-clúster-endpoints.

El siguiente comando muestra los puntos de enlace personalizados asociados a un clúster especificado en una región especificada. La salida incluye los puntos de enlace integrados y cualquier punto de enlace personalizado.

Para Linux, macOS o Unix:

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

Para Windows:

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

A continuación, se muestran algunos resultados de ejemplo de un comando `describe-db-cluster-endpoints`. El `EndpointType` de `WRITER` o `READER` denota los puntos de enlace de lectura/escritura integrados y de solo lectura para el clúster. El `EndpointType` de `CUSTOM` denota los puntos de enlace que crea y elija las instancias de base de datos asociadas. Uno de los puntos de enlace tiene un campo `StaticMembers` no vacío, lo que denota que se asocia a un conjunto preciso de instancias de base de datos. El otro punto de enlace tiene un campo `ExcludedMembers` no vacío, lo que denota que el punto de enlace se asocia a todas las instancias de base de datos que no sean las que figuran en `ExcludedMembers`. Este segundo tipo de punto de enlace personalizado aumenta para incorporar nuevas instancias a medida que se añaden al clúster.

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

```
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "powers-of-2",
   "ExcludedMembers": [],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "powers-of-2.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-08",
     "custom-endpoint-demo-01",
     "custom-endpoint-demo-02"
   ],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:powers-of-2"
  },
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "eight-and-higher",
   "ExcludedMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-02",
     "custom-endpoint-demo-07",
     "custom-endpoint-demo-05",
     "custom-endpoint-demo-03",
     "custom-endpoint-demo-06",
     "custom-endpoint-demo-01"
   ],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "eight-and-higher.cluster-custom-123456789012.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHYQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:eight-and-higher"
  }
 ]
```

```
}
```

API de RDS

Para ver puntos de enlace personalizados con la API de RDS, ejecute la operación DescribeDBclústerEndpoints.html.

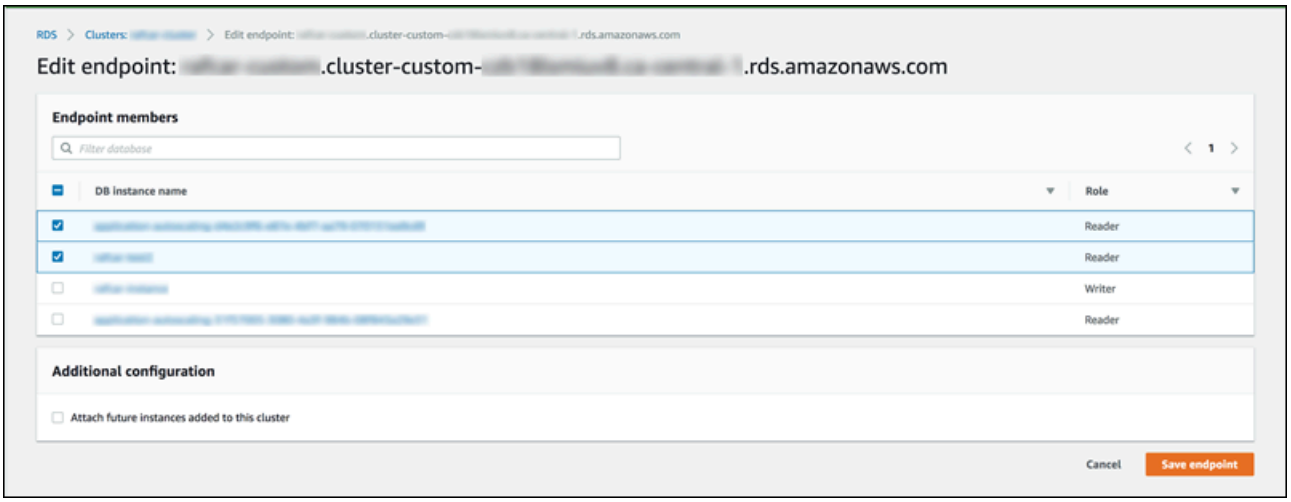
Edición de un punto de enlace personalizado

Puede editar las propiedades de un punto de enlace personalizado para cambiar las instancias de base de datos asociadas al punto de enlace. También puede cambiar un punto de enlace entre una lista estática y una lista de exclusión. Si necesita más detalles acerca de estas propiedades de punto de enlace, consulte Reglas de afiliación para puntos de enlace personalizados.

Podrá seguir conectándose a un punto de conexión personalizado mientras hay en curso cambios de una acción de edición.

Consola

Para editar un punto de enlace personalizado con la Consola de administración de AWS, puede seleccionar el punto de enlace en la página de detalles del clúster o abrir la página de detalles para el punto de enlace y elegir la acción Edit (Editar).



AWS CLI

Para editar un punto de enlace personalizado con la AWS CLI, ejecute el comando modify-db-clúster-endpoint.

Los siguientes comandos cambian el conjunto de instancias de base de datos que se aplican a un punto de enlace personalizado y, de forma opcional, se alterna entre el comportamiento de una lista

estática o de exclusión. Los parámetros `--static-members` y `--excluded-members` toman una lista separada por espacios de identificadores de instancias de bases de datos.

Para Linux, macOS o Unix:

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

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

Para Windows:

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

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

API de RDS

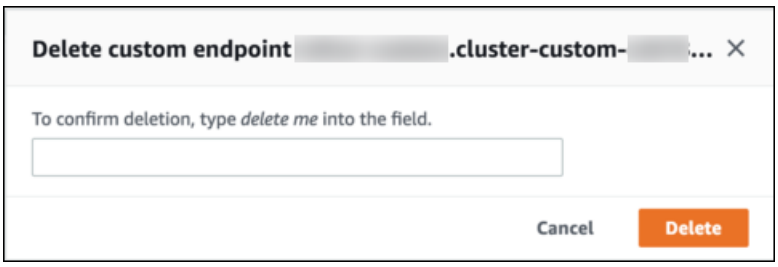
Para editar un punto de enlace personalizado con la API de RDS, ejecute la operación ModifyDBclústerEndpoint.html.

Eliminación de un punto de enlace personalizado

Para eliminar un punto de conexión personalizado, utilice la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

Consola

Para eliminar un punto de enlace personalizado con la Consola de administración de AWS, vaya a la página de detalles del clúster, seleccione el punto de enlace personalizado adecuado y seleccione la acción Delete (Eliminar).



AWS CLI

Para eliminar un punto de enlace personalizado con la AWS CLI, ejecute el comando delete-db-clúster-endpoint.

El siguiente comando elimina un punto de enlace personalizado. No tiene que especificar el clúster asociado, pero debe especificar la región.

Para Linux, macOS o Unix:

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

Para Windows:

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

API de RDS

Para eliminar un punto de enlace personalizado con la API de RDS, ejecute la operación DeleteDBclústerEndpoint.

Ejemplos de AWS CLI sobre puntos de conexión personalizados para Amazon Aurora

En el siguiente tutorial, se usan ejemplos AWS CLI con sintaxis de shell Unix para mostrar cómo se puede definir un clúster con varias instancias de base de datos “pequeñas” y algunas “grandes”; también se muestra cómo crear puntos de conexión personalizados para conectarse a cada conjunto de instancias de base de datos. Para ejecutar comandos similares en su propio sistema, debe estar suficientemente familiarizado con los aspectos básicos de trabajar con los clústeres de Aurora y el

uso de AWS CLI a fin de proporcionar sus propios valores para parámetros, como región, grupo de subredes y grupo de seguridad de VPC.

En este ejemplo se muestran los pasos de configuración iniciales: creación de un clúster de Aurora y adición de instancias de base de datos a este. Este es un clúster heterogéneo, lo que significa que no todas las instancias de base de datos tienen la misma capacidad. La mayoría de las instancias usan la clase AWS de instancia `db.r4.4xlarge`, pero las dos últimas instancias de base de datos usan `db.r4.16xlarge`. Cada uno de estos comandos `create-db-instance` de ejemplo muestra su resultado en la pantalla y ahorra una copia del JSON en un archivo para su posterior inspección.

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

for i in 01 02 03 04 05 06 07 08
do
  aws rds create-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
     --engine aurora --db-cluster-identifier custom-endpoint-demo --db-instance-class
 db.r4.4xlarge \
     --region $REGION \
     | tee custom-endpoint-demo-${i}.json
done

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

Las instancias más grandes están reservadas para tipos especializados de consultas de informes. Para que sea poco probable que se conviertan en la instancia principal, en el siguiente ejemplo la prioridad de su nivel de promoción pasa a ser la más baja. En este ejemplo se especifica la opción `--manage-master-user-password` para generar la contraseña del usuario maestro y administrarla en Secrets Manager. Para obtener más información, consulte Administración de

contraseñas con Amazon Aurora y AWS Secrets Manager. También puede utilizar la opción `--master-password` para especificar y administrar la contraseña usted mismo.

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

Supongamos que desea usar las dos instancias «más grandes» solo para las consultas que más recursos consumen. Para ello, primero puede crear un punto de enlace de solo lectura personalizado. A continuación, puede agregar una lista estática de miembros para que el punto de enlace se conecte solo a esas instancias de base de datos. El nivel de promoción de esas instancias ya es el más bajo, lo que reduce la probabilidad de que cualquiera de ellas se convierta alguna vez en la instancia principal. Si una de ellas se convirtiera en la instancia principal, resultaría inaccesible a través de este punto de enlace, ya que especificamos el tipo `READER` en lugar del tipo `ANY`.

En el siguiente ejemplo se muestran los comandos de punto de enlace de creación y modificación, así como la salida JSON de ejemplo que muestra el estado inicial y modificado del punto de enlace personalizado.

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

$ aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier big-instances \
```

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

El punto de enlace READER predeterminado del clúster puede conectarse a las instancias de base de datos “pequeñas” o “grandes”, lo que hace poco práctico predecir el rendimiento y la escalabilidad de las consultas cuando el clúster está ocupado. Para dividir la carga de trabajo limpiamente entre los conjuntos de instancias de base de datos, puede omitir el punto de enlace READER predeterminado y crear un segundo punto de enlace personalizado que se conecte a las demás instancias de base de datos. El siguiente ejemplo lo hace creando un punto de enlace personalizado y, a continuación, añadiendo una lista de exclusión. Cualquier otra instancia de base de datos que añada al clúster posteriormente se añadirá a este punto de enlace automáticamente. El tipo ANY significa que este punto de enlace se asocia a ocho instancias en total: la instancia principal y otras siete réplicas de Aurora. Si en el ejemplo se usó el tipo READER, el punto de enlace personalizado solo se asociaría a las siete réplicas de Aurora.

```
$ aws rds create-db-cluster-endpoint --region $REGION --db-cluster-identifier custom-
endpoint-demo \
    --db-cluster-endpoint-identifier small-instances --endpoint-type any
{
    "Status": "creating",
    "DBClusterEndpointIdentifier": "small-instances",
    "CustomEndpointType": "ANY",
    "EndpointType": "CUSTOM",
```

```
    "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "StaticMembers": [],
    "ExcludedMembers": [],
    "DBClusterIdentifier": "custom-endpoint-demo",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:small-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY"
}

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

En el siguiente ejemplo se comprueba el estado de los puntos de enlace de este clúster. El clúster aún tiene su punto de enlace del clúster original, con `EndPointType` de `WRITER`, que aún usaría para la administración, ETL y otras operaciones de escritura. Aún tiene su punto de enlace `READER` original, que no usaría porque cada conexión a este podría dirigirse a una instancia de base de datos “pequeña” o “grande”. Los puntos de enlace personalizados hacen que este comportamiento sea predecible, con conexiones garantizadas para usar una de las instancias de base de datos “pequeñas” o “grandes” en función del punto de enlace que especifique.

```
$ aws rds describe-db-cluster-endpoints --region $REGION
{
```

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

```
                "custom-endpoint-demo-10",
                "custom-endpoint-demo-09"
            ],
            "EndpointType": "CUSTOM",
            "DBClusterEndpointIdentifier": "big-instances",
            "Status": "available"
        }
    ]
}
```

En el ejemplo final se muestra cómo conexiones de base de datos sucesivas a los puntos de enlace personalizados se conectan a las diversas instancias de base de datos del clúster de Aurora. El punto de enlace `small-instances` siempre se conecta a las instancias de base de datos db`.r4.4xlarge`, que son los hosts con la numeración más baja de este clúster.

```
$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+--------------------------+
| @@aurora_server_id       |
+--------------------------+
| custom-endpoint-demo-02 |
+--------------------------+

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

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

El punto de enlace `big-instances` siempre se conecta a las instancias de base de datos `db.r4.16xlarge`, que son los dos hosts con la numeración más alta de este clúster.

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

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

Clases de instancia de base de datos de Amazon Aurora

La clase de instancia de base de datos determina la capacidad de computación y de memoria de una instancia de base de datos Amazon Aurora de . La clase de instancia de base de datos que se necesite dependerá de la potencia de procesamiento y de los requisitos de memoria.

Una clase de instancia de base de datos determina tanto el tamaño como el tipo de clase de instancia de base de datos. Por ejemplo, db.r6g es una clase de instancia de base de datos de memoria optimizada con tecnología de procesadores Graviton2 de AWS. Dentro del tipo de clase de instancia db.r6g, db.r6g.2xlarge es una clase de instancia de base de datos. El tamaño de esta clase es 2xlarge.

Para obtener más información acerca de los precios de las clases de instancias, consulte Precios de Amazon RDS.

Para obtener más información sobre los tipos de clases de instancias de base de datos, los motores de base de datos compatibles, las Regiones de AWS compatibles, o las especificaciones de hardware para las clases de instancias de base de datos, consulte las siguientes secciones.

Temas

- Tipos de clase de instancia de base de datos
- Motores de base de datos compatibles para clases de instancia de base de datos
- Determinación de la compatibilidad de la clase de instancia de base de datos en Regiones de AWS
- Especificaciones de hardware para clases de instancia de base de datos para Aurora

Tipos de clase de instancia de base de datos

Amazon Aurora admite las clases de instancia de base de datos para los siguientes casos de uso:

- Aurora Serverless v2
- Optimizada para memoria
- Rendimiento ampliable
- Lecturas optimizadas

Para obtener más información sobre los tipos de instancias de Amazon EC2, consulte Tipos de instancia en la documentación de Amazon EC2.

Tipo de clase de instancia de Aurora Serverless v2

El siguiente tipo de Aurora Serverless v2 está disponible:

- db.serverless: tipo de clase de instancia de base de datos especial que utiliza Aurora Serverless v2. Aurora ajusta la computación, la memoria y la red dinámicamente a medida que cambia la carga de trabajo. Para obtener más información sobre el uso, consulte Uso de Aurora Serverless v2.

Tipos de clases de instancias optimizadas para memoria

La familia X optimizada para memoria admite los siguientes tipos de clases de instancias:

- db.x2g: clases de instancia optimizadas para aplicaciones con gran uso de la memoria y con la tecnología de los procesadores Graviton2 de AWS. Estas clases de instancias ofrecen un bajo costo por GiB de memoria.

  Puede modificar una instancia de base de datos para que utilice una de las clases de instancia de base de datos con tecnología de procesadores Graviton2 AWS. Para ello, siga los mismos pasos que con cualquier otra modificación de la instancia de base de datos.

La familia R optimizada para memoria admite los siguientes tipos de clases de instancias:

- db.r8g: clases de instancia con tecnología de procesadores Graviton4 de AWS. Estas clases de instancia son idóneas para ejecutar cargas de trabajo de uso intensivo de memoria. Estas instancias ofrecen tamaños de instancia más grandes con hasta tres veces más vCPU y memoria que las instancias db.r7g de séptima generación basadas en Graviton3 de AWS. Con tecnología del nuevo sistema Nitro AWS, una combinación de hardware dedicado e hipervisor ligero.
- Puede modificar una instancia de base de datos para que utilice una de las clases de instancia de base de datos con tecnología de procesadores Graviton4 de AWS. Para ello, siga los mismos pasos que con cualquier otra modificación de la instancia de base de datos.
- db.r7g: clases de instancias con tecnología de procesadores AWS Graviton3. Estas clases de instancia son idóneas para ejecutar cargas de trabajo de uso intensivo de memoria.

  Puede modificar una instancia de base de datos para que utilice una de las clases de instancia de base de datos con tecnología de procesadores AWS Graviton3. Para ello, siga los mismos pasos que con cualquier otra modificación de la instancia de base de datos. Con tecnología del nuevo sistema Nitro AWS, una combinación de hardware dedicado e hipervisor ligero.

- db.r7i: clases de instancia con tecnología de procesadores Intel Xeon Scalable de cuarta generación. Estas clases de instancia cuentan con certificación SAP y son idóneas para cargas de trabajo de uso intensivo de memoria en . Puede modificar una instancia de base de datos para que utilice una de las clases de instancia de base de datos con tecnología de procesadores Intel Xeon Scalable de cuarta generación. Para ello, siga los mismos pasos que con cualquier otra modificación de la instancia de base de datos. Con tecnología del nuevo sistema Nitro AWS, una combinación de hardware dedicado e hipervisor ligero.
- db.r6g: clases de instancia con tecnología de procesadores Graviton2 de AWS. Estas clases de instancia son idóneas para ejecutar cargas de trabajo de uso intensivo de memoria Con tecnología del nuevo sistema Nitro AWS, una combinación de hardware dedicado e hipervisor ligero.
- Puede modificar una instancia de base de datos para que utilice una de las clases de instancia de base de datos con tecnología de procesadores Graviton2 AWS. Para ello, siga los mismos pasos que con cualquier otra modificación de la instancia de base de datos.
- db.r6i: clases de instancia con tecnología de procesadores Intel Xeon Scalable de 3.ª generación. Estas clases de instancia cuentan con certificación SAP y son idóneas para cargas de trabajo de uso intensivo de memoria.
- db.r4: clases de instancia optimizadas para aplicaciones de uso intensivo de la memoria. Estas clases de instancia ofrecen un rendimiento mejorado en redes y de Amazon Elastic Block Store (Amazon EBS). Con tecnología del nuevo sistema Nitro AWS, una combinación de hardware dedicado e hipervisor ligero.
- db.r4: estas clases de instancias solo son compatibles con las versiones 11 y 12 de Aurora MySQL 2.x Aurora PostgreSQL. Para todos los clústeres de base de datos de Aurora que utilicen clases de instancia de base de datos db.r4, recomendamos que actualice a una clase de instancia de una generación posterior lo antes posible.

  Las clases de instancia db.r4 no están disponibles para la configuración de almacenamiento de clúster Aurora I/O-Optimized.

Tipos de clases de instancias de rendimiento ampliable

A continuación, se indican los tipos de clase de instancia de base de datos de rendimiento ampliable disponibles:

- db.t4g: clases de instancia de uso general con la tecnología de los procesadores Graviton2 de AWS basados en ARM. Estas clases de instancia ofrecen un mejor rendimiento que las clases de instancia de base de datos de rendimiento ampliable anteriores para un amplio conjunto de cargas

de trabajo de uso general ampliable. Las instancias db.t4g de Amazon RDS están configuradas para el modo ilimitado. Esto significa que pueden ampliarse más allá de la línea base en una ventana de 24 horas con cargo adicional.

Puede modificar una instancia de base de datos para que utilice una de las clases de instancia de base de datos con tecnología de procesadores Graviton2 AWS. Para ello, siga los mismos pasos que con cualquier otra modificación de la instancia de base de datos.

- db.t3: clases de instancias que proporcionan un nivel de rendimiento de referencia con la capacidad de transmitir ráfagas que usen la totalidad de la CPU. Las instancias db.t3 están configuradas para el modo ilimitado. Las clases de instancia proporcionan más capacidad de computación que las clases de instancia db.t2 anteriores. Con tecnología del nuevo sistema Nitro AWS, una combinación de hardware dedicado e hipervisor ligero. Recomendamos que se usen estas clases de instancia solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción.
- db.t2: clases de instancias que proporcionan un nivel de desempeño de referencia con la capacidad de transmitir ráfagas que usen la totalidad de la CPU. Las instancias db.t2 están configuradas para el modo ilimitado. Recomendamos que se usen estas clases de instancia solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción.

  Las clases de instancia db.t2 no están disponibles para la configuración de almacenamiento de clúster Aurora I/O-Optimized.

> **Note**
>
> Recomendamos que las clases de instancias de base de datos T se utilicen solo para servidores de desarrollo, de pruebas u otros servidores que no se utilicen para la producción. Para obtener más recomendaciones sobre las clases de instancia T, consulte Utilización de clases de instancia T para el desarrollo y la prueba.

Para las especificaciones de hardware de clase de instancia de base de datos, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

Tipos de clase de instancia de lecturas optimizadas

Los siguientes tipos de clases de instancia de lecturas optimizadas están disponibles:

- db.r8gd: clases de instancia con tecnología de procesadores Graviton4. Estas clases de instancia son ideales para ejecutar cargas de trabajo con un gran uso de memoria, y ofrecen almacenamiento local en el nivel de bloque SSD basado en NVMe para aplicaciones que necesitan almacenamiento local de alta velocidad y baja latencia. Ofrecen una memoria máxima de 1,5 TiB y hasta 11,4 TB de almacenamiento SSD basado en NVMe con conexión directa.
- db.r6g: tipo de instancia con tecnología de procesadores Graviton2 de AWS. Estas clases de instancia son ideales para ejecutar cargas de trabajo con un gran uso de memoria, y ofrecen almacenamiento local en el nivel de bloque SSD basado en NVMe para aplicaciones que necesitan almacenamiento local de alta velocidad y baja latencia.
- db.r6id: clases de instancia con tecnología de procesadores Intel Xeon Scalable de 3.ª generación. Estas clases de instancias cuentan con certificación SAP y son idóneas para cargas de trabajo de uso intensivo de memoria en bases de datos de código abierto como MySQL y PostgreSQL. Ofrecen una memoria máxima de 1 TiB y hasta 7,6 TB de almacenamiento SSD basado en NVMe con conexión directa.

Motores de base de datos compatibles para clases de instancia de base de datos

En las siguientes tablas, se muestran las clases de instancia de base de datos compatibles para motores de base de datos de Amazon Aurora.

db.serverless: clase de instancia Aurora Serverless v2 con escalado de capacidad automática

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.serverless	Consulte Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.	Consulte Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.

db.x2g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton2 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.x2g.16xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.x2g.12xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.x2g.8xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.x2g.4xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.x2g.2xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.x2g.xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.x2g.large	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores

db.r6gd: clases de instancia de lectura optimizada con tecnología de procesadores Graviton2 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6gd.16xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6gd.12xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores
db.r6gd.8xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores
db.r6gd.4xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores
db.r6gd.2xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores
db.r6gd.xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores

db.r6id: clases de instancia de lectura optimizada

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6id.32xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores
db.r6id.24xlarge	No	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.4 y versiones posteriores, 14.9 y versiones posteriores

db.r8g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton4 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r8g.48xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.24xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.16xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.12xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.8xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.4xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
		versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.2xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r8g.large	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores

db.r8gd: clases de instancia de lectura optimizada con tecnología de procesadores Graviton4 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r8gd.48xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r8gd.24xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores
db.r8gd.16xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores
db.r8gd.12xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores
db.r8gd.8xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores
db.r8gd.4xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores
db.r8gd.2xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores
db.r8g.xlarge	No	17.4 y versiones posteriores, 16.3 y versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores

db.r7g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton3 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r7g.16xlarge	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores
db.r7g.12xlarge	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores
db.r7g.8xlarge	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores
db.r7g.4xlarge	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores
db.r7g.2xlarge	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores
db.r7g.xlarge	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
		versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores
db.r7g.large	2.12.0 y versiones posteriores, 3.03.1 y versiones posteriores	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.7 y versiones posteriores, 13.10 y versiones posteriores

db.r7i: clases de instancia optimizada para memoria

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r7i.48xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.24xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.16xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r7i.12xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.8xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.4xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.2xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.xlarge	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores
db.r7i.large	3.08.0 y versiones posteriores	17.4 y versiones posteriores, 16.3 versiones posteriores, 15.7 y versiones posteriores, 14.12 y versiones posteriores, 13.15 y versiones posteriores

db.r6g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton2 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6g.16xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.r6g.12xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.r6g.8xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.r6g.4xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6g.2xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.r6g.xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores
db.r6g.large	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.8 y versiones posteriores, 11.9, 11.12 y versiones posteriores

db.r6i: clases de instancia optimizada para memoria

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6i.32xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores,

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
		13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6i.24xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6i.16xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6g.12xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6i.8xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r6i.4xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6i.2xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6i.xlarge	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores
db.r6i.large	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.5 y versiones posteriores, 12.9 y versiones posteriores

db.r5: clases de instancia optimizada para memoria

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r5.24xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.16xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.12xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.8xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.4xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.2xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.xlarge	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente
db.r5.large	Todas las versiones disponibles actualmente	Todas las versiones disponibles actualmente

db.r4: clases de instancia optimizada para memoria

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r4.16xlarge	Todas las versiones 2.x; no compatible con las versiones 3.x	Todas las versiones 11 y 12 compatibles

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.r4.8xlarge	Todas las versiones 2.x; no compatible con las versiones 3.x	Todas las versiones 11 y 12 compatibles
db.r4.4xlarge	Todas las versiones 2.x; no compatible con las versiones 3.x	Todas las versiones 11 y 12 compatibles
db.r4.2xlarge	Todas las versiones 2.x; no compatible con las versiones 3.x	Todas las versiones 11 y 12 compatibles
db.r4.xlarge	Todas las versiones 2.x; no compatible con las versiones 3.x	Todas las versiones 11 y 12 compatibles
db.r4.large	Todas las versiones 2.x; no compatible con las versiones 3.x	Todas las versiones 11 y 12 compatibles

db.t4g: clases de instancia de rendimiento ampliable con la tecnología de los procesadores Graviton2 de AWS

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.t4g.2xlarge*	No	No
db.t4g.xlarge	No	No
db.t4g.large	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.7 y versiones posteriores, 11.12 y versiones posteriores
db.t4g.medium	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
		versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.7 y versiones posteriores, 11.12 y versiones posteriores
db.t4g.small	No	No

db.t3: clases de instancia de rendimiento ampliable

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.t3.2xlarge	No	No
db.t3.xlarge	No	No
db.t3.large	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.7 y versiones posteriores, 11.12 y versiones posteriores
db.t3.medium	Todas las versiones disponibles actualmente	17.4 y versiones posteriores, 16.1 y versiones posteriores, 15.2 y versiones posteriores, 14.3 y versiones posteriores, 13.3 y versiones posteriores, 12.7 y versiones posteriores, 11.12 y versiones posteriores

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.t3.small	Todas las versiones 2.x; no compatible con las versiones 3.x	No
db.t3.micro	No	No

db.t2: clases de instancia de rendimiento ampliable

Clase de instancia	Aurora MySQL	Aurora PostgreSQL
db.t2.medium	Todas las versiones 2.x; no compatible con las versiones 3.x	No
db.t2.small	Todas las versiones 2.x; no compatible con las versiones 3.x	No

Determinación de la compatibilidad de la clase de instancia de base de datos en Regiones de AWS

Para determinar las clases de instancia de base de datos admitidas por cada motor de base de datos en una Región de AWS específica, puede utilizar uno de varios métodos. Puede utilizar el comandoConsola de administración de AWS, la página Precios de Amazon RDS o elAWS CLI comando describe-orderable-db-instance-options.

> **Note**
>
> Cuando realiza operaciones con la Consola de administración de AWS, muestra automáticamente las clases de instancia de base de datos admitidas para un motor de base de datos específico, una versión del motor de base de datos y la Región de AWS. Entre los ejemplos de operaciones que puede realizar se incluyen la creación y modificación de una instancia de base de datos.

Contenido

- Uso de la página de precios de Amazon RDS para determinar la compatibilidad de las clases de instancia de base de datos en las Regiones de AWS
- Uso de la AWS CLI para determinar la compatibilidad de la clase de instancia de base de datos en las Regiones de AWS
  - Enumeración de las clases de instancia de base de datos compatibles con una versión específica del motor de base de datos en una Región de AWS
  - Enumeración de las versiones del motor de base de datos que admiten una clase de instancia de base de datos específica en una Región de AWS

Uso de la página de precios de Amazon RDS para determinar la compatibilidad de las clases de instancia de base de datos en las Regiones de AWS

Puede utilizar la página Precios de Amazon Aurora para determinar las clases de instancia de base de datos admitidas por cada motor de base de datos en una Región de AWS específica.

Para utilizar la página de precios para determinar las clases de instancia de base de datos admitidas por cada motor de una región

1. Vaya a Precios de Amazon Aurora.
2. Elija un motor Amazon Aurora en la sección Calculadora de precios de AWS.
3. En Elija una región, elija una Región de AWS.
4. En Opción de configuración del clúster, elija una opción de configuración.
5. Utilice la sección de instancias compatibles para ver las clases de instancias de base de datos compatibles.
6. (Opcional) Elija otras opciones en la calculadora y, a continuación, elija Guardar y ver resumen o Guardar y agregar servicio.

Uso de la AWS CLI para determinar la compatibilidad de la clase de instancia de base de datos en las Regiones de AWS

Puede utilizar la AWS CLI para determinar qué clases de instancia de base de datos se admiten para los motores de base de datos específicos y las versiones de motor de base de datos en una Región de AWS.

Para utilizar los ejemplos de la AWS CLI dsiguientes, ingrese valores válidos para el motor de base de datos, la versión del motor de base de datos, la clase de instancia de base de datos y la Región de AWS. En la tabla siguiente se muestran los valores válidos del motor de base de datos.

Nombre del motor	Valor del motor en comandos de CLI	Más información acerca de las versiones
Aurora compatible con MySQL 5.7 y 8.0	`aurora-mysql`	Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 2 y actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 3 en las notas de la versión de Aurora MySQL
Aurora PostgreSQL	`aurora-postgresql`	Notas de la versión de Aurora PostgreSQL

Para obtener más información sobre los nombres de la Región de AWS, consulte AWSRegiones de .

Los siguientes ejemplos muestran cómo determinar la compatibilidad de la clase de instancia de base de datos en una Región de AWS mediante el comando describe-orderable-db-instance-options de la AWS CLI.

Temas

- Enumeración de las clases de instancia de base de datos compatibles con una versión específica del motor de base de datos en una Región de AWS
- Enumeración de las versiones del motor de base de datos que admiten una clase de instancia de base de datos específica en una Región de AWS

Enumeración de las clases de instancia de base de datos compatibles con una versión específica del motor de base de datos en una Región de AWS

Para enumerar las clases de instancia de base de datos compatibles con una versión específica del motor de base de datos en una Región de AWS, ejecute el siguiente comando.

Para Linux, macOS o:Unix

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 \
```

```
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region region
```

En:Windows

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

La salida también muestra los modos del motor que son compatibles con cada clase de instancia de base de datos.

Por ejemplo, el siguiente comando enumera las clases de instancia de base de datos compatibles para la versión 13.6 del motor de base de datos Aurora PostgreSQL en Este de EE. UU. (Norte de Virginia).

Para Linux, macOS o:Unix

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

En:Windows

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

Enumeración de las versiones del motor de base de datos que admiten una clase de instancia de base de datos específica en una Región de AWS

Para enumerar las versiones del motor de base de datos que admiten una clase de instancia de base de datos específica en una Región de AWS, ejecute el siguiente comando.

Para Linux, macOS o:Unix

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

En:Windows

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

La salida también muestra los modos de motor que son compatibles con cada versión del motor de base de datos.

Por ejemplo, el siguiente comando enumera las versiones del motor de base de datos del motor de Aurora PostgreSQL base de datos que admiten la clase de instancia de base de datos db.r5.large en US East (N. Virginia).

Para Linux, macOS o:Unix

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

En:Windows

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

Especificaciones de hardware para clases de instancia de base de datos para Aurora

En la tabla de esta sección, podrá encontrar información de hardware sobre las clases de instancias de base de datos de Amazon RDS para Aurora.

Para obtener información sobre la compatibilidad del motor de base de datos de Aurora para cada clase de instancia de base de datos, consulte Motores de base de datos compatibles para clases de instancia de base de datos.

Temas

- Terminología de hardware para clases de instancias de base de datos para Aurora
- Especificaciones de hardware para las clases de instancia optimizada para memoria
- Especificaciones de hardware para las clases de instancias de rendimiento ampliable

Terminología de hardware para clases de instancias de base de datos para Aurora

La siguiente terminología se utiliza para describir las especificaciones de hardware para clases de instancia de base de datos:

vCPU

El número de unidades de procesamiento central (CPU) virtuales. Una CPU virtual es una unidad de capacidad que se puede usar para comparar clases de instancia de base de datos. En lugar de comprar o arrendar un procesamiento concreto para usarlo durante varios meses o años, la capacidad se alquila por horas. Nuestro objetivo es proporcionar una cantidad constante y específica de capacidad de CPU dentro de los límites del hardware subyacente real.

ECU

La medida relativa de la potencia de procesamiento íntegra de una instancia de Amazon EC2. Para facilitar a los desarrolladores la comparación de la capacidad de la CPU entre distintas

clases de instancia, hemos definido una unidad de computación Amazon EC2. La cantidad de CPU asignada a una instancia concreta se expresa en términos de estas unidades informáticas EC2. Actualmente, una ECU proporciona capacidad de CPU equivalente a un procesador 2007 Opteron o 2007 Xeon de 1,0–1,2 GHz.

Memoria (GiB)

La RAM, en gibibytes, asignada a la instancia de base de datos. A menudo, hay una relación coherente entre memoria y vCPU. Como ejemplo, seleccione la clase de instancia db.r4, que dispone de una memoria en la relación de vCPU similar a la clase de instancia db.r5. Sin embargo, para la mayoría de casos de uso, la clase de instancia db.r5 proporciona un mejor rendimiento y más coherente que la clase de instancia db.r4.

Ancho de banda Ancho de banda de EBS (MB/s)

El ancho de banda máximo de EBS en megabits por segundo. Divídalo entre 8 para obtener el rendimiento esperado en megabytes por segundo.

> **Note**
>
> Esta figura hace referencia al ancho de banda de E/S para almacenamiento local dentro de la instancia de base de datos. No se aplica a la comunicación con el volumen de clúster Aurora.

Ancho de banda de red

La velocidad de red relativa a otras clases de instancia de base de datos.

Para obtener más información sobre el uso de las métricas de Amazon CloudWatch para supervisar el rendimiento de la instancia de base de datos de Aurora, consulte Evaluación de la utilización de instancias de base de datos para Aurora MySQL con métricas de Amazon CloudWatch y Evaluación de la utilización de instancias de base de datos para Aurora PostgreSQL con métricas de CloudWatch.

Especificaciones de hardware para las clases de instancia optimizada para memoria

En las siguientes tablas, aparecen las especificaciones de computación, memoria, almacenamiento y ancho de banda para las clases de instancia optimizada para memoria.

db.x2g: clases de instancia optimizada para memoria con procesadores Graviton2 de AWS

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.x2g.16xlarge	64	—	1024	Solo optimizado para EBS	19 000	25
db.x2g.12xlarge	48	—	768	Solo optimizado para EBS	14 250	20
db.x2g.8xlarge	32	—	512	Solo optimizado para EBS	9500	12
db.x2g.4xlarge	16	—	256	Solo optimizado para EBS	4750	Hasta 10
db.x2g.2xlarge	8	—	128	Solo optimizado para EBS	Hasta 4750.	Hasta 10
db.x2g.xlarge	4	—	64	Solo optimizado para EBS	Hasta 4750.	Hasta 10
db.x2g.large	2	—	32	Solo optimizado para EBS	Hasta 4750.	Hasta 10

db.r8gd: clases de instancia optimizada para memoria con tecnología de procesadores Graviton4 de AWS y el almacenamiento en SSD

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r8gd.48xlarge	192	—	1536	SSD NVMe de 6 x 1900	40 000	50
db.r8gd.24xlarge	96	—	768	SSD NVMe de 3 x 1900	30.000	40
db.r8gd.16xlarge	64	—	512	SSD NVMe de 2 x 1900	20 000	30
db.r8gd.12xlarge	48	—	384	SSD NVMe de 3 x 950	15.000	22.5
db.r8gd.8xlarge	32	—	256	SSD NVMe de 1 x 1900	10 000	15
db.r8gd.4xlarge	16	—	128	SSD NVMe de 1 x 950	Hasta 10 000	Hasta 15
db.r8gd.2xlarge	8	—	64	SSD NVMe de 1 x 474	Hasta 10 000	Hasta 15
db.r8gd.xlarge	4	—	32	SSD NVMe de 1 x 237	Hasta 10 000	Hasta 12,5
db.r8gd.large	2	—	16	SSD NVMe de 1 x 118	Hasta 10 000	Hasta 12,5

db.r8g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton4 de AWS

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r8g.48xlarge	192	—	1536	Solo optimizado para EBS	40 000	50
db.r8g.24xlarge	96	—	768	Solo optimizado para EBS	30.000	40
db.r8g.16xlarge	64	—	512	Solo optimizado para EBS	20 000	30
db.r8g.12xlarge	48	—	384	Solo optimizado para EBS	15.000	22.5
db.r8g.8xlarge	32	—	256	Solo optimizado para EBS	10 000	15
db.r8g.4xlarge	16	—	128	Solo optimizado para EBS	Hasta 10 000	Hasta 15
db.r8g.2xlarge	8	—	64	Solo optimizado para EBS	Hasta 10 000	Hasta 15
db.r8g.xlarge	4	—	32	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r8g.large	2	—	16	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5

db.r7i: clases de instancia optimizada para memoria con tecnología de procesadores Intel Xeon Scalable de 4.ª generación

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r7i.48xlarge	192	—	1536	Solo optimizado para EBS	40 000	50
db.r7i.24xlarge	96	—	768	Solo optimizado para EBS	30.000	37,5
db.r7i.16xlarge	64	—	512	Solo optimizado para EBS	20 000	25
db.r7i.12xlarge	48	—	384	Solo optimizado para EBS	15.000	18.75
db.r7i.8xlarge	32	—	256	Solo optimizado para EBS	10 000	12,5

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r7i.4xlarge	16	—	128	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5
db.r7i.2xlarge	8	—	64	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5
db.r7i.xlarge	4	—	32	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5
db.r7i.large	2	—	16	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5

db.r7g: clases de instancia optimizada para memoria con procesadores Graviton3 de AWS

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r7g.16xlarge	64	—	512	Solo optimizado para EBS	20 000	30
db.r7g.12xlarge	48	—	384	Solo optimizado para EBS	15.000	22.5

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r7g.8xlarge	32	—	256	Solo optimizado para EBS	10 000	15
db.r7g.4xlarge	16	—	128	Solo optimizado para EBS	Hasta 10 000	Hasta 15
db.r7g.2xlarge	8	—	64	Solo optimizado para EBS	Hasta 10 000	Hasta 15
db.r7g.xlarge	4	—	32	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5
db.r7g.large	2	—	16	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5

db.r6id: clases de instancia optimizada para memoria con procesadores escalables Intel Xeon de 3.ª generación y almacenamiento SSD

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r6id.32xlarge	128	—	1 024	SSD NVMe de 4 x 1900	40 000	50

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r6id.24xlarge	96	—	768	SSD NVMe de 4 x 1425	30.000	37,5

db.r6gd: clases de instancia optimizada para memoria con procesadores Graviton2 de AWS y almacenamiento SSD

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r6gd.16xlarge	64	—	512	SSD NVMe de 2 x 1900	19 000	25
db.r6gd.12xlarge	48	—	384	SSD NVMe de 2 x 1425	13 500	20
db.r6gd.8xlarge	32	—	256	SSD NVMe de 1 x 1900	9,000	12
db.r6gd.4xlarge	16	—	128	SSD NVMe de 1 x 950	4750	Hasta 10
db.r6gd.2xlarge	8	—	64	SSD NVMe de 1 x 474	Hasta 4750.	Hasta 10
db.r6gd.xlarge	4	—	32	SSD NVMe de 1 x 237	Hasta 4750.	Hasta 10

db.r6g - clases de instancia optimizada para memoria con procesadores Graviton2 de AWS

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r6g.16xlarge	64	—	512	Solo optimizado para EBS	19 000	25
db.r6g.12xlarge	48	—	384	Solo optimizado para EBS	13 500	20
db.r6g.8xlarge	32	—	256	Solo optimizado para EBS	9,000	12
db.r6g.4xlarge	16	—	128	Solo optimizado para EBS	4750	Hasta 10
db.r6g.2xlarge	8	—	64	Solo optimizado para EBS	Hasta 4750.	Hasta 10
db.r6g.xlarge	4	—	32	Solo optimizado para EBS	Hasta 4750.	Hasta 10
db.r6g.large	2	—	16	Solo optimizado para EBS	Hasta 4750.	Hasta 10

db.r6i: clases de instancia optimizada para memoria con procesadores Intel Xeon Scalable de 3.ª generación

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r6i.32xlarge	128	—	1 024	Solo optimizado para EBS	40 000	50
db.r6i.24xlarge	96	—	768	Solo optimizado para EBS	30.000	37,5
db.r6i.16xlarge	64	—	512	Solo optimizado para EBS	20 000	25
db.r6g.12xlarge	48	—	384	Solo optimizado para EBS	15.000	18.75
db.r6i.8xlarge	32	—	256	Solo optimizado para EBS	10 000	12,5
db.r6i.4xlarge	16	—	128	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5
db.r6i.2xlarge	8	—	64	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5
db.r6i.xlarge	4	—	32	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r6i.large	2	—	16	Solo optimizado para EBS	Hasta 10 000	Hasta 12,5

db.r5: clases de instancia optimizada para memoria

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r5.24xlarge	96	347	768	Solo optimizado para EBS	19 000	25
db.r5.16xlarge	64	264	512	Solo optimizado para EBS	13 600	20
db.r5.12xlarge	48	173	384	Solo optimizado para EBS	9500	12
db.r5.8xlarge	32	132	256	Solo optimizado para EBS	6800	10
db.r5.4xlarge	16	71	128	Solo optimizado para EBS	4750	Hasta 10

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r5.2xlarge	8	38	64	Solo optimizado para EBS	Hasta 4750.	Hasta 10
db.r5.xlarge	4	19	32	Solo optimizado para EBS	Hasta 4750.	Hasta 10
db.r5.large	2	10	16	Solo optimizado para EBS	Hasta 4750.	Hasta 10

db.r4: clases de instancia optimizada para memoria con procesadores Intel Xeon Scalable

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r4.16xlarge	64	195	488	Solo optimizado para EBS	14 000	25
db.r4.8xlarge	32	99	244	Solo optimizado para EBS	7000	10
db.r4.4xlarge	16	53	122	Solo optimizado para EBS	3500	Hasta 10

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.r4.2xlarge	8	27	61	Solo optimizado para EBS	1.700	Hasta 10
db.r4.xlarge	4	13.5	30.5	Solo optimizado para EBS	850	Hasta 10
db.r4.large	2	7	15.25	Solo optimizado para EBS	425	Hasta 10

Especificaciones de hardware para las clases de instancias de rendimiento ampliable

En las siguientes tablas, aparecen las especificaciones de computación, memoria, almacenamiento y ancho de banda para las clases de instancia de rendimiento ampliable.

db.t4g: clases de instancia de rendimiento ampliable con la tecnología de los procesadores Graviton2 de AWS

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.t4g.large	2	—	8	Solo optimizado para EBS	Hasta 2048.	Hasta 5
db.t4g.medium	2	—	4	Solo optimizado para EBS	Hasta 2085	Hasta 5

db.t3: clases de instancia de rendimiento ampliable

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.t3.large	2	Variab	8	Solo optimizado para EBS	Hasta 2048.	Hasta 5
db.t3.medium	2	Variab	4	Solo optimizado para EBS	Hasta 1536.	Hasta 5
db.t3.small	2	Variab	2	Solo optimizado para EBS	Hasta 1536.	Hasta 5

db.t2: clases de instancia de rendimiento ampliable

Clase de instancia	vCPU	ECU	Memoria (GiB)	Almacenamiento de la instancia (GiB)	Ancho de banda Ancho de banda de EBS (MB/s)	Ancho de banda de la red (Gbps)
db.t2.medium	2	Variab	4	EBS solo	—	Moderado
db.t2.small	1	Variab	2	EBS solo	—	Bajo

Almacenamiento de Amazon Aurora

A continuación, puede obtener más información acerca del subsistema de almacenamiento Aurora. Aurora usa una arquitectura de almacenamiento distribuida y compartida que es un factor importante en el rendimiento, la escalabilidad y la fiabilidad para los clústeres de Aurora.

Temas

- Información general del almacenamiento de Amazon Aurora
- Qué contiene el volumen del clúster
- Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora
- Cómo cambia automáticamente el tamaño del almacenamiento de Aurora
- Cómo se factura el almacenamiento de datos de Aurora

Información general del almacenamiento de Amazon Aurora

Los datos de Aurora se almacenan en el volumen del clúster, que es un volumen único y virtual que usa unidades de estado sólido (SSD). Un volumen de clúster se compone de copias de los datos repartidas entre tres zonas de disponibilidad de una sola región de AWS. Como los datos se replican automáticamente entre las distintas zonas de disponibilidad, tienen una larga duración y se reduce el riesgo de pérdida de datos. Esta replicación también garantiza que su base de datos esté más disponible durante una conmutación por error. Esto es así porque las copias de datos ya existen en las otras zonas de disponibilidad y continúan respondiendo a las solicitudes de datos de las instancias de base de datos de su clúster de bases de datos. La cantidad de réplicas es independiente del número de instancias de base de datos de su clúster.

Aurora utiliza un almacenamiento local independiente para los archivos temporales no persistentes. Esto incluye archivos que se utilizan para fines tales como ordenar conjuntos de datos grandes durante el procesamiento de consultas y la creación de índices. Para obtener más información, consulte Límites de almacenamiento temporal de Aurora MySQL y Límites de almacenamiento temporal de Aurora PostgreSQL.

Qué contiene el volumen del clúster

El volumen del clúster de Aurora contiene todos sus datos de usuario, objetos de esquema y metadatos internos como las tablas del sistema y el registro binario. Por ejemplo, Aurora almacena todas las tablas, índices, objetos binarios grandes (BLOB), procedimientos almacenados, etc. para un clúster de Aurora en el volumen del clúster.

La arquitectura de almacenamiento compartida de Aurora hace que sus datos sean independientes de las instancias de base de datos del clúster. Por ejemplo, puede añadir una instancia de base de datos rápidamente, ya que Aurora no hace una nueva copia de los datos de la tabla. En su lugar, la instancia de base de datos se conecta al volumen compartido que ya contiene todos sus datos. Puede quitar una instancia de base de datos de un clúster sin quitar los datos subyacentes del clúster. Solo al eliminar todo el clúster, Aurora elimina los datos.

Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora

Amazon Aurora tiene dos configuraciones de almacenamiento para los clústeres de base de datos:

- Aurora I/O-Optimized: mejora de la relación entre precio y rendimiento y la previsibilidad para las aplicaciones con uso intensivo de E/S. Solo paga por el uso y el almacenamiento de sus clústeres de base de datos, sin cargos adicionales por las operaciones de E/S de lectura y escritura.

  Aurora I/O-Optimized es la mejor opción cuando su gasto en E/S es igual o superior al 25 % del gasto total en bases de datos de Aurora.

  Puede elegir Aurora I/O-Optimized cuando cree o modifique un clúster de base de datos con una versión de motor de datos que admita la configuración del clúster de Aurora I/O-Optimized. Puede cambiar de Aurora I/O-Optimized a Aurora Standard en cualquier momento.
- Aurora Standard: precios rentables para muchas aplicaciones con un uso moderado de E/S. Además del uso y el almacenamiento de sus clústeres de base de datos, también paga una tarifa estándar por cada millón de solicitudes de operaciones de E/S.

  Aurora Standard es la mejor opción cuando su gasto en E/S es inferior al 25 % del gasto total en bases de datos de Aurora.

  Puede cambiar de Aurora Standard a Aurora I/O-Optimized una vez cada 30 días. Al cambiar entre las opciones de almacenamiento Aurora Standard y Aurora I/O-Optimized para instancias de base de datos que no están basadas en NVMe, no se produce ningún tiempo de inactividad. Sin embargo, en el caso de las instancias de bases de datos basadas en NVMe, el cambio entre las opciones de almacenamiento Aurora I/O-Optimized y Aurora Standard requiere reiniciar el motor de base de datos, lo que puede provocar un breve tiempo de inactividad.

Para obtener más información sobre Región de AWS y la compatibilidad de versiones, consulte Regiones y motores de base de datos Aurora admitidos para configuraciones de almacenamiento en clúster.

Para obtener más información acerca de los precios de las configuraciones de almacenamiento de Aurora, consulte Precios de Amazon Aurora.

Para obtener información sobre cómo elegir la configuración de almacenamiento al crear un clúster de base de datos, consulte Creación de un clúster de base de datos. Para obtener información

sobre cómo modificar la configuración de almacenamiento de un clúster de base de datos, consulte Configuración para Amazon Aurora.

Cómo cambia automáticamente el tamaño del almacenamiento de Aurora

Los volúmenes de clúster de Aurora crecen automáticamente a medida que se incrementa la cantidad de datos de la base de datos. Para obtener información sobre los tamaños máximos de volumen del clúster Aurora para cada versión de motor, consulte Límites de tamaño de Amazon Aurora. Este escalado automático de almacenamiento se combina con un subsistema de almacenamiento de alto rendimiento y altamente distribuido. Esto hace de Aurora una buena elección para sus datos empresariales importantes cuando los objetivos principales son la fiabilidad y la alta disponibilidad.

Para mostrar el estado del volumen, consulte Visualización del estado del volumen para un clúster de base de datos de Aurora MySQL o Visualización del estado del volumen para un clúster de bases de datos de Aurora PostgreSQL. Para encontrar formas de equilibrar los costos de almacenamiento con otras prioridades, Escalado del almacenamiento describe cómo monitorear las métricas `AuroraVolumeBytesLeftTotal` y `VolumeBytesUsed` de Amazon Aurora en CloudWatch.

Cuando se eliminan datos de Aurora, se libera el espacio asignado a esos datos. Algunos ejemplos de eliminación de datos incluyen la eliminación o el truncamiento de una tabla. Esta reducción automática del uso de almacenamiento le ayuda a minimizar los cargos de almacenamiento.

> **Note**
>
> Los límites de almacenamiento y el comportamiento dinámico de cambio de tamaño que se describen aquí se aplican a las tablas persistentes y otros datos almacenados en el volumen del clúster.
>
> Para Aurora PostgreSQL, los datos de las tablas temporales se almacenan en la instancia de base de datos local.
>
> Para la versión 2 de Aurora MySQL, los datos de las tablas temporales se almacenan de forma predeterminada en el volumen del clúster para las instancias de escritura y en el almacenamiento local para las instancias de lectura. Para obtener más información, consulte Motor de almacenamiento para tablas temporales en disco.
>
> Para la versión 3 de Aurora MySQL, los datos de la tabla temporal se almacenan en la instancia de base de datos local o en el volumen del clúster. Para obtener más información, consulte Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3.
>
> El tamaño máximo de las tablas temporales que residen en el almacenamiento local está limitado por el tamaño máximo de almacenamiento local de la instancia de base de datos. El

tamaño del espacio de almacenamiento depende de la clase de instancia que utilice. Para obtener más información, consulte Límites de almacenamiento temporal de Aurora MySQL y Límites de almacenamiento temporal de Aurora PostgreSQL.

Algunas características de almacenamiento, como el tamaño máximo de un volumen de clúster y el cambio de tamaño automático cuando se eliminan datos, dependen de la versión de Aurora del clúster. Para obtener más información, consulte Escalado del almacenamiento. También puede aprender a evitar problemas de almacenamiento y a monitorear el almacenamiento asignado y el espacio libre en el clúster.

Cómo se factura el almacenamiento de datos de Aurora

Aunque un volumen de clúster de Aurora puede aumentar hasta 256 tebibytes (TiB) para versiones de motor específicas, solo se le cobra por el espacio que utilice en un volumen de clúster de Aurora. En versiones anteriores de Aurora, el volumen del clúster podía reutilizar el espacio liberado cuando eliminaba datos, pero el espacio de almacenamiento asignado nunca disminuiría. Ahora, cuando se eliminan datos de Aurora, por ejemplo, al eliminar una tabla o una base de datos, el espacio asignado general disminuye en una cantidad comparable. Por lo tanto, puede reducir los cargos de almacenamiento eliminando tablas, índices, bases de datos, etc. que ya no necesite.

Tip

Para versiones anteriores sin la característica de cambio de tamaño dinámico, restablecer el uso de almacenamiento de un clúster implicaba realizar un volcado lógico y restaurar un clúster nuevo. Esa operación puede tardar mucho tiempo en obtener un volumen sustancial de datos. Si se encuentra con esta situación, plantéese la posibilidad de actualizar el clúster a una versión que admita el cambio de tamaño dinámico del volumen.

Para obtener información sobre qué versiones de Aurora admiten el cambio de tamaño dinámico y cómo minimizar los gastos de almacenamiento mediante la supervisión del uso del almacenamiento de su clúster, consulte Escalado del almacenamiento. Para obtener información sobre la factuación del almacenamiento de las copias de seguridad de Aurora, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora. Para obtener información acerca del almacenamiento de datos de Aurora, consulte Precios de Amazon RDS para Aurora.

Fiabilidad de Amazon Aurora

Aurora se ha diseñado para ofrecer fiabilidad, durabilidad y tolerancia a errores. Puede diseñar la arquitectura de su clúster de bases de datos Aurora para mejorar la disponibilidad por medio de acciones como añadir réplicas de Aurora y situarlas en distintas zonas de disponibilidad, y Aurora incluye también varias características automáticas que la convierten en una solución de base de datos confianza.

Temas

- Reparación automática del almacenamiento
- Caché de páginas que puede sobrevivir
- Recuperación de reinicios no planificados

Reparación automática del almacenamiento

Como Aurora mantiene varias copias de sus datos en tres zonas de disponibilidad, el riesgo de perder datos como resultado de un error de disco se reduce sustancialmente. Aurora también detecta automáticamente los errores de los volúmenes de disco que integran el volumen de clúster. Cuando se produce un error en un segmento de un volumen de disco, Aurora repara inmediatamente el segmento. Cuando Aurora repara el segmento de disco, utiliza los datos de los otros volúmenes que componen el volumen de clúster para garantizar que los datos del segmento reparado están actualizados. Como resultado, Aurora evita las pérdidas de datos y reduce la necesidad de realizar una restauración a un momento dado para recuperarse cuando se produce un error del disco.

Caché de páginas que puede sobrevivir

En Aurora, la memoria caché de páginas de cada instancia de base de datos se administra en un proceso independiente de la base de datos, lo que permite a la memoria caché de páginas sobrevivir con independencia de la base de datos. (La memoria caché de páginas también se denomina grupo de búferes InnoDB en Aurora MySQL y memoria caché de búferes en Aurora PostgreSQL).

En el improbable caso de que se produzca un fallo en la base de datos, la memoria caché de páginas permanece en la memoria, lo que mantiene las páginas de datos actuales "calientes" en la memoria caché de páginas cuando se reinicia la base de datos. Esto proporciona una ventaja de rendimiento al evitar la necesidad de que las consultas iniciales ejecuten operaciones de E/S de lectura para "calentar" la memoria caché de páginas.

En el caso de Aurora MySQL, el comportamiento de la memoria caché de páginas al reiniciar y conmutar por error es el siguiente:

- Puede reiniciar la instancia de escritor sin reiniciar las instancias de lector.
  - Si las instancias del lector no se reinician cuando se reinicia la instancia del escritor, no pierden su memoria caché de páginas.
  - Si las instancias del lector se reinician cuando se reinicia la instancia del escritor, pierden su memoria caché de páginas.
- Cuando se reinicia una instancia del lector, la memoria caché de páginas de las instancias del escritor y del lector sobreviven.
- Cuando el clúster de base de datos falla, el efecto es similar a cuando se reinicia una instancia del escritor. En la nueva instancia del escritor (anteriormente era la instancia del lector) la memoria caché de páginas sobrevive, pero en la instancia del lector (anteriormente era la instancia del escritor), la memoria caché de páginas no sobrevive.

En el caso de Aurora PostgreSQL, puede utilizar la administración de la memoria caché de páginas del clúster para preservar la memoria caché de páginas de una instancia del lector designada que se convierte en la instancia del escritor después de la conmutación por error. Para obtener más información, consulte Recuperación rápida después de una conmutación por error con la administración de caché del clúster para Aurora PostgreSQL.

Recuperación de reinicios no planificados

Aurora se ha diseñado para recuperarse de reinicio no planificado casi instantáneamente y continuar sirviendo sus datos de aplicación sin el registro binario. Aurora se recupera de forma asíncrona en subprocesos paralelos, de forma que su base de datos permanece abierta y disponible inmediatamente después de un reinicio no planificado.

Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora y Optimizaciones para reducir el tiempo de reinicio de la base de datos.

A continuación se indican consideraciones para registro binario y recuperación de reinicio no planificado en Aurora MySQL:

- Habilitar el registro binario en Aurora afecta directamente al tiempo de recuperación tras un reinicio no planificado, ya que fuerza a la instancia de base de datos a realizar la recuperación de log binario.

- El tipo de registro binario utilizado afecta al tamaño y a la eficiencia del registro. Para la misma cantidad de actividad de base de datos, algunos formatos registran más información que otros en los logs binarios. La siguiente configuración del parámetro `binlog_format` da lugar a distintas cantidades de datos de log:
  - `ROW`: la mayor cantidad de datos de registro
  - `STATEMENT`: la menor cantidad de datos de registro
  - `MIXED`: una cantidad moderada de datos de registro que habitualmente ofrece la mejor combinación de integridad de datos y rendimiento

  La cantidad de datos de log binario afecta al tiempo de recuperación. Si hay más datos registrados en los logs binarios, la instancia de base de datos debe procesar más datos durante la recuperación, lo que aumenta el tiempo de recuperación.
- Para reducir la sobrecarga computacional y mejorar los tiempos de recuperación con el registro binario, puede utilizar el binlog mejorado. El binlog mejorado acelera el tiempo de recuperación de la base de datos hasta en un 99 %. Para obtener más información, consulte Configuración del binlog mejorado para Aurora MySQL.
- Aurora no necesita que los logs binarios repliquen datos dentro de un clúster de bases de datos o que realicen una restauración a un momento dado (PITR).
- Si no necesita el log binario para replicación externa (o un flujo de log binario externo), le recomendamos que establezca el parámetro `binlog_format` en `OFF` para deshabilitar el registro binario. Al hacerlo se reduce el tiempo de recuperación.

Para obtener más información sobre el registro binario de Aurora y la replicación, consulte Replicación con Amazon Aurora. Para obtener más información acerca de las implicaciones de distintos tipos de replicación de MySQL, consulte Advantages and Disadvantages of Statement-Based and Row-Based Replication en la documentación de MySQL.

Seguridad de Amazon Aurora

La seguridad de Amazon Aurora se administra en tres niveles:

- Para controlar quién puede realizar acciones de administración de Amazon RDS en clústeres e instancias de base de datos de Aurora, se usa AWS Identity and Access Management (IAM). Cuando se conecta a AWS con credenciales de IAM, la cuenta de AWS debe tener políticas de IAM que concedan los permisos necesarios para realizar operaciones de administración de

Amazon RDS. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.

Si usa IAM para acceder a la consola de Amazon RDS, debe iniciar sesión primero en la Consola de administración de AWS con sus credenciales de usuario y luego ir a la consola de Amazon RDS en https://console.aws.amazon.com/rds.

- Los clústeres de base de datos de Aurora deben crearse en una nube virtual privada (VPC) basada en el servicio de Amazon VPC. Para controlar qué dispositivos e instancias Amazon EC2 pueden abrir conexiones al punto de enlace y al puerto de la instancia de base de datos los clústeres de base de datos Aurora en una VPC, debe usar un grupo de seguridad de VPC. Puede establecer estas conexiones de puerto y punto de enlace mediante Transport Layer Security (TLS)/ Capa de conexión segura (SSL). Además, las reglas del firewall de su compañía pueden controlar si los dispositivos que se ejecutan en ella pueden abrir conexiones a una instancia de base de datos. Para obtener más información acerca de las VPC, consulte VPC de Amazon y Amazon Aurora.
- Para autenticar los inicios de sesión y los permisos de un clúster de bases de datos Amazon Aurora, puede usar cualquiera de los siguientes procedimientos o una combinación de ellos.
  - Puede seguir el mismo procedimiento que con una instancia de base de datos independiente de MySQL o PostgreSQL.

    Las técnicas de autenticación de inicios de sesión y permisos para instancias de base de datos independientes de MySQL o PostgreSQL como, por ejemplo, el uso de los comandos SQL o la modificación de las tablas de los esquemas de las bases de datos, también funcionan con Aurora. Para obtener más información, consulte Seguridad con Amazon Aurora MySQL o Seguridad con Amazon Aurora PostgreSQL.
  - Puede utilizar la autenticación de base de datos de IAM.

    Con la autenticación de bases de datos de IAM, debe autenticarse en el clúster de bases de datos de Aurora con un usuario o con un rol de IAM y un token de autenticación. Un token de autenticación es un valor único que se genera utilizando el proceso de firma Signature Version 4. Mediante la autenticación de base de datos de IAM, puede utilizar las mismas credenciales para controlar el acceso a AWS los recursos y a las bases de datos. Para obtener más información, consulte Autenticación de bases de datos de IAM .
  - Puede usar la autenticación Kerberos para Aurora PostgreSQL y Aurora MySQL.

    Puede usar Kerberos para autenticar a los usuarios cuando se conecten al clúster de bases de datos de Aurora PostgreSQL y Aurora MySQLDB. En este caso, el clúster de bases de datos

funciona con AWS Directory Service for Microsoft Active Directory para habilitar la autenticación Kerberos. AWS Directory Service for Microsoft Active Directory también se llama AWS Managed Microsoft AD. Mantener todas las credenciales en el mismo directorio puede ahorrarle tiempo y esfuerzo. Dispone de un lugar centralizado para almacenar y administrar credenciales para varios clústeres de bases de datos. El uso de un directorio también puede mejorar su perfil de seguridad general. Para obtener más información, consulte Uso de la autenticación Kerberos con Aurora PostgreSQL y Uso de la autenticación Kerberos para Aurora MySQL.

Para obtener información acerca de la configuración de seguridad, consulte Seguridad en Amazon Aurora.

Uso de SSL con clústeres de base de datos de Aurora

Los clústeres de base de datos Amazon Aurora admiten conexiones de Capa de conexión segura (SSL) desde aplicaciones con el mismo proceso y la misma clave pública que las instancias de base de datos de Amazon RDS. Para obtener más información, consulte Seguridad con Amazon Aurora MySQL, Seguridad con Amazon Aurora PostgreSQL o Uso de TLS/SSL con Aurora Serverless v1.

Alta disponibilidad para Amazon Aurora

La arquitectura de Amazon Aurora implica la separación del almacenamiento y el cómputo. Aurora incluye algunas características de alta disponibilidad que se aplican a los datos de su clúster de base de datos. Los datos se mantienen seguros incluso si alguno a todas las instancias de base de datos del clúster dejan de estar disponibles. Otras características de alta disponibilidad se aplican a las instancias de base de datos. Estas características ayudan a garantizar que una o varias instancias de base de datos estén preparadas para gestionar las solicitudes de base de datos desde su aplicación.

Temas

- Alta disponibilidad para datos de Aurora
- Alta disponibilidad para instancias de bases de datos de Aurora
- Alta disponibilidad en todas las regiones de AWS con bases de datos de Aurora globales
- Tolerancia a errores para un clúster de base de datos de Aurora
- Alta disponibilidad con Amazon RDS Proxy

Alta disponibilidad para datos de Aurora

Aurora almacena copias de los datos en un clúster de base de datos en varias zonas de disponibilidad en una única Región de AWS. Aurora almacena estas copias independientemente de si las instancias de base de datos en el clúster de base de datos abarcan varias zonas de disponibilidad. Para obtener más información sobre Aurora, consulte Administración de un clúster de base de datos de Amazon Aurora.

Cuando los datos se escriben en la instancia de base de datos principal, Aurora replica de forma síncrona los datos en zonas de disponibilidad a seis nodos de almacenamiento asociados con el volumen de clúster. Este enfoque proporciona redundancia de datos, elimina los bloqueos de E/S y minimiza los picos de latencia durante las copias de seguridad del sistema. Ejecutar una instancia de base de datos con alta disponibilidad puede mejorar la disponibilidad durante el mantenimiento de sistema planificado y ayuda a proteger las bases de datos contra los errores y las interrupciones de las zonas de disponibilidad. Para obtener más información acerca de las zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.

Alta disponibilidad para instancias de bases de datos de Aurora

Después de crear la instancia principal (de escritor), puede crear hasta 15 réplicas de Aurora de solo lectura. Las réplicas de Aurora también se conocen como instancias de lector. Las réplicas de Aurora utilizan la replicación asíncrona para soportar una alta disponibilidad sin afectar al rendimiento de la instancia principal.

Durante las operaciones diarias, puede descargar parte del trabajo para aplicaciones de lectura intensiva utilizando las instancias del lector para procesar consultas `SELECT`. Cuando un problema afecta a la instancia principal, una de estas instancias de lector toma el relevo como instancia principal. Este mecanismo se conoce como failover. Muchas características de Aurora se aplican al mecanismo de conmutación por error. Por ejemplo, Aurora detecta problemas de base de datos y activa automáticamente el mecanismo de conmutación por error cuando sea necesario. Aurora también tiene características que reducen el tiempo de finalización de la conmutación por error. Al hacerlo, se minimiza el tiempo en que la base de datos no está disponible para escribir durante una conmutación por error.

Aurora se ha diseñado para recuperarse lo más rápido posible y, por lo general, el camino más rápido hacia la recuperación es reiniciar o realizar una conmutación por error a la misma instancia de base de datos. El reinicio es más rápido e implica menos sobrecarga que la conmutación por error.

Para utilizar una cadena de conexión que permanece igual incluso cuando una conmutación por error promueve una nueva instancia principal, se conecta al punto de enlace del clúster. El punto de enlace del clúster siempre representa la instancia principal actual del clúster. Para obtener más información sobre el punto de enlace del clúster, consulte Conexiones de puntos de conexión de Amazon Aurora.

> **Tip**
>
> Dentro de cada Región de AWS, las zonas de disponibilidad representan ubicaciones distintas entre sí para proporcionar aislamiento en caso de interrupciones. Se recomienda distribuir la instancia principal y las instancias de lector del clúster de base de datos entre varias AZ para mejorar la disponibilidad del clúster de base de datos. De esta forma, un problema que afecta a una AZ completa no causará una interrupción en el clúster.
> Puede configurar un clúster de base de datos multi-AZ haciendo una elección sencilla al crear el clúster. Puede utilizar la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS. También puede convertir un clúster de base de datos de Aurora existente en uno multi-AZ añadiendo una nueva instancia de base de datos de lector y especificando una AZ distinta.

Alta disponibilidad en todas las regiones de AWS con bases de datos de Aurora globales

Para una alta disponibilidad en varias Regiones de AWS, puede configurar bases de datos de Aurora globales. Cada base de datos global de Aurora abarca varias Regiones de AWS, lo que permite lecturas globales de baja latencia y la recuperación de desastres de interrupciones en una Región de AWS. Aurora replica de forma asíncrona todos los datos y actualizaciones desde la Región de AWS principal a cada una de las regiones secundarias. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

Tolerancia a errores para un clúster de base de datos de Aurora

Un clúster de base de datos de Aurora ofrece tolerancia a errores por diseño. El volumen del clúster abarca varias zonas de disponibilidad en una única Región de AWS y cada zona de disponibilidad contiene una copia de los datos del volumen del clúster. Esta funcionalidad significa que el clúster de base de datos puede tolerar un error de una zona de disponibilidad sin perder datos y con tan solo una interrupción breve del servicio.

Si se produce un error en la instancia principal de un clúster de base de datos, Aurora conmuta automáticamente a una nueva instancia principal de una de las dos formas siguientes:

- Promoviendo una réplica de Aurora ya existente a nueva instancia principal
- Creando una nueva instancia principal

Si el clúster de base de datos tiene una o varias réplicas de Aurora, se promueve una réplica de Aurora a instancia principal durante un evento de error. Un evento de error provoca una interrupción breve durante la cual las operaciones de lectura y escritura generan errores con una excepción. Sin embargo, el servicio se suele restaurar en menos de 60 segundos y, en muchos casos, en menos de 30 segundos. Para aumentar la disponibilidad de su clúster de base de datos, es recomendable que cree al menos una o varias réplicas de Aurora en dos o más zonas de disponibilidad diferentes.

> **Tip**
>
> En Aurora MySQL, puede mejorar la disponibilidad durante una conmutación por error si tiene más de una instancia de base de datos de lector en un clúster. En Aurora MySQL, Aurora reinicia solo la instancia de base de datos de escritor y la instancia de lector a la que se conmuta por error. Otras instancias del lector en el clúster siguen disponibles durante una conmutación por error para continuar el procesamiento de consultas mediante conexiones con el punto de conexión del lector.
>
> También puede mejorar la disponibilidad durante una conmutación por error mediante el uso de RDS Proxy con el clúster de bases de datos de Aurora. Para obtener más información, consulte Alta disponibilidad con Amazon RDS Proxy.

Puede personalizar el orden en que se promueven las réplicas de Aurora a instancia principal tras un error mediante la asignación de una prioridad a cada réplica. Las prioridades van desde 0 para la prioridad más alta hasta 15 para la más baja. Si la instancia principal falla, Amazon RDS promueve la réplica de Aurora con la prioridad más alta a la nueva instancia principal. Puede modificar la prioridad de una réplica de Aurora en cualquier momento. Al modificar la prioridad, no se activa una conmutación por error.

Puede haber más de una réplica de Aurora con la misma prioridad, lo que genera niveles de promoción. Si dos o más réplicas de Aurora comparten la misma prioridad, Amazon RDS promueve la réplica que tiene un tamaño mayor. Si dos o más réplicas de Aurora tienen la misma prioridad y el mismo tamaño, Amazon RDS promueve una réplica arbitraria del mismo nivel de promoción.

> **Note**
>
> En la identificación de un objetivo de conmutación por error intervienen varios factores. Tras cinco intentos fallidos de conmutación por error, ya no se tienen en cuenta los niveles de promoción.

Si el clúster de base de datos no contiene ninguna Réplica de Aurora, la instancia principal se vuelve a crear en la misma AZ durante un evento de error. Un evento de error provoca una interrupción durante la cual las operaciones de lectura y escritura generan errores con una excepción. El servicio se restaura cuando se crea la nueva instancia principal, un proceso que normalmente dura menos de 10 minutos. Promover una réplica de Aurora a instancia principal es mucho más rápido que crear una nueva instancia principal.

Supongamos que la instancia principal del clúster no está disponible debido a una interrupción que afecta a una zona de disponibilidad completa. En este caso, la forma de poner en línea una nueva instancia principal depende de si el clúster utiliza una configuración Multi-AZ:

- Si el clúster aprovisionado o de Aurora Serverless v2 contiene alguna instancia de lector en otras AZ, Aurora utiliza el mecanismo de conmutación por error para promover una de esas instancias de lector como la nueva instancia principal.
- Si el clúster aprovisionado o de Aurora Serverless v2 solo contiene una única instancia de base de datos, o si la instancia principal y todas las instancias de lector están en la misma zona de disponibilidad, asegúrese de crear manualmente una o más instancias de base de datos nuevas en otra zona de disponibilidad.
- Si el clúster utiliza Aurora Serverless v1, Aurora crea automáticamente una nueva instancia de base de datos en otra zona de disponibilidad. Sin embargo, este proceso implica un reemplazo de anfitrión y, por lo tanto, toma más tiempo que una conmutación por error.

> **Note**
>
> Amazon Aurora también admite la replicación con una base de datos de MySQL externa o una instancia de base de datos de RDS MySQL. Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Alta disponibilidad con Amazon RDS Proxy

Con RDS Proxy, puede crear aplicaciones que puedan tolerar de forma transparente los errores de las bases de datos sin necesidad de escribir código complejo de gestión de errores. El proxy dirige automáticamente el tráfico a una nueva instancia de base de datos y conserva las conexiones de la aplicación. También evita las cachés del Sistema de nombres de dominio (DNS) para reducir los tiempos de conmutación por error hasta en un 66 % para las bases de datos Aurora Multi-AZ. Para obtener más información, consulte Amazon RDS Proxy para Aurora.

Replicación con Amazon Aurora

Existen varias opciones de replicación con Aurora. Cada clúster de base de datos Aurora tiene replicación integrada entre varias instancias de base de datos en el mismo clúster. También puede configurar la replicación con el clúster Aurora como origen o destino. Al replicar datos dentro o fuera de un clúster Aurora, puede elegir entre características integradas como bases de datos Aurora globales o los mecanismos de replicación tradicionales para los motores de base de datos MySQL o PostgreSQL. Puede elegir las opciones adecuadas en función de cuál proporciona la combinación correcta de alta disponibilidad, conveniencia y rendimiento para sus necesidades. En las siguientes secciones se explica cómo y cuándo elegir cada técnica.

Temas

- Réplicas de Aurora
- Replicación con Aurora MySQL
- Replicación con Aurora PostgreSQL

Réplicas de Aurora

Cuando crea una segunda, tercera, etc. instancia de base de datos en un clúster de base de datos Aurora aprovisionada, Aurora configura automáticamente la replicación desde la instancia de base de datos de escritor a todas las demás instancias de base de datos. Estas otras instancias de base de datos son de solo lectura y se conocen como réplicas Aurora. También nos referimos a ellas como instancias de lector al analizar las formas en que puede combinar instancias de base de datos de escritor y lector dentro de un clúster.

Las réplicas Aurora tienen dos propósitos principales. Puede emitirles consultas para escalar las operaciones de lectura de la aplicación. Normalmente lo hace conectándose al punto de enlace del lector del clúster. De esta forma, Aurora puede distribuir la carga de conexiones de solo lectura entre

tantas réplicas Aurora como tenga en el clúster. Las réplicas Aurora también ayudan a aumentar la disponibilidad. Si la instancia de escritor de un clúster deja de estar disponible, Aurora promociona automáticamente una de las instancias de lector para que tome su lugar como el nuevo escritor.

Un clúster de base de datos Aurora puede contener hasta réplicas 15 Aurora. Se pueden distribuir réplicas de Aurora entre las distintas zonas de disponibilidad que abarca un clúster de base de datos dentro de una región de AWS.

Los datos del clúster de base de datos tienen sus propias características de alta disponibilidad y confiabilidad, independientemente de las instancias de base de datos del clúster. Si no está familiarizado con las funciones Aurora de almacenamiento, consulte Información general del almacenamiento de Amazon Aurora. El volumen del clúster de base de datos consta físicamente de varias copias de los datos del clúster de base de datos. La instancia principal y las réplicas Aurora del clúster de base de datos ven los datos del volumen del clúster como un único volumen lógico.

Como resultado, todas las réplicas de Aurora devuelven los mismos datos para los resultados de las consultas con un retraso de réplica mínimo. Este retraso suele ser inferior a 100 milisegundos después de que la instancia principal haya escrito una actualización. El retardo de la réplica varía en función de la velocidad de cambio de la base de datos. Es decir, durante los periodos en los que se produce una gran cantidad de operaciones de escritura en la base de datos, puede registrarse un aumento del retardo de la réplica.

> **Note**
>
> La réplica de Aurora se reinicia automáticamente cuando pierde la comunicación con la instancia de base de datos del escritor durante más de 60 segundos en las siguientes versiones de Aurora PostgreSQL:
>
> - Versión 14.6 y anteriores
> - Versión 13.9 y anteriores
> - Versión 12.13 y anteriores
> - Todas las versiones de Aurora PostgreSQL 11
>
> Con la característica de disponibilidad de lectura, las réplicas de Aurora no se reinician automáticamente. Para obtener más información sobre la característica de disponibilidad de lectura y las versiones en las que se aplica, consulte Mejora de la disponibilidad de lectura de las réplicas de Aurora.

Las réplicas de Aurora funcionan bien para el escalado de lectura porque están totalmente dedicadas a las operaciones de lectura en el volumen del clúster. Las operaciones de escritura se administran en la instancia principal. Como el volumen del clúster se comparte entre todas las instancias de base de datos del clúster de base de datos, se requiere un trabajo adicional mínimo para replicar una copia de los datos para cada réplica de Aurora.

Para incrementar la disponibilidad, puede usar las réplicas de Aurora como objetivos de conmutación por error. Es decir, que si la instancia principal da error, una réplica de Aurora se convierte en la instancia principal. En este proceso se produce una breve interrupción durante la cual las solicitudes de escritura y lectura realizadas a la instancia principal generan errores con una excepción.

Promover una réplica de Aurora por conmutación por error es mucho más rápido que volver a crear la instancia principal. Si el clúster de la base de datos de Aurora no incluye ninguna réplica de Aurora, el clúster de la base de datos no estará disponible mientras la instancia de base de datos se recupera del error.

Cuando se produce la conmutación por error, algunas de las réplicas de Aurora podrían reiniciarse, según la versión del motor de base de datos. Por ejemplo, en Aurora MySQL, Aurora reinicia solo la instancia de base de datos del escritor y el destino de conmutación por error durante una conmutación por error. Para obtener más información sobre el comportamiento de reinicio de las diferentes versiones del motor de base de datos de Aurora, consulte Reinicio de un clúster de base de datos de Amazon Aurora o de una instancia de base de datos de Amazon Aurora. Para obtener información sobre lo que ocurre con las cachés de páginas al reiniciar o realizar una conmutación por error, consulte Caché de páginas que puede sobrevivir.

Para escenarios de alta disponibilidad, le recomendamos que cree una o más réplicas de Aurora. Dichas réplicas deberían ser de la misma clase de instancia de base de datos que la instancia principal y de zonas de disponibilidad distintas para el clúster de base de datos Aurora. Para obtener más información sobre las réplicas de Aurora como destinos de conmutación por error, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

No puede crear una réplica de Aurora cifrada para un clúster de base de datos de Aurora sin cifrar. No puede crear una réplica de Aurora sin cifrar para un clúster de base de datos de Aurora cifrado.

> **Tip**
>
> Puede utilizar Réplicas Aurora dentro de un clúster Aurora como única forma de replicación para mantener los datos altamente disponibles. También puede combinar la Aurora

replicación integrada con los otros tipos de replicación. Hacerlo puede ayudar a proporcionar un nivel adicional de alta disponibilidad y distribución geográfica de sus datos.

Para obtener información detallada acerca de la forma de crear una réplica de Aurora, consulte Adición de réplicas de Aurora a un clúster de base de datos.

Replicación con Aurora MySQL

Además de las réplicas de Aurora, dispone de las siguientes opciones para replicar con Aurora MySQL:

- Los clústeres de base de datos de Aurora MySQL en diferentes regiones de AWS.
  - Puede replicar datos en varias regiones mediante una base de datos Aurora global. Para obtener más información, consulte Alta disponibilidad en todas las regiones de AWS con bases de datos de Aurora globales.
  - Puede crear una réplica de lectura de Aurora de un clúster de base de datos MySQL de Aurora en una región de AWS diferente, mediante el uso de la replicación de registros binarios (binlog) de MySQL. Cada clúster puede tener hasta cinco réplicas de lectura creadas de esta manera, cada una en una región diferente.
- Dos clústeres de base de datos de Aurora MySQL en la misma región de mediante la utilización de la reproducción del registro binario (binlog) de MySQL.
- Una instancia de base de datos RDS for MySQL como origen de los datos y un clúster d e base de datos de Aurora MySQL, al crear una réplica de lectura de Aurora de una instancia de base de datos RDS for MySQL. Normalmente, este método se usa para la migración de Aurora MySQL y no para una replicación continua.

Para obtener más información sobre cómo replicar con Aurora MySQL, consulte Replicación con Amazon Aurora MySQL.

Replicación con Aurora PostgreSQL

Además de las réplicas de Aurora, dispone de las siguientes opciones para replicar con Aurora PostgreSQL:

- Un clúster de base de datos principal de Aurora en una región y hasta 10 clústeres de base de datos secundarios de solo lectura en diferentes regiones mediante una base de datos global de

Aurora. Aurora PostgreSQL no admite réplicas de Aurora entre regiones. Sin embargo, se puede usar la base de datos global de Aurora para escalar las capacidades de lectura de su clúster de Aurora PostgreSQL DB a más de una región de AWS y cumplir con los objetivos de disponibilidad. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

- Dos clústeres de base de datos Aurora PostgreSQL en la misma región, mediante el uso de la característica de replicación lógica de PostgreSQL.
- Una instancia de base de datos RDS for PostgreSQL como origen de los datos y un clúster de base de datos Aurora PostgreSQL, creando una réplica de lectura Aurora de una instancia de base de datos RDS for PostgreSQL. Por lo general, se usa este enfoque para la migración a Aurora PostgreSQL, más que para la replicación continua.

Para obtener más información sobre cómo replicar con Aurora PostgreSQL, consulte Replicación con Amazon Aurora PostgreSQL.

Facturación de instancia de base de datos para Aurora

Las instancias aprovisionadas por Amazon RDS en un clúster de Amazon Aurora se facturan en función de los siguientes componentes:

- Horas de instancia de base de datos (por hora): en función de la clase de instancia de base de datos (por ejemplo, db.t2.small o db.m4.large). Los precios se muestran por hora, pero las facturas se ajustan hasta el segundo y muestran las horas en formato decimal. El uso de RDS se factura por incrementos de un segundo, con un mínimo de 10 minutos. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.
- Almacenamiento (por GiB al mes): la capacidad de almacenamiento que ha aprovisionado para su instancia de base de datos. Si escala la capacidad de almacenamiento aprovisionada durante el mes, la factura se prorratea. Para obtener más información, consulte Almacenamiento de Amazon Aurora.
- Solicitudes de entrada/salida (E/S) (por millón de solicitudes): número total de solicitudes de E/S de almacenamiento realizadas en un ciclo de facturación, solo la configuración de clúster de base de datos de Aurora Standard.

  Para obtener más información acerca del uso de claves KMS en Amazon Aurora, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.
- Almacenamiento de copias de seguridad (por GiB al mes): el almacenamiento de copias de seguridad es el almacenamiento asociado a copias de seguridad de base de datos automatizadas

y cualquier instantánea de base de datos activa que haya realizado. Aumentar el período de retención de copia de seguridad u obtener instantáneas de base de datos adicionales aumenta el almacenamiento de copias de seguridad consumido por su base de datos. La facturación por segundo no se aplica al almacenamiento de copia de seguridad (medido en GB/mes).

Para obtener más información, consulte Copias de seguridad y restauración de un clúster de base de datos de Amazon Aurora.

- Transferencia de datos (por GB): las transferencias de datos de entrada y de salida de su instancia de base de datos, desde y hacia Internet y otras regiones de AWS. Para ver ejemplos útiles, consulte la entrada del blog de AWS Exploring Data Transfer Costs for AWS Managed Databases.

Amazon RDS proporciona las siguientes opciones de compra para que pueda optimizar los costos en función de sus necesidades:

- On-Demand Instances (Instancias bajo demanda): pague por las horas de instancia de base de datos que use. Los precios se muestran por hora, pero las facturas se ajustan hasta el segundo y muestran las horas en formato decimal. El uso de RDS ahora se factura por incrementos de un segundo, con un mínimo de 10 minutos.
- Reserved Instances (Instancias reservadas): reserve una instancia de base de datos durante un plazo de uno a tres años y obtenga descuentos importantes en comparación con los precios de instancias de base de datos bajo demanda. Cuando use instancias reservadas, podrá lanzar, eliminar, iniciar o detener varias instancias dentro de una misma hora y obtener el beneficio de instancia reservada para todas las instancias.
- Aurora Serverless v2: Aurora Serverless v2 proporciona capacidad bajo demanda cuando la unidad de facturación es horas de unidad de capacidad de Aurora (ACU) en lugar de horas de instancia de base de datos. La capacidad de Aurora Serverless v2 aumenta y disminuye dentro del rango que especifique, en función de la carga de la base de datos. Puede configurar un clúster donde toda la capacidad sea Aurora Serverless v2. O puede configurar una combinación de Aurora Serverless v2 e instancias aprovisionadas bajo demanda o reservadas. Para obtener más información sobre cómo funcionan las ACU de Aurora Serverless v2, consulte Cómo funciona Aurora Serverless v2.
- Base de datos ilimitada de Aurora PostgreSQL: Base de datos ilimitada de Aurora PostgreSQL es una capacidad de escalado horizontal automatizada que se amplía más allá del rendimiento de escritura y los límites de almacenamiento de una sola instancia de base de datos. Base de datos ilimitada distribuye la carga de trabajo entre varias instancias de escritura de Aurora y, al mismo tiempo, mantiene la facilidad de operación como una única base de datos. Base de datos ilimitada

proporciona capacidad bajo demanda cuando la unidad de facturación son las horas en unidades de capacidad de Aurora (ACU) en un grupo de particiones de bases de datos. Para obtener más información sobre cómo funcionan las ACU de Base de datos ilimitada, consulte Creación de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL.

Para obtener información acerca de los precios de Aurora, consulte la página de precios de Aurora.

Temas

- Instancias de base de datos bajo demanda para Aurora
- Instancias de base de datos reservadas para Amazon Aurora

Instancias de base de datos bajo demanda para Aurora

Las instancias de base de datos bajo demanda de Amazon RDS se facturan en función de la clase de instancia de base de datos (por ejemplo, db.t3.small o db.m5.large). Para obtener información acerca de los precios de Amazon RDS, consulte la página del producto de Amazon RDS.

La facturación de una instancia de base de datos comienza en cuanto la instancia está disponible. Los precios se muestran por hora, pero las facturas se ajustan hasta el segundo y muestran las horas en formato decimal. El uso de Amazon RDS se factura por incrementos de un segundo, con un mínimo de 10 minutos. En caso de un cambio de configuración facturable, como el escalado informático o la capacidad de almacenamiento, se le cobrará un mínimo de 10 minutos. La facturación continúa hasta que se termina la instancia de base de datos, lo que tiene lugar cuando se elimina la instancia de base de datos o produce un error.

Si ya no desea que se le cobre por su instancia de base de datos, debe detenerla o eliminarla para evitar que se le cobren horas de instancia de base de datos adicionales. Para obtener más información acerca de los estados de instancias de base de datos que se le cobran, consulte Visualización del .

Instancias de base de datos detenidas

Mientras la instancia de base de datos está detenida, se le cobra el almacenamiento provisionado, incluidas las IOPS provisionadas. También se le cobra el almacenamiento de copias de seguridad, incluido el almacenamiento de las instantáneas manuales y las copias de seguridad automatizadas en el periodo de retención especificado. No se le cobrarán las horas de instancia de base de datos.

Instancias de base de datos Multi-AZ

Una configuración multi-AZ mejora la durabilidad y disponibilidad de los datos al aprovisionar y mantener automáticamente una réplica en espera sincrónica dentro de una zona de disponibilidad diferente. Debido a los recursos adicionales y al aumento de la disponibilidad, las implementaciones multi-AZ tienen un precio más alto que las implementaciones en AZ única y pueden costar aproximadamente el doble debido a la instancia de reserva adicional y a los recursos asociados.

Tenga en cuenta los siguientes detalles importantes acerca de los precios de multi-AZ:

- Costos informáticos: se facturan por hora de instancia de base de datos para instancias principales y en espera.
- Costos de almacenamiento: se cobran por GB al mes por el almacenamiento aprovisionado para las instancias principales y en espera.

- Costos de transferencia de datos: la replicación entre las instancias principales y en espera está incluida en el costo, pero es posible que se apliquen otros cargos por transferencia de datos en función del uso.

Para estimar con precisión los costos mensuales en función del caso de uso específico y Región de AWS, puede usar Calculadora de precios de AWS. Esta herramienta le permite ingresar los detalles de configuración y proporciona un desglose completo de los costos.

> **Note**
>
> Los precios están sujetos a cambios. Consulte la página de precios de Amazon RDS para obtener la información más actualizada.

Instancias de base de datos reservadas para Amazon Aurora

Puede reservar una instancia de base de datos durante un periodo de un año o de tres años mediante instancias de base de datos reservadas. Las instancias de base de datos reservadas ofrecen un descuento importante en comparación con los precios de las instancias de base de datos bajo demanda. Las instancias de base de datos reservadas no son instancias físicas sino más bien un descuento de facturación que se aplica al uso de determinadas instancias de base de datos bajo demanda en su cuenta. Los descuentos están vinculados al tipo de instancia y a la Región de AWS.

El proceso general de trabajo con instancias de base de datos reservadas es el siguiente: en primer lugar, obtener información sobre las ofertas de instancias de base de datos reservadas; en segundo lugar, comprar una oferta y, por último, obtener información sobre las instancias de base de datos reservadas.

Para obtener información sobre la compra de instancias de base de datos reservadas y sobre la consulta de la facturación de dichas instancias, consulte las siguientes secciones.

- Compra de instancias de base de datos reservadas para Amazon Aurora
- Visualización de la facturación de las instancias de bases de datos reservadas para Amazon Aurora

Información general sobre instancias de base de datos reservadas

Cuando adquiere una instancia de base de datos reservada en Amazon RDS, adquiere un compromiso para obtener una tarifa con descuento en un tipo de instancia de base de datos específico durante el periodo de duración de la instancia de base de datos reservada. Para usar una instancia de base de datos reservada de Amazon RDS, debe crear una instancia de base de datos nueva, tal como haría para una instancia bajo demanda.

La nueva instancia de base de datos que cree deberá tener las mismas especificaciones que la instancia de base de datos reservada, es decir:

- Región de AWS
- Motor de base de datos (no es necesario que el número de versión del motor de base de datos coincida).
- Tipo de instancia de base de datos

Si las especificaciones de la nueva instancia de base de datos nueva coinciden con una instancia reservada existente para su cuenta, se le facturará con la tarifa con descuento ofrecida para la instancia reservada. De lo contrario, la instancia de base de datos se factura con una tarifa bajo demanda.

Puede modificar una instancia de base de datos que utilice como instancia de base de datos reservada. Si la modificación está dentro de las especificaciones de la instancia de base de datos reservada, parte o todo el descuento seguirá siendo aplicable a la instancia de base de datos modificada. Si la modificación está fuera de las especificaciones, como cambiar la clase de instancia, el descuento ya no se aplica. Para obtener más información, consulte Flexibilidad del tamaño de las instancias de base de datos reservadas.

Temas

- Tipos de ofertas
- Flexibilidad de la configuración del clúster de base de datos de Aurora
- Flexibilidad del tamaño de las instancias de base de datos reservadas
- Ejemplos de facturación de instancias de base de datos reservadas de Aurora
- Eliminación de una instancia de base de datos reservada

Para obtener más información acerca de las instancias de base de datos reservadas, incluidos los precios, consulte Instancias reservadas de Amazon RDS.

Tipos de ofertas

Las instancias de base de datos reservadas están disponibles en tres variedades: sin pago inicial, pago inicial parcial y pago inicial total, lo cual le permite optimizar sus costos de Amazon RDS en función del uso previsto.

> **Note**
>
> No todas las clases de instancias de RDS admiten todos los tipos de ofertas de instancias reservadas. Por ejemplo, es posible que algunas clases de instancias no ofrezcan la opción Sin pago inicial. Para confirmar la disponibilidad, revise las ofertas de instancias reservadas en Consola de administración de AWS o use el comando `describe-reserved-db-instances-offerings` de la AWS CLI.

Sin pago inicial

Esta opción proporciona acceso a una instancia de base de datos reservada sin que haya que hacer un pago inicial. Su instancia de base de datos reservada sin pago inicial le cobra una tarifa por hora con descuento por cada hora dentro del plazo, independientemente del uso. No es necesario realizar ningún pago inicial. Esta opción solo está disponible en la modalidad de reserva de un año.

Pago inicial parcial

Esta opción exige que parte de la instancia de base de datos reservada se pague por adelantado. Las horas restantes del plazo se cobran a una tarifa por hora con descuento, independientemente del uso que haga. Esta opción sustituye la anterior opción de utilización intensa.

Pago inicial total

Se realiza un pago total al comienzo del plazo, y no se aplicará ningún otro costo el resto del plazo, independientemente del número de horas de uso.

Si está utilizando la facturación unificada, todas las cuentas de la organización se tratan como una sola. Esto quiere decir que todas las cuentas de la organización pueden beneficiarse del precio por hora reducido de las instancias de base de datos reservadas adquiridas por otra cuenta. Para obtener más información sobre la facturación unificada, consulte Instancias de base de datos reservadas de Amazon RDS en la Guía del usuario de Administración de costos y facturación de AWS.

Flexibilidad de la configuración del clúster de base de datos de Aurora

Puede utilizar las instancias de base de datos reservadas de Aurora con ambas configuraciones del clúster de base de datos:

- Aurora I/O-Optimized: solo paga por el uso y el almacenamiento de sus clústeres de bases de datos, sin cargos adicionales por las operaciones de E/S de lectura y escritura.
- Aurora Standard: además del uso y el almacenamiento de sus clústeres de bases de datos, también paga una tarifa estándar por cada millón de solicitudes de operaciones de E/S.

Aurora contabiliza automáticamente la diferencia de precio entre estas configuraciones. Aurora I/O-Optimizedconsume un 30 % más de unidades normalizadas por hora que Aurora Standard.

Para obtener más información acerca de las configuraciones de almacenamiento en clústeres de base de datos de Aurora, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora. Para obtener más información acerca de los precios de las configuraciones de almacenamiento en clústeres de Aurora, consulte Precios de Amazon Aurora.

Flexibilidad del tamaño de las instancias de base de datos reservadas

Al comprar una instancia de base de datos reservada, una de las cosas que especifica es la clase de instancia, por ejemplo, db.r5.large. Para obtener más información sobre las clases de instancias de bases de datos, consulte Clases de instancia de base de datos de Amazon Aurora.

Si tiene una instancia de base de datos y debe escalarla para aumentar la capacidad, la instancia de base de datos reservada se aplica automáticamente a la instancia de base de datos escalada. Es decir que las instancias de base de datos reservadas se aplican automáticamente entre todos los tamaños de clase de instancia de base de datos. Las instancias de base de datos reservadas con flexibilidad de tamaño están disponibles para las instancias de base de datos de la misma Región de AWS y motor de base de datos. Las instancias de base de datos reservadas con flexibilidad de tamaño solo se pueden escalar en su tipo de clase de instancia. Por ejemplo, una instancia de base de datos reservada para una db.r6i.large se puede aplicar en una db.r6i.xlarge, pero no en una db.r6id.large ni db.r7g.large, ya que db.r6id.large y db.r7g.large son clases distintas de instancias.

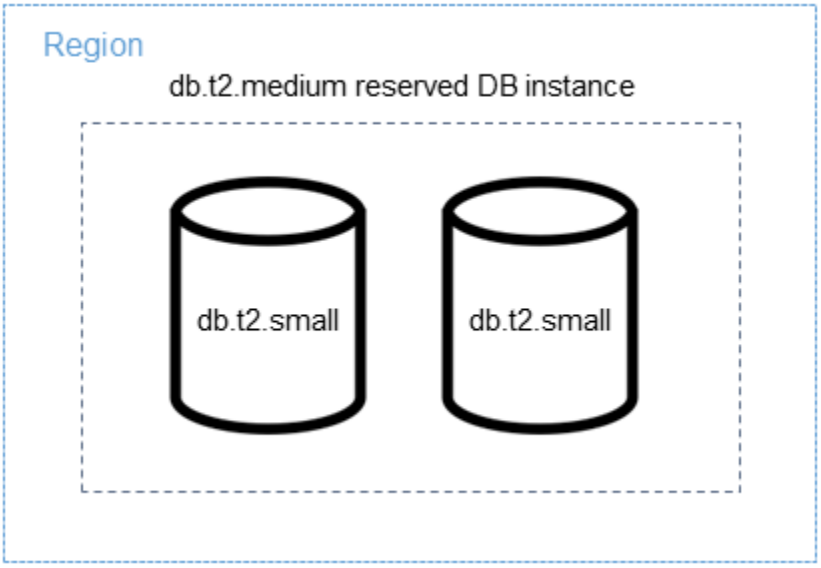
Las instancias de base de datos reservadas con flexibilidad de tamaño están disponibles para los siguientes motores de base de datos de Aurora:

- Aurora MySQL
- Aurora PostgreSQL

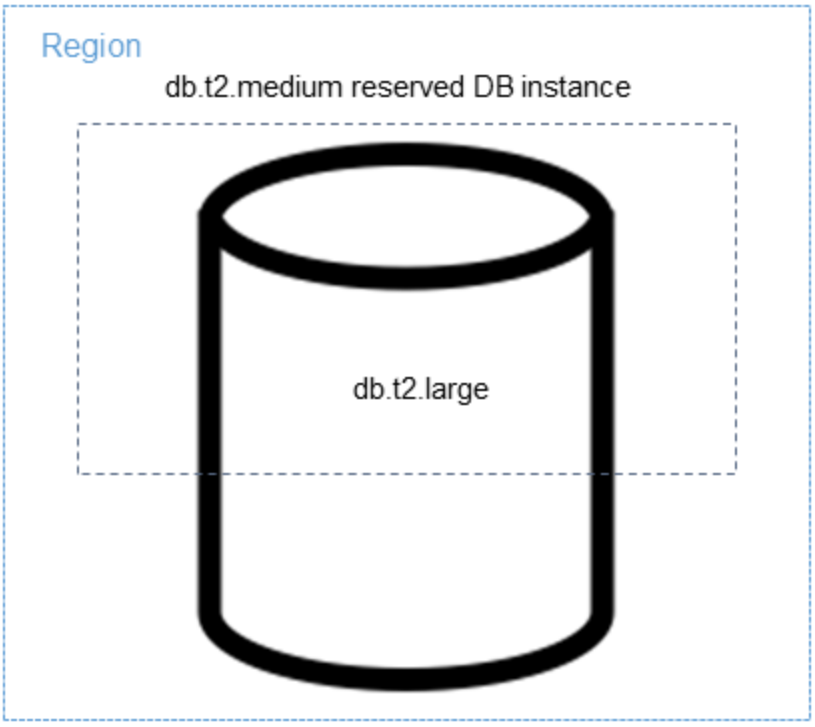
Puede comparar el uso de diferentes tamaños de instancias de base de datos reservadas utilizando unidades normalizadas por hora. Por ejemplo, una unidad de uso en dos instancias de base de datos db.r3.large equivale a ocho unidades de uso normalizadas por hora en una db.r3.small. En la tabla siguiente se muestra el número de unidades normalizadas por hora por cada tamaño de instancia de base de datos.

Tamaño de instancia	Unidades normalizadas por hora para una instancia de base de datos, Aurora Standard	Unidades normalizadas por hora para una instancia de base de datos, Aurora I/O-Optimized	Unidades normalizadas por hora para tres instancias de base de datos (escritor y dos lectores), Aurora Standard	Unidades normalizadas por hora para tres instancias de base de datos (escritor y dos lectores), Aurora I/O-Optimized
small	1	1.3	3	3.9
medium	2	2.6	6	7.8
large	4	5.2	12	15.6
xlarge	8	10.4	24	31.2
2xlarge	16	20.8	48	62.4
4xlarge	32	41.6	96	124.8
8xlarge	64	83.2	192	249.6
12xlarge	96	124.8	288	374.4
16xlarge	128	166.4	384	499.2
24xlarge	192	249.6	576	748.8
32xlarge	256	332.8	768	998.4

Por ejemplo, suponga que adquiere una instancia de base de datos reservada `db.t2.medium` y tiene dos instancias de base de datos `db.t2.small` en ejecución en su cuenta en la misma Región de AWS. En este caso, el beneficio de facturación se aplica en su totalidad a las dos instancias.



De forma alternativa, si tiene una instancia `db.t2.large` en ejecución en su cuenta en la misma Región de AWS, el beneficio de facturación se aplica al 50 % del uso de la instancia de base de datos.



> **Note**
>
> Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.

Ejemplos de facturación de instancias de base de datos reservadas de Aurora

Los siguientes ejemplos ilustran los precios de las instancias de base de datos reservadas para los clústeres de base de datos de Aurora que utilizan las configuraciones del clúster de base de datos de Aurora Standard y de Aurora I/O-Optimized.

Ejemplo de uso de Aurora Standard

El precio de una instancia de base de datos reservada no incluye descuentos en los costos asociados al almacenamiento, las copias de seguridad y la E/S. En el ejemplo siguiente, se muestra el costo total mensual de una instancia de base de datos reservada:

- Una clase de instancia de base de datos reservada db.r5.large Single-AZ de Aurora MySQL en Este de EE. UU. (Norte de Virginia) con un costo de 0,19 USD por hora o de 138,70 USD al mes
- Almacenamiento de Aurora con un costo de 0,10 USD por GiB al mes (asuma 45,60 USD al mes para este ejemplo)
- E/S de Aurora con un costo de 0,20 USD por 1 millón de solicitudes (asuma 20 USD al mes para este ejemplo)
- Almacenamiento de copia de seguridad de Aurora con un costo de 0,021 USD por GiB al mes (asuma 30 USD al mes para este ejemplo)

Sume todas estas opciones (138,70 USD + 45,60 USD + 20 USD + 30 USD) a la instancia de base de datos reservada y el costo total mensual es de 234.30 USD.

Si elige una instancia de base de datos bajo demanda en lugar de una instancia de base de datos reservada, una clase de instancia de base de datos reservada db.r5.large Single-AZ de Aurora MySQL en Este de EE. UU. (Norte de Virginia) cuesta 0,29 USD por hora o 217,50 USD al mes. Así que para una instancia de base de datos bajo demanda, sume todas estas opciones (217,50 USD + 45,60 USD + 20 USD + 30 USD) y el costo total mensual es de 313,10 USD. Ahorra casi 79 USD al mes al utilizar la instancia de base de datos reservada.

Ejemplo de uso de un clúster de base de datos de Aurora Standard con dos instancias de lector

Para usar instancias reservadas para los clústeres de base de datos de Aurora, tan solo debe comprar una instancia reservada para cada instancia de base de datos del clúster.

Si ampliamos el primer ejemplo, tiene un clúster de base de datos de Aurora MySQL con una instancia de base de datos de escritor y dos réplicas de Aurora, lo que hace un total de tres instancias de base de datos en el clúster. Las dos réplicas de Aurora no incurren en cargos adicionales de almacenamiento ni copias de seguridad. Si compra tres instancias de base de datos db.r5.large reservadas de Aurora MySQL, el costo será de 234,30 USD (para la instancia de base de datos de escritor) más 2 x (138,70 USD + 20 USD de E/S por réplica de Aurora), lo que supone un total de 551,70 USD al mes.

El costo correspondiente bajo demanda de un clúster de base de datos de Aurora MySQL con una instancia de base de datos de escritor y dos réplicas de Aurora es de 313,10 USD + 2* (217,50 USD + 20 USD de E/S por instancia), lo que supone un total de 788,10 USD al mes. Ahorra casi 236,40 USD al mes al utilizar las instancias de base de datos reservadas.

Ejemplo de uso de Aurora I/O-Optimized

Puede reutilizar las instancias de base de datos reservadas de Aurora Standard existentes con Aurora I/O-Optimized. Para aprovechar al máximo las ventajas de los descuentos de instancias reservadas con Aurora I/O-Optimized, puede comprar un 30 % más de instancias reservadas similares a las instancias reservadas actuales.

La siguiente tabla muestra ejemplos sobre cómo estimar las instancias reservadas adicionales cuando se utiliza Aurora I/O-Optimized. Si las instancias reservadas requeridas son una fracción. Puede aprovechar la flexibilidad de tamaño disponible con las instancias reservadas para obtener un número entero. En estos ejemplos, «actual» hace referencia a las instancias reservadas de Aurora Standard que tiene ahora. Las instancias reservadas adicionales son la cantidad de instancias reservadas de Aurora Standard que debe comprar para mantener los descuentos actuales de instancias reservadas al utilizar Aurora I/O-Optimized.

Clase de instancia de base de datos	Instancias reservadas de Aurora Standard actuales	Se requieren instancias reservadas para Aurora I/O-Optimized	Se requieren instancias reservadas adicionales	Se requieren instancias reservadas adicionales, utilizando la flexibilidad de tamaño
db.r6g.large	10	10 x 1,3 = 13	3 x db.r6g.large	3 x db.r6g.large
db.r6g.4xlarge	20	20 x 1,3 = 26	6 x db.r6g.4xlarge	6 x db.r6g.4xlarge
db.r6g.12xlarge	5	5 x 1,3 = 6,5	1,5 x db.r6g.12xlarge	Uno de cada db.r6g.12xlarge, r6g.4xlarge y r6g.2xlarge (0,5 x db.r6g.12xlarge = 1 x db.r6g.4xlarge + 1 x db.r6g.2xlarge)
db.r6i.24xlarge	15	15 x 1,3 = 19,5	4,5 x db.r6i.24xlarge	4 x db.r6i.24xlarge + 1 x db.r6i.12xlarge (0,5 x db.r6i.24xlarge = 1 x db.r6i.12xlarge)

Ejemplo de uso de un clúster de base de datos de Aurora I/O-Optimized con dos instancias de lector

Tiene un clúster de base de datos de Aurora MySQL con una instancia de base de datos de escritor y dos réplicas de Aurora, lo que hace un total de tres instancias de base de datos en el clúster. Utilizan la configuración del clúster de base de datos de Aurora I/O-Optimized. Para usar instancias de base de datos reservadas para este clúster, tendrá que comprar cuatro instancias de base de

datos reservadas de la misma clase de instancia de base de datos. Tres instancias de base de datos que utilizan Aurora I/O-Optimized consumen 3,9 unidades normalizadas por hora, en comparación con las tres unidades normalizadas por hora que utilizan tres instancias de base de datos con Aurora Standard. Sin embargo, se ahorra los costos mensuales de E/S de cada instancia de base de datos.

> **Note**
>
> Los precios que aparecen aquí son ejemplos y podrían no coincidir con los precios reales. Para obtener información acerca de los precios de Aurora, consulte Precios de Amazon Aurora.

Eliminación de una instancia de base de datos reservada

Los términos de una instancia de base de datos reservada implican un compromiso de un año o de tres años. No tiene autorización para cancelar una instancia de base de datos reservada. Sin embargo, puede eliminar una instancia de base de datos que tenga un descuento de instancia de base de datos reservada. El proceso para eliminar una instancia de base de datos con un descuento de instancia de base de datos reservada es el mismo que para cualquier otra instancia de base de datos.

Los costos iniciales se facturarán independientemente de si utiliza los recursos.

Si elimina una instancia de base de datos con un descuento de instancia de base de datos reservada, puede lanzar otra instancia de base de datos con especificaciones compatibles. En este caso, sigue disfrutando de la tarifa de descuento mientras dure la reserva (de uno o tres años).

Compra de instancias de base de datos reservadas para Amazon Aurora

Puede utilizar la Consola de administración de AWS, la AWS CLI y la API de RDS para trabajar con instancias de base de datos reservadas.

Consola

Puede utilizar la Consola de administración de AWS para trabajar con instancias de base de datos reservadas, tal como se muestra en los siguientes procedimientos.

Para obtener precios e información sobre ofertas de instancias de base de datos reservadas disponibles

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Reserved instances (Instancias reservadas).
3. Elija Purchase Reserved DB Instance.
4. En Product description (Descripción de producto), elija el motor de base de datos y el tipo de licencia.
5. En DB instance class (Clase de instancia de base de datos), elija la clase de instancia de base de datos.
6. En Opción de implementación, elija si quiere una implementación de instancias single-AZ o multi-AZ.

> **Note**
>
> Las instancias reservadas de Amazon Aurora siempre tienen la opción de implementación definida en Instancia de base de datos Single-AZ. Sin embargo, al crear un clúster de base de datos de Aurora, la opción de implementación predeterminada es Crear una réplica o lector de Aurora en una AZ diferente (multi-AZ).
> Debe comprar una instancia de base de datos reservada para cada instancia que planee utilizar, incluidas las réplicas de Aurora. Por lo tanto, para las implementaciones Multi-AZ en Aurora, debe comprar instancias de base de datos reservadas adicionales.

7. En Periodo, elija el tiempo durante el cual desea que se reserve la instancia de base de datos.
8. En Offering type (Tipo de oferta), elija el tipo de oferta.

   Después de seleccionar el tipo de oferta, podrá ver la información sobre los precios.

> **Important**
>
> Elija Cancel (Cancelar) para evitar comprar la instancia de base de datos reservada y generar cargos.

Después de recibir la información sobre las ofertas disponibles de instancias de base de datos reservadas, podrá utilizar dicha información para adquirir una oferta, tal como se explica a continuación.

Para comprar una instancia de base de datos reservada

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Reserved instances (Instancias reservadas).
3. **Important**

   Antes de continuar, compruebe que se encuentra en la Región de AWS correcta. Las instancias de base de datos reservadas son específicas de cada región y no se pueden transferir entre regiones. Compruebe el selector de región en la esquina superior derecha de la consola para asegurarse de que está comprando la instancia reservada en la región deseada.
4. Elija Purchase Reserved DB Instance (Comprar instancia de base de datos reservada).
5. En Product description (Descripción de producto), elija el motor de base de datos y el tipo de licencia.
6. En DB instance class (Clase de instancia de base de datos), elija la clase de instancia de base de datos.
7. En Implementación multi-AZ, elija si quiere una implementación de instancias de base de datos sinle-AZ o multi-AZ.

   Note

   Las instancias reservadas de Amazon Aurora siempre tienen la opción de implementación definida en Instancia de base de datos Single-AZ. Si crea un clúster de base de datos de Amazon Aurora a partir de su instancia de base de datos reservada, el clúster de base de datos se crea automáticamente como multi-AZ. Asegúrese de comprar una instancia de base de datos reservada para cada instancia de base de datos que tenga pensado utilizar, incluidas las réplicas de Aurora.
8. En Term, elija el tiempo durante el cual desea que se reserve la instancia de base de datos.
9. En Offering type (Tipo de oferta), elija el tipo de oferta.

Después de seleccionar el tipo de oferta, podrá ver la información sobre los precios.
Consola de compra de instancia de base de datos reservada

10. (Opcional) Puede asignar su propio identificador a las instancias de base de datos reservadas que adquiera para poder realizar un seguimiento de estas. En Reserved Id, escriba un identificador para la instancia de base de datos reservada.
11. Elija Enviar.

    La instancia de base de datos reservada se compra y, a continuación, se muestra en la lista de Reserved instances (Instancias reservadas).

Después de adquirir las instancias de base de datos reservadas, podrá obtener información sobre las instancias de base de datos reservadas, tal como se muestra a continuación.

Para obtener información sobre instancias de base de datos reservadas para su cuenta de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel Navigation (Navegación), elija Reserved instances (Instancias reservadas).

   Aparecerán las instancias de base de datos reservadas de la cuenta. Para ver información detallada sobre la instancia de base de datos reservada particular, elija dicha instancia de la lista. Entonces, podrá ver información detallada sobre esa instancia en el panel de detalles ubicado en la parte inferior de la consola.

AWS CLI

Puede utilizar la AWS CLI para trabajar con instancias de base de datos reservadas, tal como se muestra en los siguientes ejemplos.

Example de obtener ofertas de instancias de base de datos reservadas disponibles

Para obtener información sobre las ofertas disponibles de instancias de base de datos reservadas, llame al comando AWS CLI de la `describe-reserved-db-instances-offerings`.

```
aws rds describe-reserved-db-instances-offerings
```

Esta llamada devuelve un resultado similar al siguiente:

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

Después de recibir la información sobre las ofertas disponibles de instancias de base de datos reservadas, podrá utilizar dicha información para adquirir una oferta.

Para adquirir una instancia de base de datos reservada, use el comando AWS CLI de la `purchase-reserved-db-instances-offering` con los siguientes parámetros:

- `--reserved-db-instances-offering-id` – El ID de la oferta que desea adquirir. Consulte el ejemplo anterior para obtener el ID de la oferta.
- `--reserved-db-instance-id` – Puede asignar su propio identificador a las instancias de base de datos reservadas que adquiera para poder realizar un seguimiento de estas.

Example de adquirir una instancia de base de datos reservada

El siguiente ejemplo adquiere la oferta de instancia de base de datos reservada con el ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f* y le asigna el identificador *MyReservation*.

Para Linux, macOS, o:Unix

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Para Windows:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

El comando devuelve un resultado similar al siguiente:

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State           Description  Offering Type
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

Después de adquirir las instancias de base de datos reservadas, podrá obtener información sobre las instancias de base de datos reservadas.

Para obtener información sobre las instancias de base de datos reservadas de su cuenta de AWS, llame al comando de AWS CLI`describe-reserved-db-instances`, como se muestra en el siguiente ejemplo.

Example de obtener sus instancias de base de datos reservadas

```
aws rds describe-reserved-db-instances
```

El comando devuelve un resultado similar al siguiente:

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

API de RDS

Puede usar la API de RDS para trabajar con instancias de base de datos reservadas:

- Para obtener información sobre las ofertas de instancias de base de datos reservadas disponibles, llame a la operación de API de Amazon RDS `DescribeReservedDBInstancesOfferings`.
- Después de recibir la información sobre las ofertas disponibles de instancias de base de datos reservadas, podrá utilizar dicha información para adquirir una oferta. Llame a la operación de API de RDS `PurchaseReservedDBInstancesOffering` con los siguientes parámetros:
  - `--reserved-db-instances-offering-id` – El ID de la oferta que desea adquirir.

- `--reserved-db-instance-id` – Puede asignar su propio identificador a las instancias de base de datos reservadas que adquiera para poder realizar un seguimiento de estas.
- Después de adquirir las instancias de base de datos reservadas, podrá obtener información sobre las instancias de base de datos reservadas. Llame a la operación de la API de RDS `DescribeReservedDBInstances`.

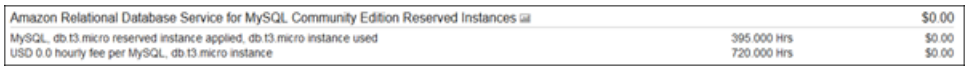
Visualización de la facturación de las instancias de bases de datos reservadas para Amazon Aurora

Puede ver la facturación de las instancias de base de datos reservadas en Billing Dashboard (Panel de facturación) en la Consola de administración de AWS.

Para ver la facturación de una instancia de base de datos reservada

1. Inicie sesión en la Consola de administración de AWS.
2. Desde el account menu (menú de cuenta) en la parte superior derecha, elija Billing Dashboard (Panel de facturación).
3. Elija Bill Details (Detalles de la factura) que aparece en la parte superior derecha del panel.
4. En AWSService Charges (Cargos de servicio), expanda Relational Database Service (Servicio de base de datos relacional).
5. Expanda la Región de AWS en la que se encuentran las instancias de base de datos reservadas, por ejemplo, US West (Oregon) (Oeste de EE. UU. [Oregón]).

   Las instancias de base de datos reservadas y sus cargos por hora del mes en curso se muestran en Amazon Relational Database Service for ***Database Engine*** Reserved Instances (Amazon Relational Database Service para Instancias reservadas del motor de base de datos).

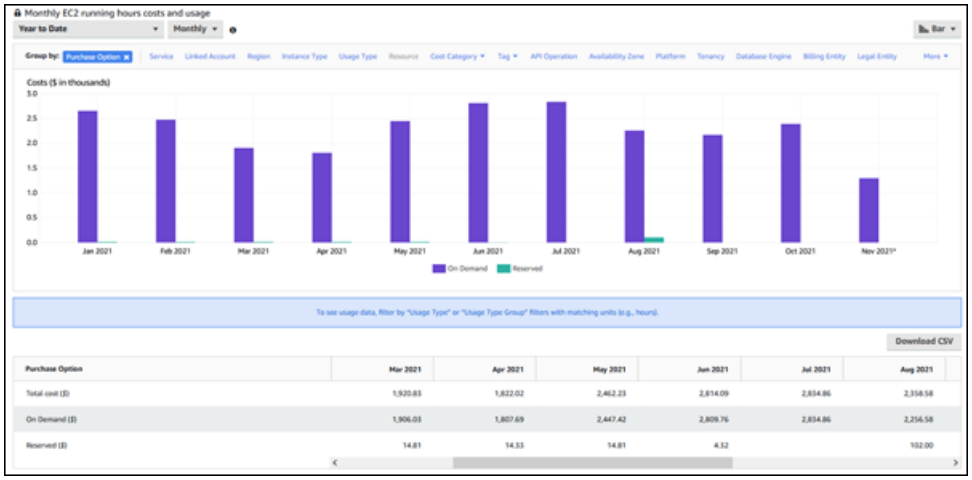
   

   La instancia de base de datos reservada en este ejemplo se compró con un pago total por adelantado, por lo que no hay cargos por hora.
6. Elija el icono Cost Explorer (gráfico de barras) que aparece junto al encabezado de Reserved Instances (Instancias reservadas).

   Cost Explorer muestra el gráfico Monthly EC2 running hours costs and usage (Uso y costos por horas de ejecución de EC2 mensuales).
7. Borre el filtro Usage Type Group (Grupo de tipo de uso) a la derecha del gráfico.

8. Elija el periodo y la unidad de tiempo para los que quiere examinar los costos de uso.

   En el siguiente ejemplo se muestran los costos de uso de las instancias de base de datos bajo demanda y reservadas para el año hasta la fecha por mes.



   Los costos de las instancias de base de datos reservadas de enero a junio de 2021 son cargos mensuales para una instancia parcial inicial, mientras que el costo de agosto de 2021 es un cargo único para una instancia de pago total por adelantado.

   El descuento de la instancia reservada para la instancia de pago inicial parcial venció en junio de 2021, pero la instancia de base de datos no se eliminó. Después de la fecha de vencimiento, simplemente se cobró según la tarifa bajo demanda.

Configuración del entorno para Amazon Aurora

Antes de usar Amazon Aurora por primera vez, realice las siguientes tareas.

Temas

- Cómo crear una Cuenta de AWS
- Creación de un usuario con acceso administrativo
- Conceder acceso programático
- Determinar las necesidades
- Proporcionar acceso al clúster de base de datos en la VPC mediante la creación de un grupo de seguridad

Si ya tiene una Cuenta de AWS, conoce los requisitos de Aurora y prefiere usar los valores predeterminados para los grupos de seguridad de IAM y VPC, vaya directo a Introducción a Amazon Aurora.

Cómo crear una Cuenta de AWS

Si no dispone de una Cuenta de AWS, siga estos pasos para crear una.

Procedimiento para registrarse en Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica o mensaje de texto e indicar un código de verificación en el teclado del teléfono.

   Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS le enviará un correo electrónico de confirmación cuando complete el proceso de registro. Puede ver la actividad de la cuenta y administrarla en cualquier momento entrando en https://aws.amazon.com/ y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de registrarse para obtener una Cuenta de AWS, proteja su Usuario raíz de la cuenta de AWS, habilite AWS IAM Identity Center y cree un usuario administrativo para no usar el usuario raíz en las tareas cotidianas.

Protección de Usuario raíz de la cuenta de AWS

1. Inicie sesión en Consola de administración de AWS como propietario de la cuenta; para ello, elija Usuario raíz e introduzca el correo electrónico de su Cuenta de AWS. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In.
2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitación de un dispositivo MFA virtual para su usuario raíz de la Cuenta de AWS (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center.
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre cómo usar Directorio de IAM Identity Center como origen de identidad, consulte Configuración del acceso de los usuarios con el Directorio de IAM Identity Center predeterminado en la Guía del usuario de AWS IAM Identity Center.

Inicio de sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, use la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario de IAM Identity Center, consulte Inicio de sesión en el portal de acceso de AWS en la Guía del usuario de AWS Sign-In.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center.
2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center.

Conceder acceso programático

Los usuarios necesitan acceso programático si desean interactuar con AWS fuera de la Consola de administración de AWS. La forma de conceder el acceso programático depende del tipo de usuario que acceda a AWS.

Para conceder acceso programático a los usuarios, seleccione una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(Recomendado) Use credenciales de consola como credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK y las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: • Para la AWS CLI, consulte Iniciar sesión para el desarrollo local de AWS en la Guía del usuario de AWS Command Line Interface. • Para los AWS SDK, consulte Iniciar sesión para el desarrollo local de AWS en la Guía de referencia de AWS SDK y herramientas.

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Utiliza credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK y las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: • Para utilizar la AWS CLI, consulte Configuring the AWS CLI to usa AWS IAM Identity Center en la Guía del usuario de AWS Command Line Interface. • Para usar AWS SDK, las herramientas y las API de AWS, consulta IAM Identity Center authentication en la Guía de referencia del SDK y las herramientas de AWS.
IAM	Utiliza credenciales temporales para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK y las API de AWS.	Siguiendo las instrucciones de Uso de credenciales temporales con recursos de AWS de la Guía del usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilizar credenciales a largo plazo para firmar las solicitudes programáticas a la AWS CLI, los AWS SDK o las API de AWS.	Siga las instrucciones de la interfaz que desea utilizar: • Para la AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario de AWS Command Line Interface. • Para ver los AWS SDK y las herramientas, consulte Autenticar mediante credenciales a largo plazo en la Guía de referencia de AWS SDK y herramientas. • Para las API de AWS, consulta Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM.

Determinar las necesidades

El componente básico de Aurora es el clúster de base de datos. Una o más instancias de base de datos pueden pertenecer a un clúster de base de datos. Un clúster de base de datos proporciona una dirección de red llamada punto de enlace del clúster. Sus aplicaciones se conectan al punto de enlace del clúster expuesto por el clúster de base de datos siempre que necesitan acceso a las bases de datos creadas en ese clúster de base de datos. La información especificada al crear el clúster de base de datos controla elementos de configuración como la memoria, el motor de base de datos y la versión, la configuración de red, la seguridad y los periodos de mantenimiento.

Antes de crear un clúster de base de datos y un grupo de seguridad, debe conocer los requisitos de red y de clúster de la base de datos. Aquí se indican algunos aspectos importantes que se deben tener en cuenta:

- Requisitos de recursos: ¿cuáles son los requisitos de memoria y de procesador para su aplicación o su servicio? Usará esta configuración cuando determine la clase de instancia de base de datos que se usará al crear el clúster de base de datos. Para conocer las especificaciones de las clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora.
- VPC, subred y grupo de seguridad–: el clúster de base de datos estará en una nube virtual privada (VPC). Debe configurar reglas de grupo de seguridad para conectarse a un clúster de base de datos. En la siguiente lista se describen las reglas de cada opción de VPC:
  - VPC predeterminada: si su cuenta de AWS tiene una VPC predeterminada en la región de AWS, esa VPC se configura de modo que sea compatible con los clústeres de base de datos. Si especifica la VPC predeterminada al crear el clúster de base de datos:
    - Asegúrese de crear un grupo de seguridad de VPC que autorice las conexiones entre la aplicación o el servicio y el clúster de base de datos de Aurora. Use la opción Security Group (Grupo de seguridad) de la consola de VPC o la AWS CLI para crear grupos de seguridad de VPC. Para obtener información, consulte Paso 3: Crear un grupo de seguridad de VPC.
    - Debe especificar el grupo de subredes de base de datos predeterminado. Si este es el primer clúster de base de datos que ha creado en la región de AWS, Amazon RDS creará el grupo de subredes de base de datos predeterminado cuando cree el clúster de base de datos.
  - VPC definida por el usuario: si desea especificar una VPC definida por el usuario al crear un clúster de base de datos:
    - Asegúrese de crear un grupo de seguridad de VPC que autorice las conexiones entre la aplicación o el servicio y el clúster de base de datos de Aurora. Use la opción Security Group (Grupo de seguridad) de la consola de VPC o la AWS CLI para crear grupos de seguridad de VPC. Para obtener información, consulte Paso 3: Crear un grupo de seguridad de VPC.
    - La VPC debe cumplir algunos requisitos para alojar los clústeres de base de datos, como tener al menos dos subredes, cada una en una zona de disponibilidad diferente. Para obtener información, consulte VPC de Amazon y Amazon Aurora.
    - Debe especificar un grupo de subredes de base de datos que defina qué subredes de esa VPC puede usar el clúster de base de datos. Para obtener información, consulte la sección DB Subnet Group de Uso de una clúster de base de datos en una VPC.
- Alta disponibilidad: ¿necesita compatibilidad con conmutación por error? En Aurora, una implementación Multi-AZ crea una instancia primaria y réplicas de Aurora. Puede configurar la

instancia primaria y las réplicas de Aurora para que estén en zonas de disponibilidad diferentes para permitir la conmutación por error. Es recomendable usar implementaciones Multi-AZ para las cargas de trabajo de producción con el objeto de mantener una alta disponibilidad. Para fines de desarrollo y de pruebas, puede utilizar una implementación no Multi-AZ. Para obtener más información, consulte Alta disponibilidad para Amazon Aurora.

- Políticas de IAM: ¿Tiene la cuenta de AWS políticas que conceden los permisos necesarios para realizar operaciones de Amazon RDS? Si se conecta a AWS con credenciales de IAM, la cuenta de IAM debe tener políticas de IAM que concedan los permisos necesarios para realizar operaciones de Amazon RDS. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.
- Puertos abiertos: ¿con qué puerto TCP o IP se comunicará la base de datos? Los firewalls de algunas compañías podrían bloquear las conexiones al puerto predeterminado para el motor de base de datos. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el nuevo clúster de base de datos. Una vez que cree un clúster de base de datos que escuche en un puerto especificado, puede cambiar el puerto modificando el clúster de base de datos.
- Región de AWS: ¿en qué región de AWS desea que esté la base de datos? Tener la base de datos cerca de la aplicación o el servicio web podría reducir la latencia de la red. Para obtener más información, consulte Regiones y zonas de disponibilidad.

Una vez que tenga la información que necesita para crear el grupo de seguridad y el clúster de base de datos, vaya al siguiente paso.

Proporcionar acceso al clúster de base de datos en la VPC mediante la creación de un grupo de seguridad

El clúster de base de datos se creará en una VPC. Los grupos de seguridad proporcionan acceso al clúster de base de datos en la VPC. Actúan como firewall para el clúster de base de datos asociado y controlan el tráfico entrante y saliente en el nivel del clúster. Por ejemplo, los clústeres de base de datos se crean de manera predeterminada con un firewall y un grupo de seguridad predeterminado que impide el acceso al clúster de base de datos. Por tanto, debe agregar reglas a un grupo de seguridad que le permita conectarse al clúster de base de datos. Use la información de red y de configuración que determinó en el paso anterior para crear reglas que permitan el acceso al clúster de base de datos.

Por ejemplo, si tiene una aplicación que accederá a una base de datos del clúster de base de datos en una VPC, debe agregar una regla de TCP personalizada que especifique el rango de puertos y direcciones IP que la aplicación usará para acceder a la base de datos. Si tiene una aplicación en una instancia de Amazon EC2, puede usar el grupo de seguridad de VPC que configuró para la instancia de Amazon EC2.

Puede configurar la conectividad entre una instancia de Amazon EC2 y el nuevo clúster de base de datos durante la creación del clúster de base de datos. Para obtener más información, consulte Configurar la conectividad de red automática con una instancia de EC2.

> **Tip**
>
> Puede configurar la conectividad de red entre una instancia de Amazon EC2 y un clúster de base de datos automáticamente al crear el clúster de base de datos. Para obtener más información, consulte Configurar la conectividad de red automática con una instancia de EC2.

Para obtener información sobre cómo conectar los recursos en Amazon Lightsail a los clústeres de base de datos, consulte Connect Lightsail resources to Servicios de AWS using VPC peering.

Para obtener más información sobre cómo crear una VPC a fin de usarla con Aurora, consult Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4). Para obtener información sobre situaciones comunes del acceso a una instancia de base de datos, consulte Escenarios de acceso a un clúster de base de datos en una VPC.

Para crear un grupo de seguridad de VPC

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.

   > **Note**
   >
   > Asegúrese de estar en la consola de VPC, no en la consola de RDS.

2. En la esquina superior derecha de la Consola de administración de AWS, elija la región de AWS en la que desea crear el grupo de seguridad de VPC y el clúster de base de datos. En la lista de recursos de Amazon VPC para esa región de AWS, debería ver, al menos, una VPC y varias subredes. Si no es así, no tiene una VPC predeterminada en esa AWS región.
3. En el panel de navegación, elija Security Groups (Grupos de seguridad).

4. Elija Create Security Group (Crear grupo de seguridad).

   Aparece la página Create security group (Crear grupo de seguridad).

5. En Basic details (Detalles básicos), ingrese el Security group name (Nombre del grupo de seguridad) y la Description (Descripción). En VPC, elija la VPC en la que desea crear el clúster de base de datos.

6. En Inbound rules (Reglas de entrada), elija Add rule (Agregar regla).

   a. En Type (Tipo), elija Custom TCP (TCP personalizada).

   b. En Port range (Rango de puertos), ingrese el valor de puerto que se usará en el clúster de base de datos.

   c. En Source (Fuente), elija un nombre de grupo de seguridad o escriba el rango de direcciones IP (valor de CIDR) desde el que accederá al clúster de base de datos. Si elige My IP (Mi IP), esto permite el acceso al clúster de base de datos desde la dirección IP detectada en el navegador.

7. Si necesita agregar más direcciones IP o distintos rangos de puertos, elija Add rule (Agregar regla) e ingrese la información de la regla.

8. (Opcional) En Outbound rules (Reglas de salida), agregue reglas para el tráfico saliente. De forma predeterminada, se permite todo el tráfico de salida.

9. Elija Create Security Group (Crear grupo de seguridad).

Puede usar el grupo de seguridad de VPC que acaba de crear como grupo de seguridad del clúster de base de datos cuando lo cree.

> **Note**
>
> Si se usa una VPC predeterminada, se crea un grupo de subredes predeterminado que abarca todas las subredes de la VPC. Al crear un clúster de base de datos, puede seleccionar la VPC predeterminada y usar default (valor predeterminado) para DB Subnet Group (Grupo de subred de base de datos).

Una vez que haya completado los requisitos de configuración, puede crear un clúster de base de datos mediante los requisitos y el grupo de seguridad según las instrucciones de Creación de un clúster de base de datos de Amazon Aurora. Para obtener información sobre cómo comenzar a crear

un clúster de base de datos que usa un motor de base de datos específico, consulte Introducción a Amazon Aurora.

Introducción a Amazon Aurora

Esta sección le muestra cómo crear un clúster de base de datos Aurora con Amazon RDS y cómo conectarse a él.

Estos siguientes procedimientos son tutoriales que muestran los conceptos básicos de la introducción a Aurora. Las secciones posteriores presentan procedimientos y conceptos de Aurora más avanzados, como los diferentes tipos de puntos de enlace y cómo escalar clústeres de Aurora para ampliarlos o reducirlos.

> **Important**
>
> Debe completar las tareas que aparecen en Configuración del entorno para Amazon Aurora antes de crear un clúster de base de datos o conectarse a él.

Temas

- Creación de un clúster de base de datos de Aurora MySQL y conexión a él
- Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él
- Explicación: crear un servidor web y un clúster de base de datos de Amazon Aurora

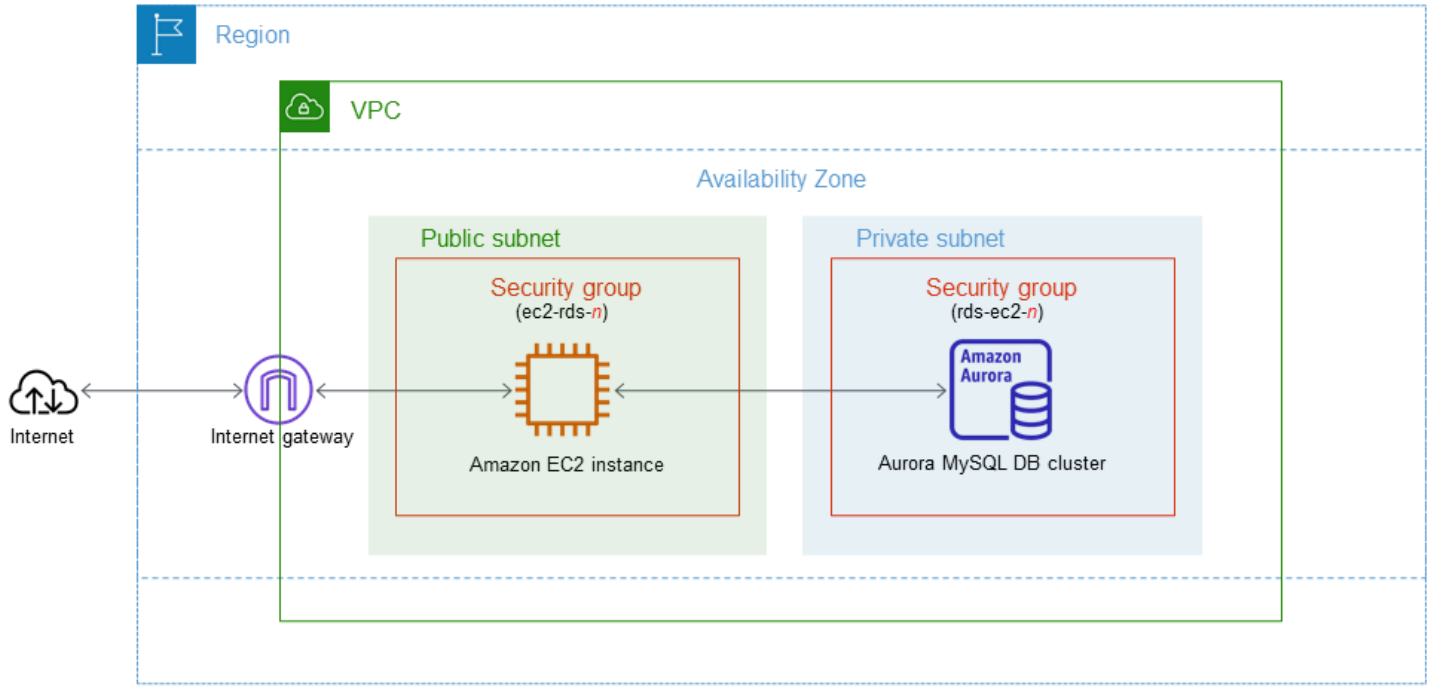
Creación de un clúster de base de datos de Aurora MySQL y conexión a él

En este tutorial, se crea una instancia de EC2 y un clúster de base de datos de Aurora MySQL. El tutorial muestra cómo acceder al clúster de base de datos desde la instancia de EC2 mediante un cliente MySQL estándar. Como práctica recomendada, este tutorial crea un clúster de base de datos privado en una nube privada virtual (VPC). En la mayoría de los casos, otros recursos de la misma VPC, como las instancias de EC2, pueden acceder al clúster de base de datos, pero los recursos ajenos a la VPC no pueden acceder a él.

Tras completar el tutorial, habrá una subred pública y una privada en cada zona de disponibilidad de la VPC. En una zona de disponibilidad, la instancia de EC2 está en la subred pública y la instancia de base de datos está en la subred privada.

> **Important**
>
> La creación de una cuenta de AWS no supone ningún coste. No obstante, al completar este tutorial, puede incurrir en costos de los recursos de AWS que utilice. Puede eliminar estos recursos después de completar el tutorial si ya no son necesarios.

El siguiente diagrama muestra la configuración cuando el tutorial se completa.



Este tutorial le permite crear sus recursos mediante uno de los métodos siguientes:

1. Use la Consola de administración de AWS: Paso 1: crear una instancia de EC2 y Paso 2: crear un clúster de base de datos de Aurora MySQL
2. Use CloudFormation para crear la instancia de base de datos y la instancia de EC2: (Opcional) Crear una VPC, una instancia de EC2 y un clúster de Aurora MySQL mediante CloudFormation

El primer método utiliza Creación sencilla para crear un clúster de base de datos Aurora MySQL privado con la Consola de administración de AWS. Aquí, únicamente debe especificar el tipo de motor de base de datos, el tamaño de instancia de base de datos y el identificador de clúster de base de datos. Easy create (Creación sencilla) utiliza los ajustes predeterminados para otras opciones de configuración.

Cuando usa Creación estándar, se especifican más opciones de configuración al crear un clúster de base de datos. Estas opciones incluyen la configuración de la disponibilidad, la seguridad, las copias de seguridad y el mantenimiento. Para crear un clúster de base de datos público, debe utilizar Creación estándar. Para obtener más información, consulte the section called “Creación de un clúster de base de datos”.

Temas

- Requisitos previos
- Paso 1: crear una instancia de EC2
- Paso 2: crear un clúster de base de datos de Aurora MySQL
- (Opcional) Crear una VPC, una instancia de EC2 y un clúster de Aurora MySQL mediante CloudFormation
- Paso 3: conectarse a un clúster de base de datos de Aurora MySQL
- Paso 4: eliminar la instancia de EC2 y el clúster de base de datos
- (Opcional) Eliminar la instancia de EC2 y el clúster de base de datos creados con CloudFormation
- (Opcional) Conecte el clúster de base de datos a una función de Lambda

Requisitos previos

Antes de empezar, complete los pasos de las siguientes secciones:

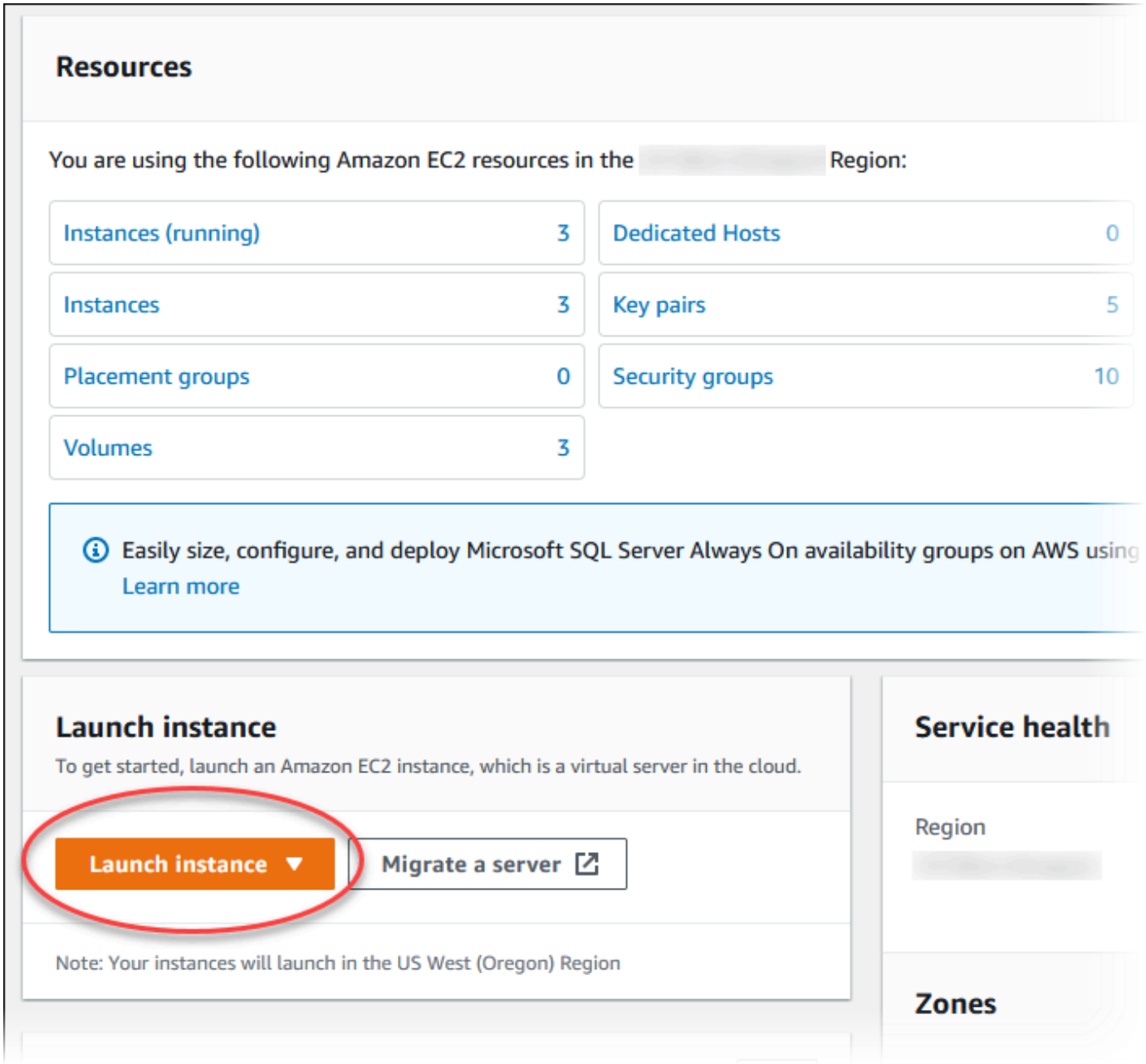
- Cómo crear una Cuenta de AWS
- Creación de un usuario con acceso administrativo

Paso 1: crear una instancia de EC2

Cree una instancia de Amazon EC2 que utilicará para conectarse a la base de datos.

Para crear una instancia EC2;

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En la esquina superior derecha de la Consola de administración de AWS, elija la Región de AWS en la que desea crear la instancia de EC2.
3. Elija Panel de EC2 y, a continuación, Lanzar instancia, como se muestra en la siguiente imagen.

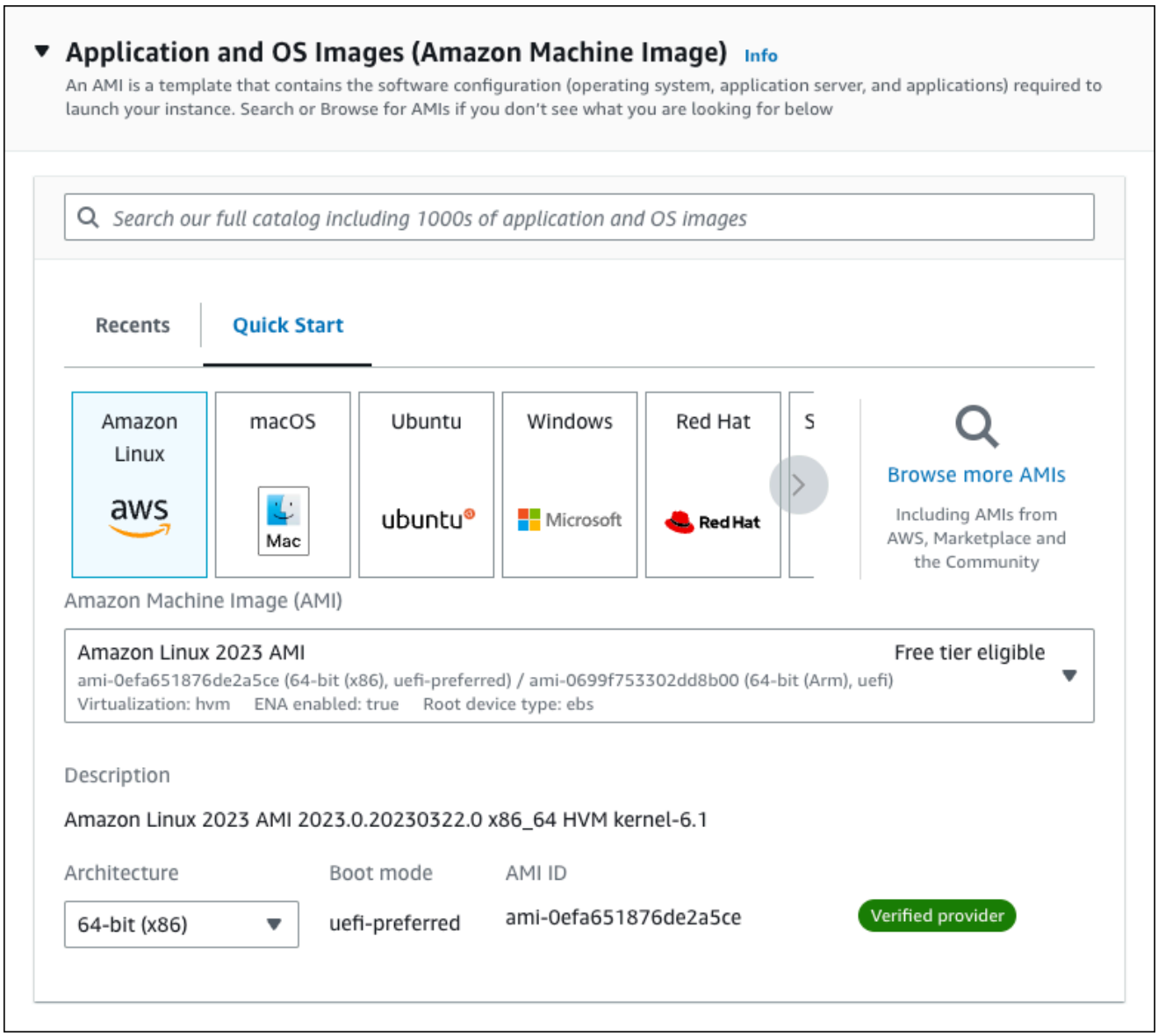


Se abre la página Lanzar una instancia.

4. Elija los siguientes ajustes en la página Lanzar una instancia.

   a. En Name and tags (Nombre y etiquetas), en Name (Nombre), introduzca **`ec2-database-connect`**.

   b. En Imágenes de aplicaciones y sistema operativo (Imagen de máquina de Amazon), elija Amazon Linux y, a continuación, AMI de Amazon Linux 2023. Mantenga los valores predeterminados para las demás opciones.



c. En Instance type (Tipo de instancia), elija t2.micro.

d. En Key pair (login) [Par de claves (inicio)], elija Key pair name (Nombre de par de claves) para utilizar un par de claves existente. Para crear un nuevo par de claves para la instancia de Amazon EC2, que se muestra a continuación, elija Create new key pair (Crear nuevo par de claves) y, a continuación, utilice la ventana Create key pair (Crear un par de claves).

   Para obtener más información sobre la creación de un nuevo par de claves, consulte Crear un par de claves en la Guía del usuario de Amazon EC2.

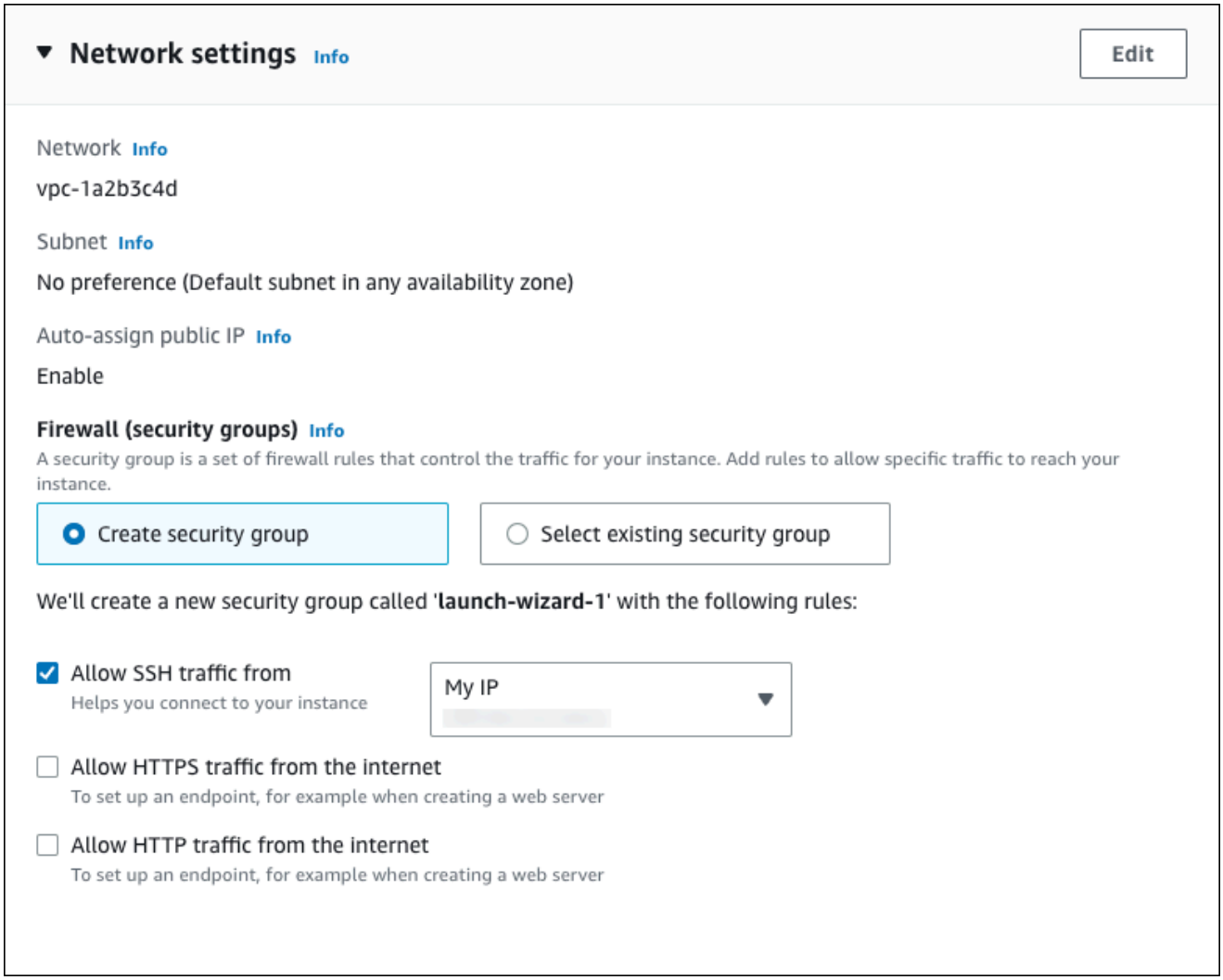
e. En Permitir tráfico de SSH en Configuraciones de red, elija el origen de las conexiones SSH a la instancia de EC2.

Puede elegir My IP (Mi IP) si la dirección IP que se muestra es correcta para las conexiones SSH. De lo contrario, puede determinar la dirección IP que usará para conectarse a las instancias de EC2 en su VPC mediante Secure Shell (SSH). Para determinar su dirección IP pública, en una ventana o pestaña distinta del navegador, puede utilizar el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es 192.0.2.1/32.

En muchos casos, puede conectarse a través de un proveedor de servicios de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, asegúrese de identificar el rango de direcciones IP que utilizan los equipos cliente.

> **Warning**
>
> Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias de EC2 públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias de EC2 mediante SSH.

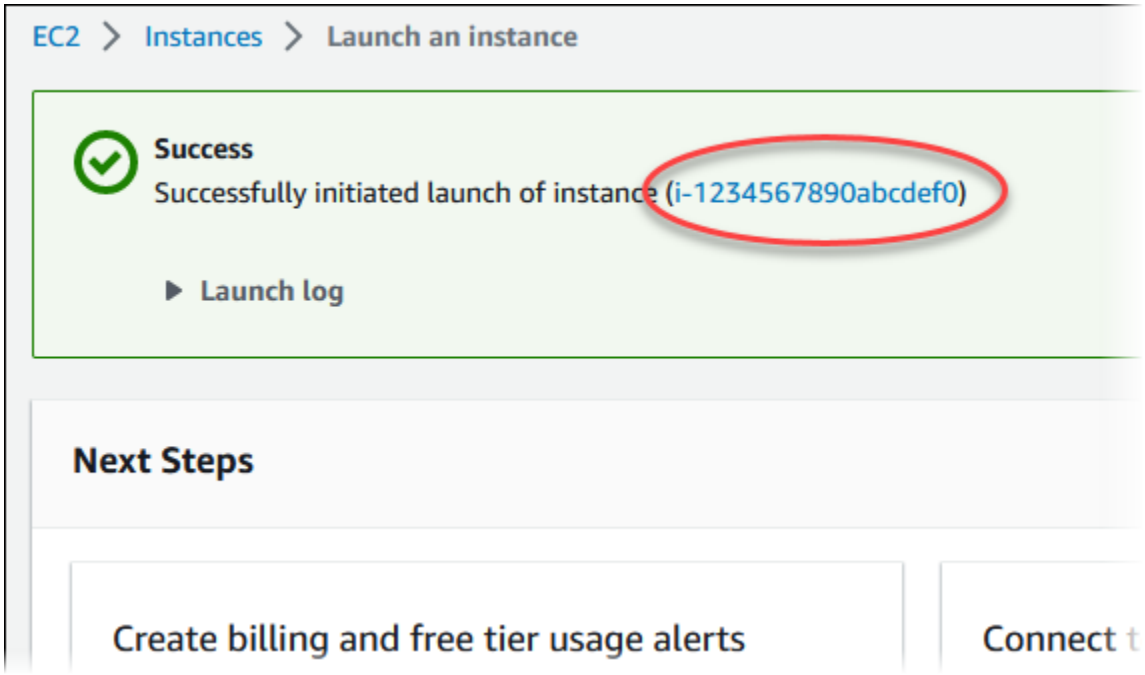
La siguiente imagen muestra un ejemplo de la sección Configuraciones de red.



f. No cambie los valores predeterminados del resto de las secciones.

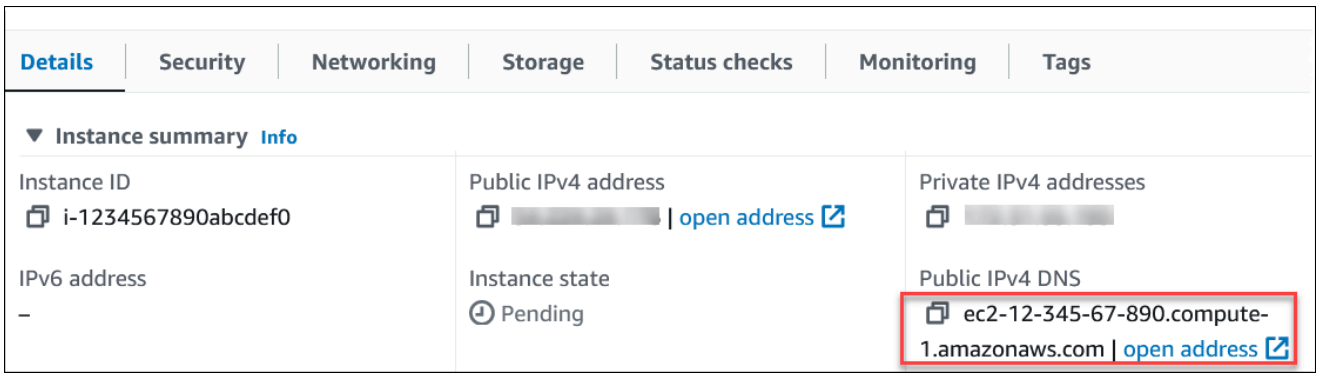
g. Revise un resumen de la configuración de su instancia de EC2 en el panel Resumen; cuando haya terminado, elija Lanzar instancia.

5. En la página Launch Status, que se muestra a continuación, anote el identificador de la nueva instancia de EC2, por ejemplo, `i-1234567890abcdef0`.

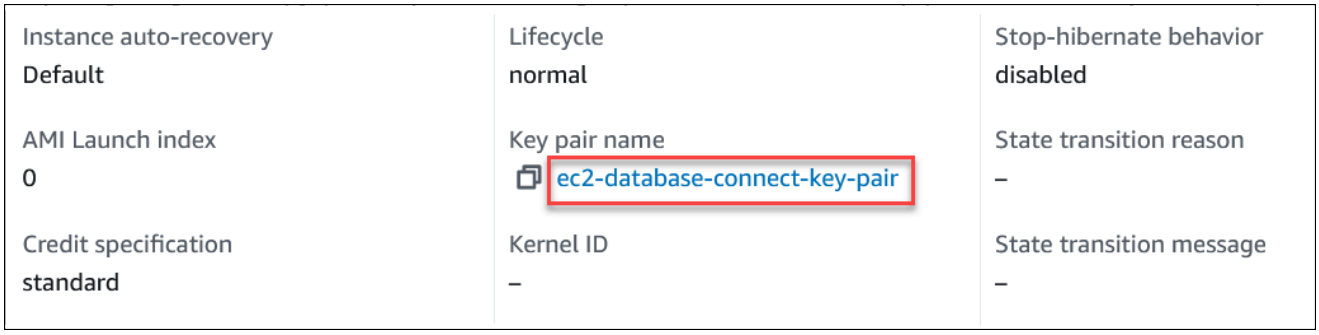


6. Elija el identificador de instancia de EC2 para abrir la lista de instancias de EC2 y, a continuación, seleccione su instancia de EC2.
7. En la pestaña Detalles, anote los siguientes valores, ya que los necesitará cuando se conecte mediante SSH:

   a. En Resumen de la instancia, anote el valor del DNS IPv4 público.



   b. En Detalles de la instancia, anote el valor de Nombre del par de claves.



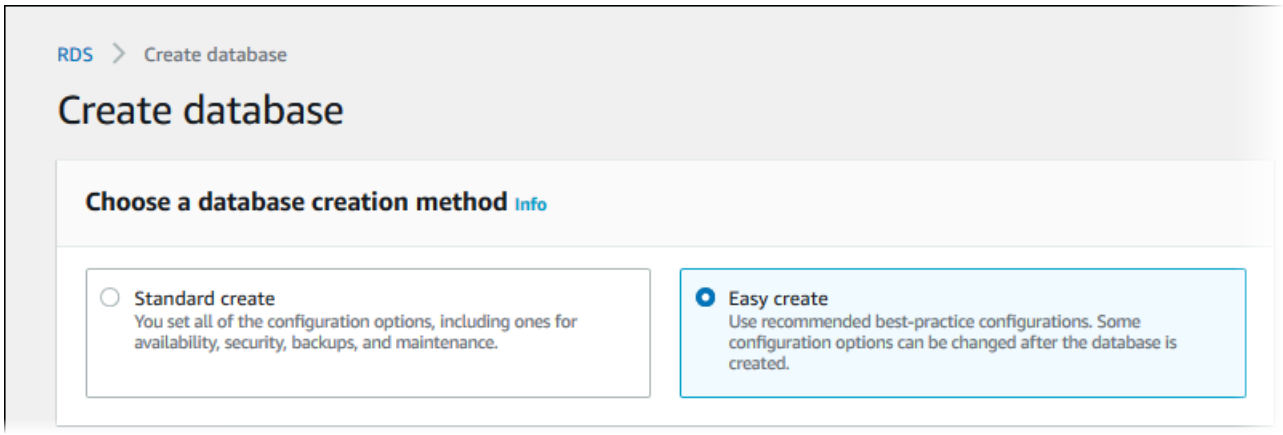
8. Espere hasta que el Estado de la instancia de su instancia de EC2 tenga el estado En ejecución antes de continuar.

Paso 2: crear un clúster de base de datos de Aurora MySQL

En este ejemplo, utilice la opción Creación sencilla para crear un clúster de base de datos de Aurora MySQL con una clase de instancia de base de datos db.r6g.large.

Para crear un clúster de base de datos de Aurora MySQL con Creación sencilla

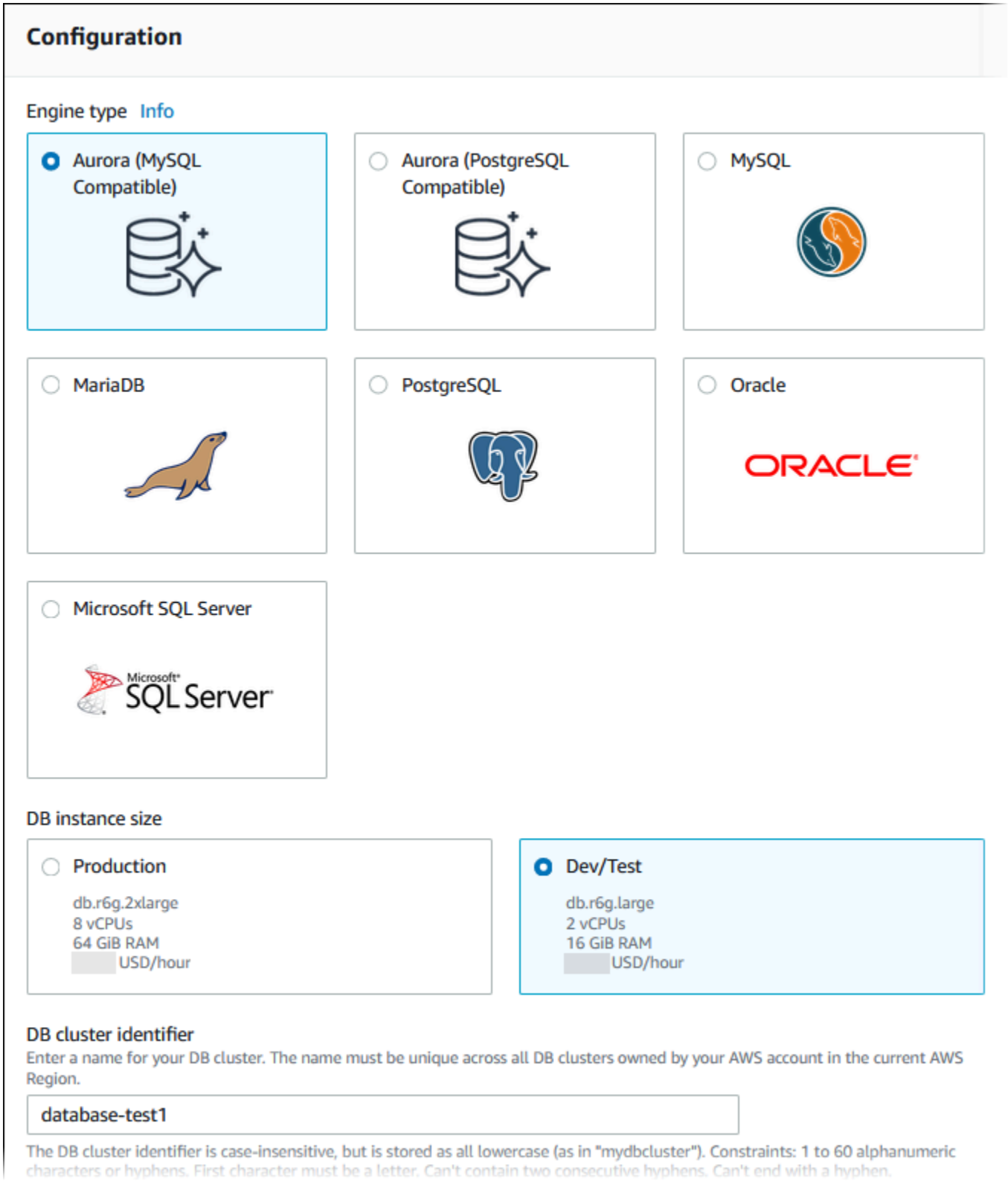
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la consola de Amazon RDS, elija la Región de AWS en la que desea crear el clúster de base de datos.
3. En el panel de navegación, seleccione Databases (Bases de datos).
4. Seleccione Create database (Crear base de datos) y asegúrese de que la opción Easy Create (Creación sencilla) esté seleccionada.



5. En Configuración, elija Aurora (compatible con MySQL) en Tipo de motor.

6. En DB instance size (Tamaño de la instancia de base de datos), seleccione Dev/Test (Desarrollo/Prueba).
7. En Identificador de clúster de bases de datos, introduzca **`database-test1`**.

   La página Create database (Crear base de datos) debe ser similar a la siguiente imagen.



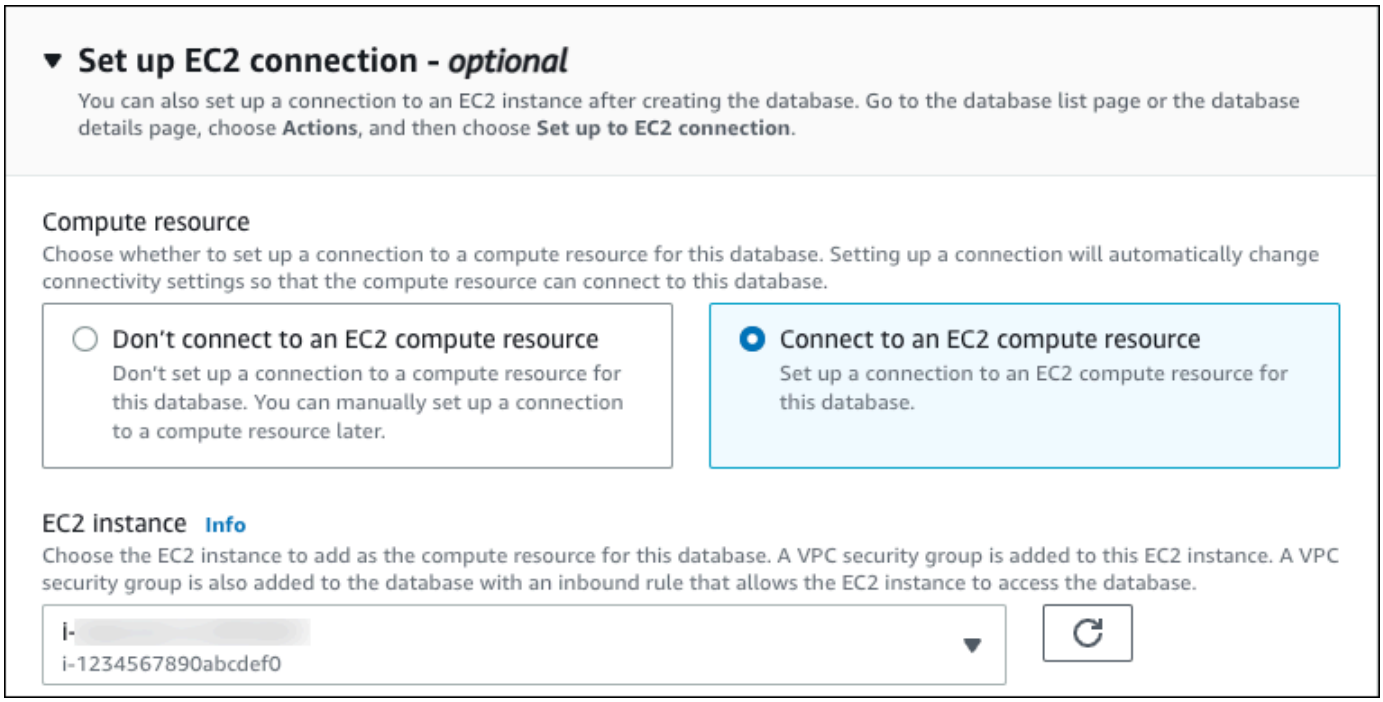
8. En Nombre de usuario maestro, introduzca un nombre para el usuario maestro o deje el nombre predeterminado.

9. Para utilizar una contraseña maestra generada automáticamente para el clúster de base de datos, seleccione Generación automática de contraseña.

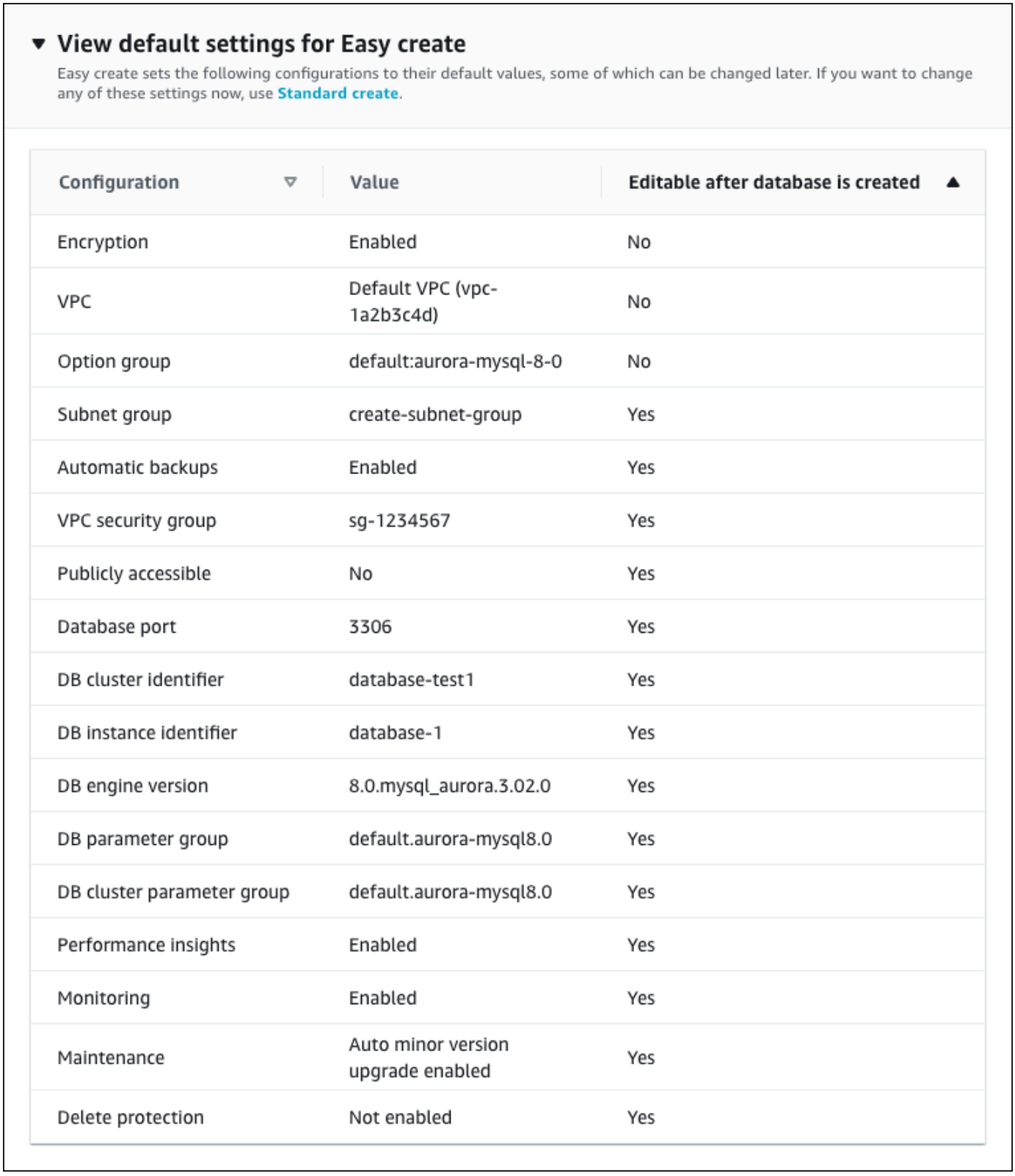
   Para introducir la contraseña maestra, asegúrese de desactivar la casilla Generación automática de contraseña y luego introduzca la misma contraseña en Contraseña maestra y Confirmar contraseña.

10. Para configurar una conexión con la instancia de EC2 que creó anteriormente, abra Configurar conexión a EC2 - (opcional).

    Seleccione Conectarse a un recurso informático de EC2. Elija la instancia de EC2 que ha creado anteriormente.



11. Abra la opción Ver la configuración predeterminada de la creación sencilla.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-mysql-8-0	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	8.0.mysql_aurora.3.02.0	Yes
DB parameter group	default.aurora-mysql8.0	Yes
DB cluster parameter group	default.aurora-mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Puede examinar la configuración predeterminada utilizada con Easy create (Creación sencilla). La columna Editable después de crear la base de datos muestra las opciones que puede cambiar después de crear la base de datos.

- Si una configuración tiene No en esa columna y desea una configuración diferente, puede usar Creación estándar para crear el clúster de base de datos.
- Si una configuración tiene Sí en esa columna y desea una configuración diferente, puede utilizar Creación estándar para crear el clúster de base de datos o modificar el clúster de base de datos después de crearlo para cambiar la configuración.

12. Elija Creación de base de datos.

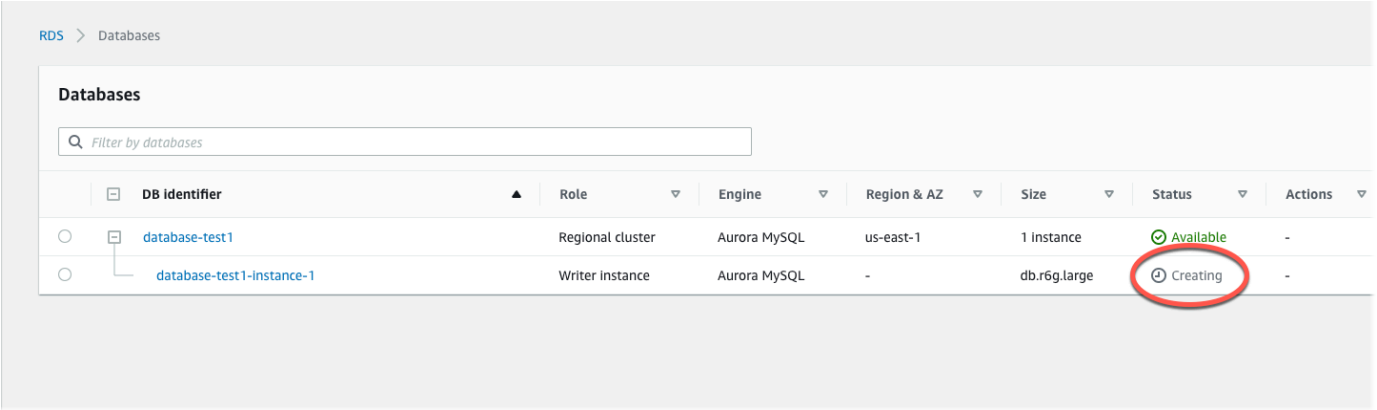
    Para consultar la contraseña y el nombre de usuario maestros del clúster de base de datos, seleccione Ver detalles de credenciales.

    Puede utilizar la contraseña y el nombre de usuario que aparecen para conectarse al clúster de base de datos como el usuario maestro.

    > **Important**
    >
    > No puede ver la contraseña de usuario maestro de nuevo. Si no la registra, es posible que tenga que cambiarla.
    > Si tiene que cambiar la contraseña de usuario maestro después de que el clúster de base de datos esté disponible, puede modificar el clúster de base de datos para ello. Para obtener más información sobre la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

13. En la lista Bases de datos, seleccione el nombre del nuevo clúster de base de datos de Aurora MySQL para ver sus detalles.

    La instancia de escritor tiene el estado Creando hasta que el clúster de base de datos está listo para usarse.



Cuando el estado de la instancia de escritor cambie a Disponible, puede conectarse al clúster de la base de datos. Dependiendo de la clase de instancia de base de datos y de la cantidad de almacenamiento, es posible que el nuevo clúster de base de datos tarde hasta 20 minutos en estar disponible.

(Opcional) Crear una VPC, una instancia de EC2 y un clúster de Aurora MySQL mediante CloudFormation

En lugar de utilizar la consola para crear la VPC, la instancia de EC2 y el clúster de base de datos de Aurora MySQL, puede utilizar CloudFormation para aprovisionar recursos de AWS tratando la infraestructura como código. Para ayudarle a organizar sus recursos de AWS en unidades más pequeñas y fáciles de administrar, puede utilizar la funcionalidad de pila anidada de CloudFormation. Para obtener más información, consulte Creación de una pila en la consola CloudFormation y Uso de pilas anidadas.

> **Important**
>
> CloudFormation es gratuito, pero los recursos que CloudFormation crea están activos. Se le facturan las tarifas de uso estándar por estos recursos hasta que los finalice. Para obtener más información, consulte Precios de Amazon Aurora.

Para crear sus recursos con la consola CloudFormation, siga estos pasos:

- Paso 1: Descargar la plantilla de CloudFormation
- Paso 2: Configurar los recursos mediante CloudFormation

Descargar la plantilla de CloudFormation

Una plantilla de CloudFormation es un archivo de texto con formato JSON o YAML que contiene la información de configuración de los recursos que desea crear en la pila. Esta plantilla también crea una VPC y un host bastión para usted junto con el clúster de Aurora.

Para descargar el archivo de plantilla, abra el enlace Aurora MySQL CloudFormation template.

En la página de Github, haga clic en el botón Descargar archivo sin procesar para guardar el archivo YAML de la plantilla.

Configurar los recursos mediante CloudFormation

> **Note**
>
> Antes de iniciar este proceso, asegúrese de tener un par de claves para una instancia EC2 en su Cuenta de AWS. Para obtener más información, consulte Pares de claves de Amazon EC2 e instancias Linux.

Al utilizar la plantilla de CloudFormation, debe seleccionar los parámetros correctos para asegurarse de que los recursos se crean correctamente. Siga los pasos que se indican a continuación:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Elija Crear pila.
3. En la sección Especificar la plantilla, seleccione Cargar un archivo de plantilla desde el ordenador y Siguiente.
4. En la página Especificar detalles de la pila, introduzca los siguientes parámetros:
   a. Ponga el Nombre de la pila en AurMySQLTestStack.
   b. En Parámetros, defina las zonas de disponibilidad seleccionando dos zonas de disponibilidad.
   c. En Configuración de host bastión de Linux, en Nombre de la clave, seleccione un par de claves para iniciar sesión en su instancia de EC2.
   d. En los ajustes de Configuración de host bastión de Linux, ponga el rango de IP permitido en su dirección IP. Para conectarse a las instancias de EC2 de su VPC mediante Secure Shell (SSH), determine su dirección IP pública mediante el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es 192.0.2.1/32.

> **Warning**
>
> Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias de EC2 públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias de EC2 mediante SSH.

e. En Configuración general de la base de datos, ponga la Clase de instancia de base de datos en db.r6g.large.

f. Ponga el Nombre de la base de datos en **`database-test1`**.

g. En Nombre de usuario maestro, introduzca un nombre para el usuario maestro.

h. Ponga Administrar contraseña de usuario maestro de base de datos con Secrets Manager en `false` para este tutorial.

i. En Contraseña de base de datos, ponga la contraseña que desee. Recuerde esta contraseña para poder ver los pasos adicionales del tutorial.

j. Ponga Implementación multi-AZ en `false`.

k. Deje el resto de la configuración con los valores predeterminados. Haga clic en Siguiente para continuar.

5. En la página Configurar opciones de pila, deje todas las opciones predeterminadas. Haga clic en Siguiente para continuar.
6. En la página Revisar la pila, seleccione Enviar después de comprobar las opciones de base de datos y de host bastión de Linux.

Una vez finalizado el proceso de creación de la pila, consulte las pilas con los nombres BastionStack y AMSNS para anotar la información que necesita a fin de conectarse a la base de datos. Para obtener más información, consulte Viewing CloudFormation stack data and resources on the Consola de administración de AWS.

Paso 3: conectarse a un clúster de base de datos de Aurora MySQL

Puede usar cualquier aplicación cliente de SQL estándar para conectarse al clúster de base de datos. En este ejemplo, se conecta a un clúster de base de datos de MySQL mediante el cliente de línea de comandos mysql.

Para conectarse a un clúster de base de datos de Aurora MySQL

1. Busque el punto de conexión (nombre de DNS) y el número de puerto de la instancia de escritor de su clúster de base de datos.

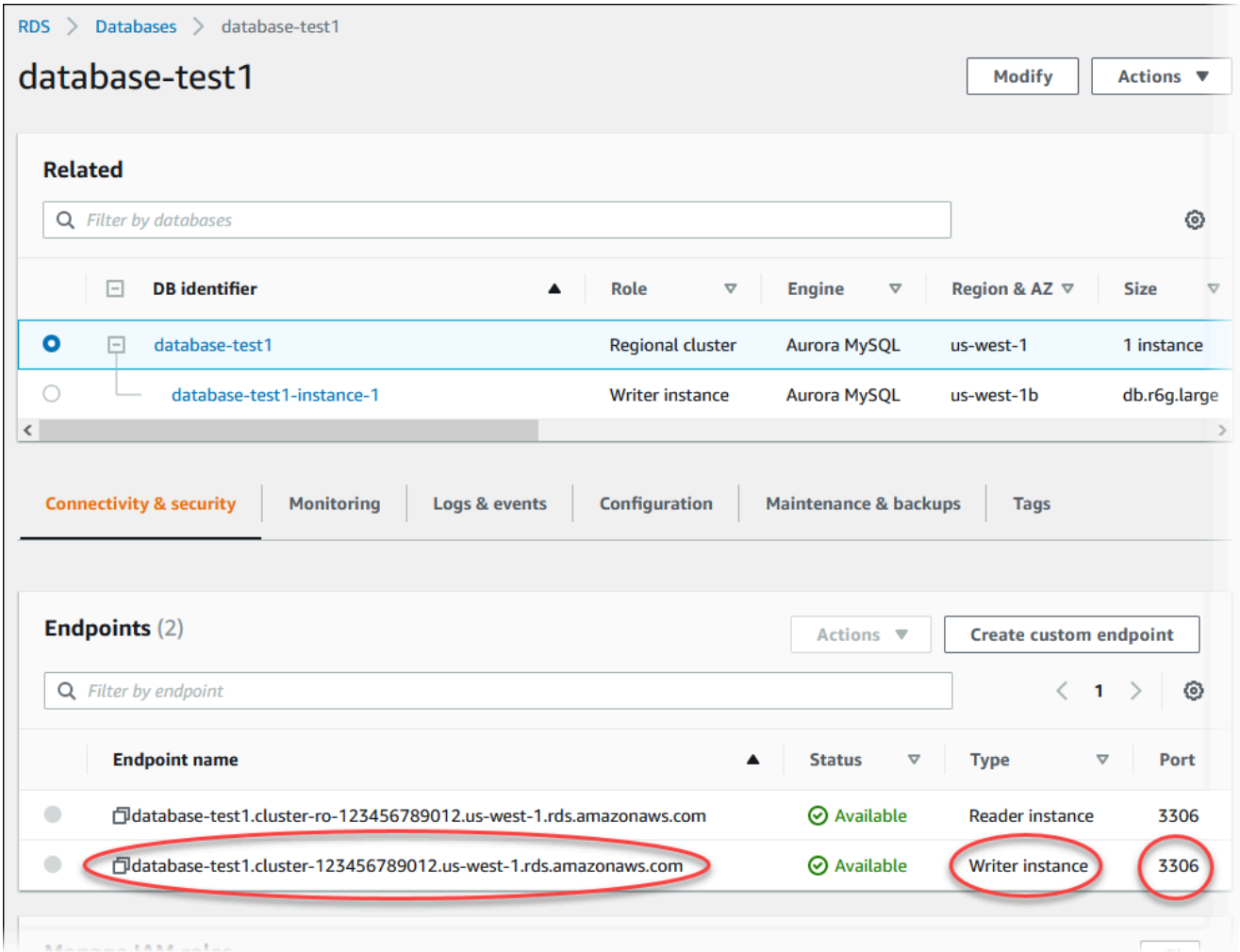
   a. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

   b. En la esquina superior derecha de la consola de Amazon RDS, elija la Región de AWS del clúster de base de datos.

   c. En el panel de navegación, elija Databases (Bases de datos).

   d. Elija el nombre del clúster de base de datos de Aurora MySQL para ver sus detalles.

   e. En la pestaña Conectividad y seguridad, copie el punto de conexión de la instancia de escritor. También anote el número de puerto. Necesita el punto de conexión y el número de puerto para conectarse al clúster de base de datos.



2. Conéctese a la instancia de EC2 que ha creado anteriormente siguiendo los pasos que se indican en Conexión con la instancia de Linux en la Guía del usuario de Amazon EC2.

   Le recomendamos que se conecte a la instancia de EC2 mediante SSH. Si la utilidad de cliente SSH está instalada en Windows, Linux o Mac, puede conectarse a la instancia con el siguiente formato de comando:

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Por ejemplo, supongamos que `ec2-database-connect-key-pair.pem` está almacenado en `/dir1` en Linux y que el DNS IPv4 público de su instancia de EC2 es `ec2-12-345-678-90.compute-1.amazonaws.com`. En ese caso, su comando SSH tendría el siguiente aspecto:

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. Obtenga las correcciones de errores y las actualizaciones de seguridad más recientes actualizando el software en su instancia de EC2. Para ello, utilice el siguiente comando.

   > **Note**
   >
   > La opción `-y` instala las actualizaciones sin necesidad de confirmación. Para examinar las actualizaciones antes de la instalación, omita esta opción.

   ```
   sudo dnf update -y
   ```

4. Para instalar el cliente de línea de comandos mysql de MariaDB en Amazon Linux 2023, ejecute el siguiente comando:

   ```
   sudo dnf install mariadb105
   ```

5. Conéctese al clúster de base de datos de Aurora MySQL. Por ejemplo, introduzca el siguiente comando. Esta acción le permite conectarse al clúster de base de datos de Aurora MySQL mediante el cliente de MySQL.

Sustituya el punto de conexión de la instancia de escritor por *endpoint* y sustituya el nombre de usuario maestro que utilizó por *admin*. Proporcione la contraseña maestra que utilizó cuando se le solicite una contraseña.

```
mysql -h endpoint -P 3306 -u admin -p
```

Una vez especificada la contraseña del usuario, debería ver un resultado similar al siguiente.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 217
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Para obtener más información acerca de la conexión a un clúster de base de datos de Aurora MySQL, consulte Conexión a un clúster de base de datos Amazon Aurora MySQL. Si no puede conectarse al clúster de base de datos, consulte No puede conectarse a la instancia de base de datos de Amazon RDS.

Por motivos de seguridad, se recomienda utilizar conexiones cifradas. Utilice sólo una conexión MySQL sin cifrar cuando el cliente y el servidor están en la misma VPC y la red es de confianza. Para obtener información sobre el uso de de conexiones cifradas, consulte Conexión a Aurora MySQL mediante SSL.

6. Ejecutar comandos SQL.

   Por ejemplo, el siguiente comando de SQL muestra la fecha y la hora actuales:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Paso 4: eliminar la instancia de EC2 y el clúster de base de datos

Después de conectarse y explorar la instancia de EC2 de muestra y el clúster de base de datos que creó, elimínelos para que no le sigan cobrando por ellos.

Si ha utilizado CloudFormation para crear recursos, omita este paso y vaya al siguiente.

Para eliminar la instancia de EC2

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En el panel de navegación, seleccione Instances (Instancias).
3. Seleccione la instancia de EC2 y elija Estado de la instancia y Terminar instancia.
4. Cuando se le indique que confirme, elija Terminar.

Para obtener más información sobre la eliminación de una instancia de EC2, consulte Terminar la instancia en la Guía del usuario de Amazon EC2.

Para eliminar el clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos asociada al clúster de base de datos.
3. En Actions (Acciones), elija Delete (Eliminar).
4. Desmarque ¿Crear instantánea final?
5. Complete la confirmación y seleccione Eliminar.

Después de eliminar todas las instancias de base de datos asociadas al clúster de base de datos, se elimina automáticamente el clúster de base de datos .

(Opcional) Eliminar la instancia de EC2 y el clúster de base de datos creados con CloudFormation

Si ha utilizado CloudFormation para crear recursos, elimine la pila de CloudFormation después de conectarse a la instancia de EC2 y el clúster de base de datos de muestra y de explorarlos; de este modo, ya no se le cobrará por ellos.

Para eliminar los recursos de CloudFormation

1. Abra la consola de CloudFormation.

2. En la página Pilas de la consola de CloudFormation, seleccione la pila raíz (la pila sin el nombre VPCStack, BastionStack o AMSNS).
3. Elija Eliminar.
4. Cuando se le pida confirmación, seleccione Eliminar pila.

Para obtener información sobre cómo eliminar una pila en CloudFormation, consulte Eliminación de una pila en la consola de CloudFormation, en la Guía del usuario de AWS CloudFormation.

(Opcional) Conecte el clúster de base de datos a una función de Lambda

También puede conectar el clúster de base de datos de Aurora MySQL a un recurso de computación sin servidor de Lambda. Las funciones de Lambda permiten ejecutar código sin aprovisionar ni administrar la infraestructura. Una función de Lambda también permite responder automáticamente a las solicitudes de ejecución de código a cualquier escala, desde una docena de eventos al día hasta cientos de eventos por segundo. Para obtener más información, consulte Conexión automática de una función de Lambda y un clúster de base de datos de Aurora.

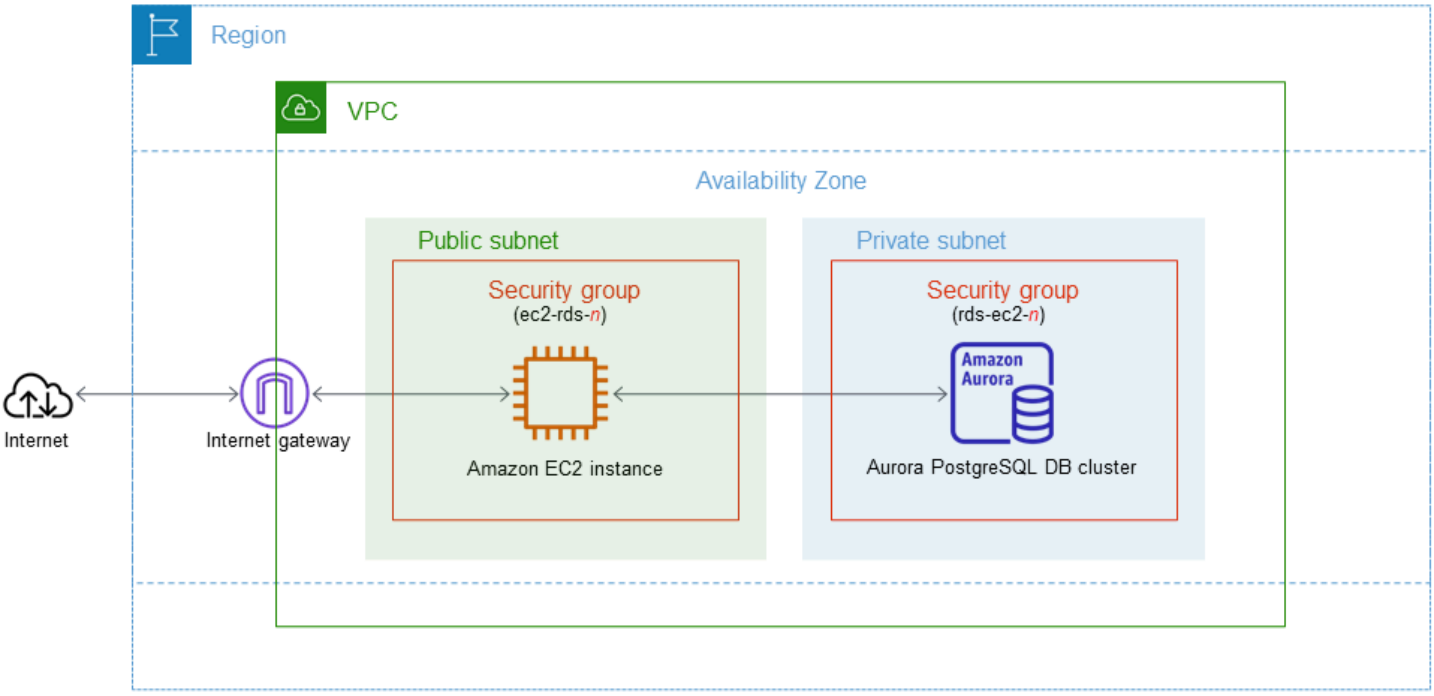
Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él

En este tutorial, se crea una instancia de EC2 y un clúster de base de datos de Aurora PostgreSQL. El tutorial muestra cómo acceder al clúster de base de datos desde la instancia de EC2 mediante un cliente PostgreSQL estándar. Como práctica recomendada, este tutorial crea un clúster de base de datos privado en una nube privada virtual (VPC). En la mayoría de los casos, otros recursos de la misma VPC, como las instancias de EC2, pueden acceder al clúster de base de datos, pero los recursos ajenos a la VPC no pueden acceder a él.

Tras completar el tutorial, habrá una subred pública y una privada en cada zona de disponibilidad de la VPC. En una zona de disponibilidad, la instancia de EC2 está en la subred pública y la instancia de base de datos está en la subred privada.

> **Important**
>
> La creación de una cuenta de AWS no supone ningún coste. No obstante, al completar este tutorial, puede incurrir en costos de los recursos de AWS que utilice. Puede eliminar estos recursos después de completar el tutorial si ya no son necesarios.

El siguiente diagrama muestra la configuración cuando el tutorial se completa.



Este tutorial le permite crear sus recursos mediante uno de los métodos siguientes:

1. Usar la Consola de administración de AWS: Paso 1: crear una instancia de EC2 y Paso 2: crear un clúster de base de datos de Aurora PostgreSQL
2. Use CloudFormation para crear la instancia de base de datos y la instancia de EC2: (Opcional) Crear una VPC, una instancia EC2 y un clúster de Aurora PostgreSQL mediante CloudFormation

El primer método utiliza Creación sencilla para crear un clúster de base de datos Aurora PostgreSQL privado con la Consola de administración de AWS. Aquí, únicamente debe especificar el tipo de motor de base de datos, el tamaño de instancia de base de datos y el identificador de clúster de base de datos. Easy create (Creación sencilla) utiliza los ajustes predeterminados para otras opciones de configuración.

Cuando usa Creación estándar, se especifican más opciones de configuración al crear un clúster de base de datos. Estas opciones incluyen la configuración de la disponibilidad, la seguridad, las copias de seguridad y el mantenimiento. Para crear un clúster de base de datos público, debe utilizar Creación estándar. Para obtener más información, consulte the section called “Creación de un clúster de base de datos”.

Temas

- Requisitos previos
- Paso 1: crear una instancia de EC2
- Paso 2: crear un clúster de base de datos de Aurora PostgreSQL
- (Opcional) Crear una VPC, una instancia EC2 y un clúster de Aurora PostgreSQL mediante CloudFormation
- Paso 3: conectarse a un clúster de base de datos de Aurora PostgreSQL
- Paso 4: eliminar la instancia de EC2 y el clúster de base de datos
- (Opcional) Eliminar la instancia de EC2 y el clúster de base de datos creados con CloudFormation
- (Opcional) Conecte el clúster de base de datos a una función de Lambda

Requisitos previos

Antes de empezar, complete los pasos de las siguientes secciones:

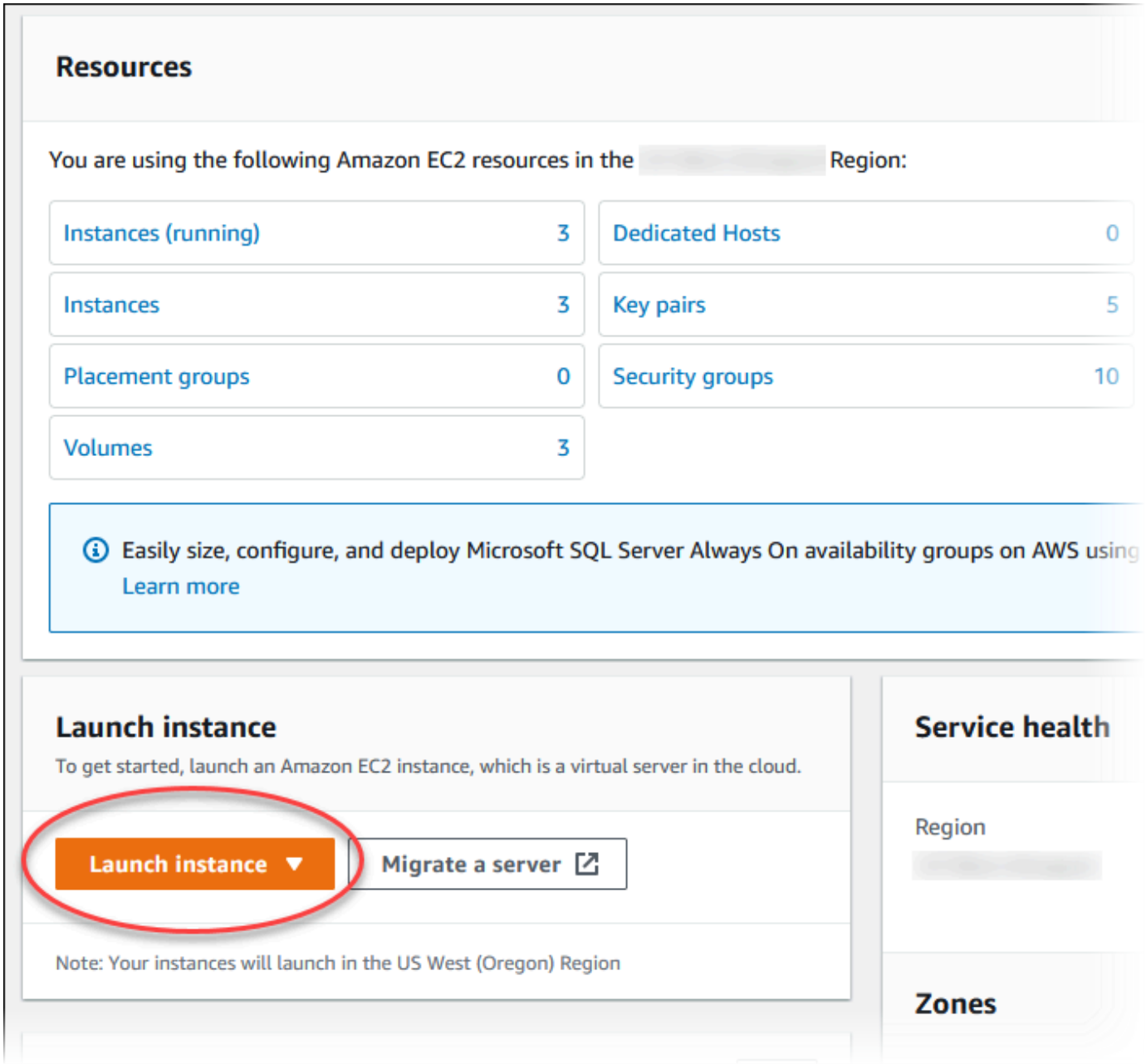
- Cómo crear una Cuenta de AWS
- Creación de un usuario con acceso administrativo

Paso 1: crear una instancia de EC2

Cree una instancia de Amazon EC2 que utilicará para conectarse a la base de datos.

Para crear una instancia EC2;

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En la esquina superior derecha de la Consola de administración de AWS, elija la Región de AWS en la que desea crear la instancia de EC2.
3. Elija Panel de EC2 y, a continuación, Lanzar instancia, como se muestra en la siguiente imagen.

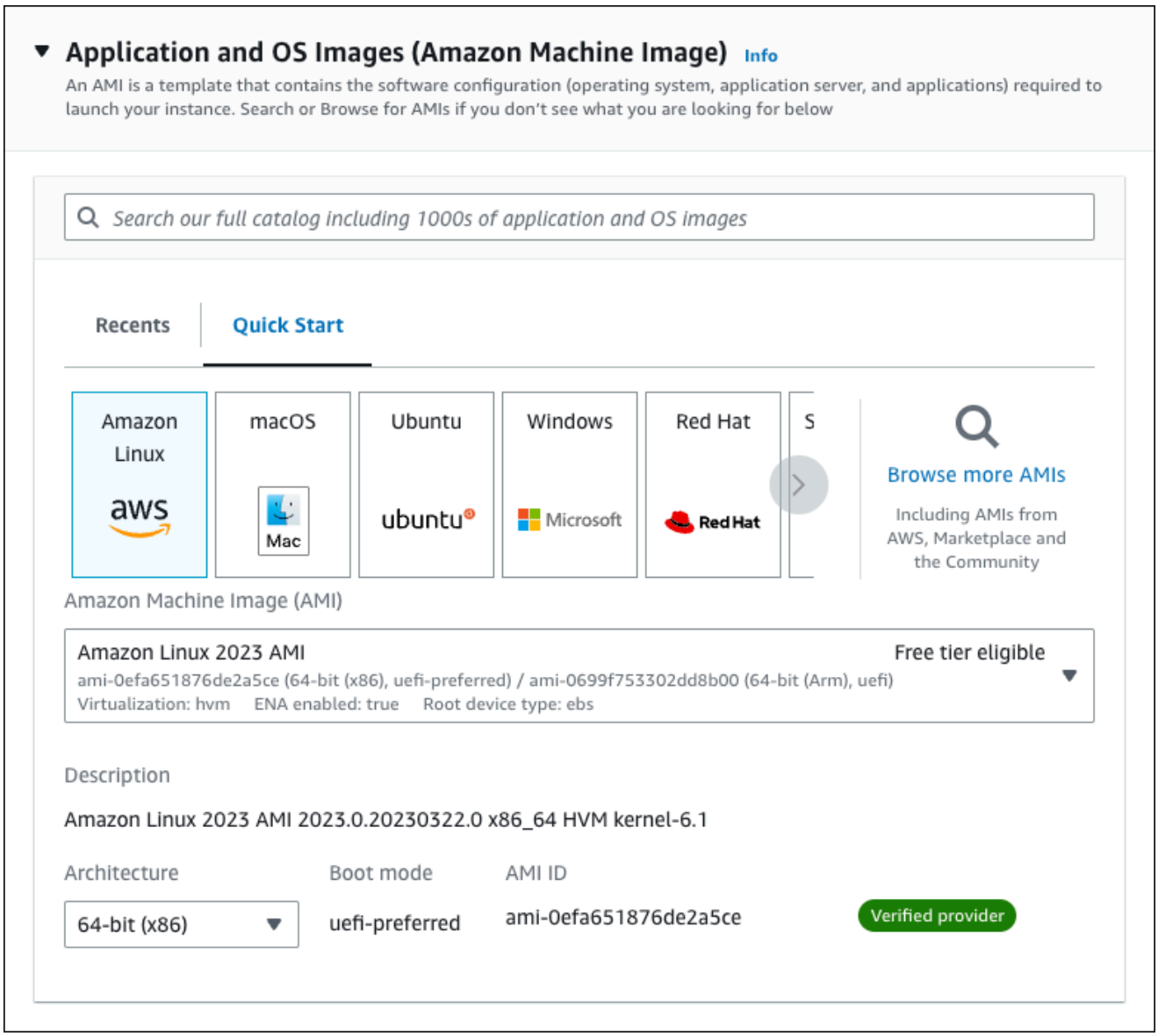


Se abre la página Lanzar una instancia.

4. Elija los siguientes ajustes en la página Lanzar una instancia.

   a. En Name and tags (Nombre y etiquetas), en Name (Nombre), introduzca **`ec2-database-connect`**.

   b. En Imágenes de aplicaciones y sistema operativo (Imagen de máquina de Amazon), elija Amazon Linux y, a continuación, AMI de Amazon Linux 2023. Mantenga los valores predeterminados para las demás opciones.



c. En Instance type (Tipo de instancia), elija t2.micro.

d. En Key pair (login) [Par de claves (inicio)], elija Key pair name (Nombre de par de claves) para utilizar un par de claves existente. Para crear un nuevo par de claves para la instancia de Amazon EC2, que se muestra a continuación, elija Create new key pair (Crear nuevo par de claves) y, a continuación, utilice la ventana Create key pair (Crear un par de claves).

   Para obtener más información sobre la creación de un nuevo par de claves, consulte Crear un par de claves en la Guía del usuario de Amazon EC2.

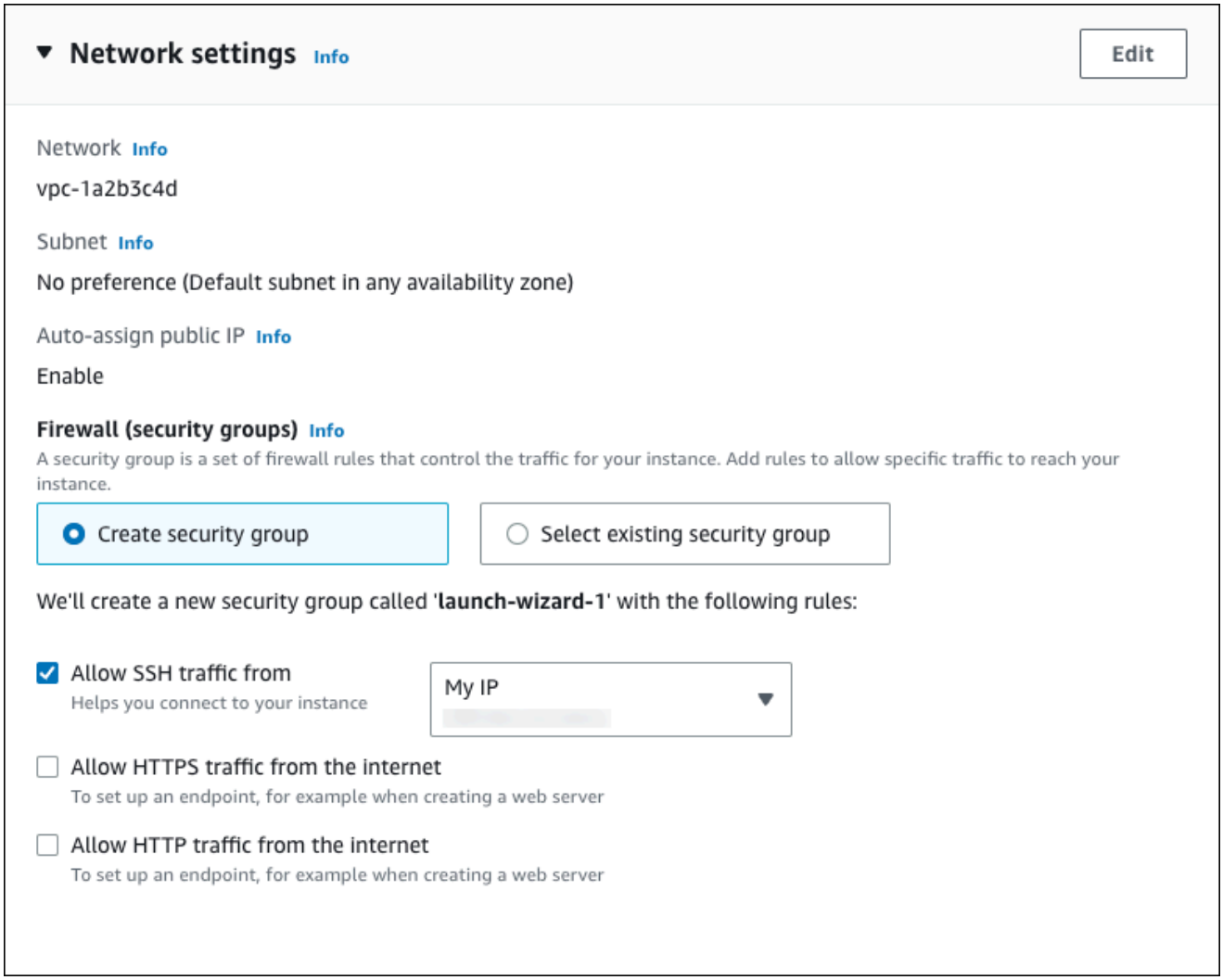
e. En Permitir tráfico de SSH en Configuraciones de red, elija el origen de las conexiones SSH a la instancia de EC2.

Puede elegir My IP (Mi IP) si la dirección IP que se muestra es correcta para las conexiones SSH. De lo contrario, puede determinar la dirección IP que usará para conectarse a las instancias de EC2 en su VPC mediante Secure Shell (SSH). Para determinar su dirección IP pública, en una ventana o pestaña distinta del navegador, puede utilizar el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es 192.0.2.1/32.

En muchos casos, puede conectarse a través de un proveedor de servicios de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, asegúrese de identificar el rango de direcciones IP que utilizan los equipos cliente.

> **Warning**
>
> Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias de EC2 públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias de EC2 mediante SSH.

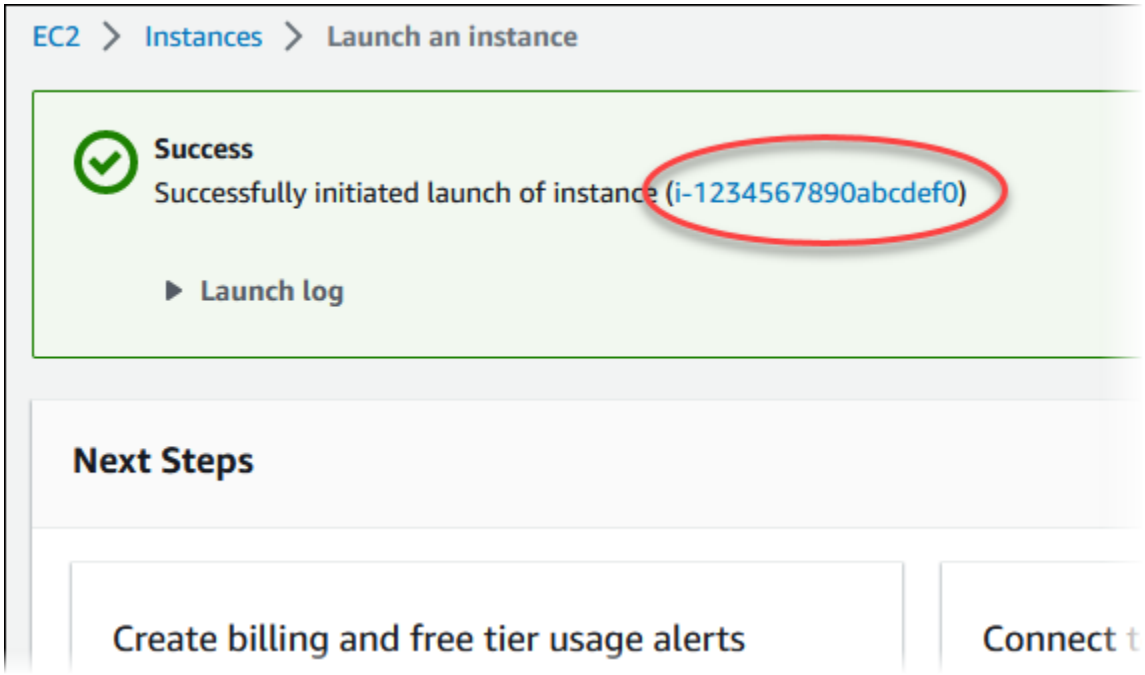
La siguiente imagen muestra un ejemplo de la sección Configuraciones de red.



f. No cambie los valores predeterminados del resto de las secciones.

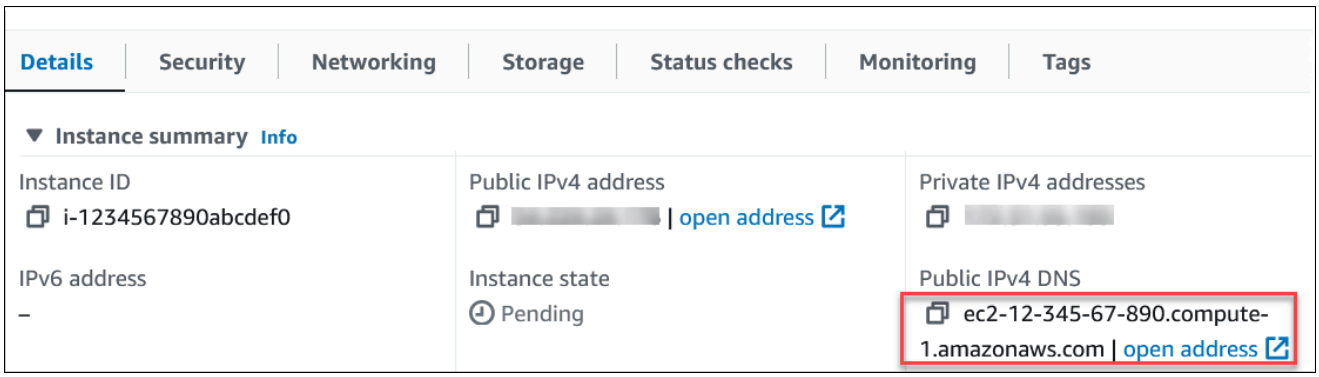
g. Revise un resumen de la configuración de su instancia de EC2 en el panel Resumen; cuando haya terminado, elija Lanzar instancia.

5. En la página Launch Status, que se muestra a continuación, anote el identificador de la nueva instancia de EC2, por ejemplo, `i-1234567890abcdef0`.

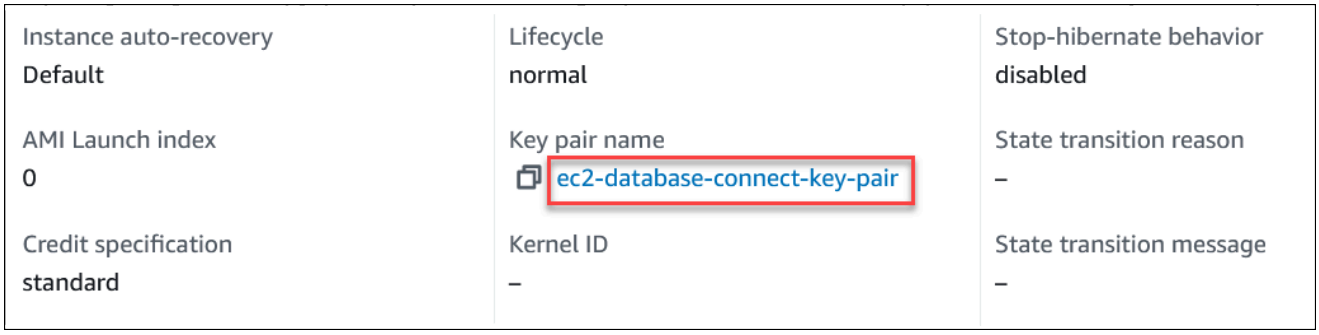


6. Elija el identificador de instancia de EC2 para abrir la lista de instancias de EC2 y, a continuación, seleccione su instancia de EC2.
7. En la pestaña Detalles, anote los siguientes valores, ya que los necesitará cuando se conecte mediante SSH:

   a. En Resumen de la instancia, anote el valor del DNS IPv4 público.



   b. En Detalles de la instancia, anote el valor de Nombre del par de claves.



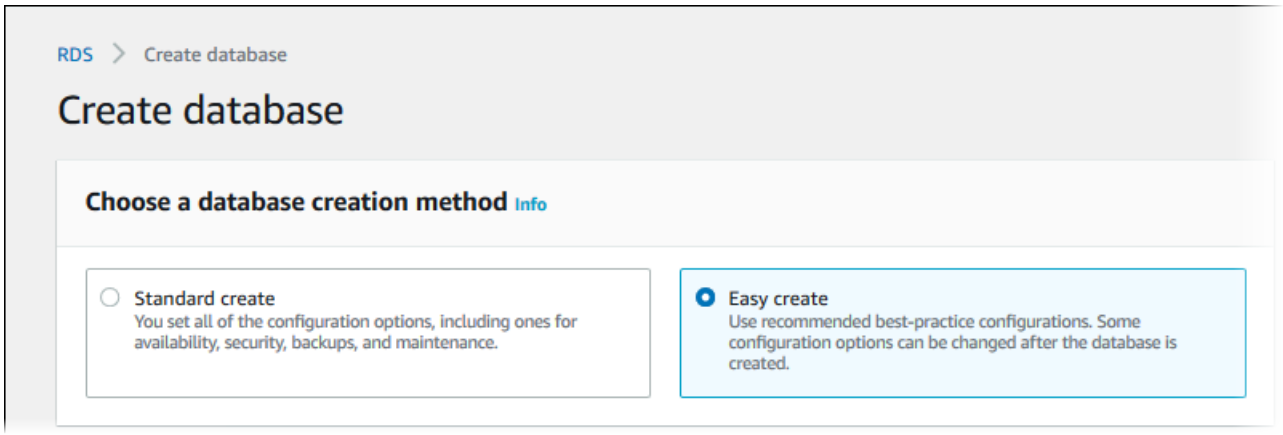
8. Espere hasta que el Estado de la instancia de su instancia de EC2 tenga el estado En ejecución antes de continuar.

Paso 2: crear un clúster de base de datos de Aurora PostgreSQL

En este ejemplo, utilice la opción Creación sencilla para crear un clúster de base de datos de Aurora PostgreSQL con una clase de instancia de base de datos db.t4g.large.

Para crear un clúster de base de datos de Aurora PostgreSQL con la opción Creación sencilla

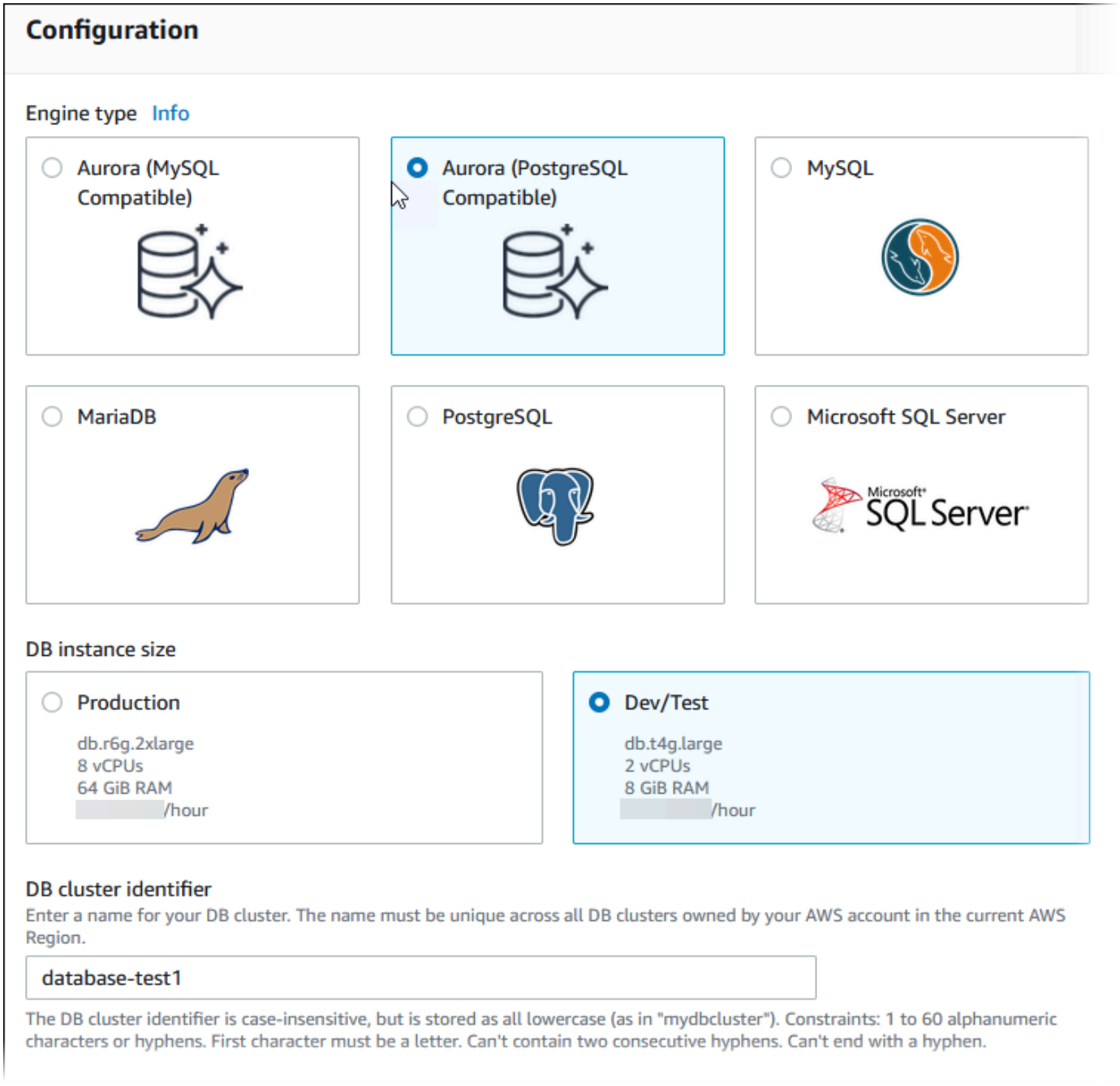
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la consola de Amazon RDS, elija la Región de AWS en la que desea crear el clúster de base de datos.
3. En el panel de navegación, elija Databases (Bases de datos).
4. Seleccione Crear base de datos y asegúrese de que la opción Creación sencilla esté seleccionada.



5. En Configuración, elija Aurora (compatible con PostgreSQL) en Tipo de motor.

6. En DB instance size (Tamaño de la instancia de base de datos), seleccione Dev/Test (Desarrollo/Prueba).
7. En Identificador de clúster de bases de datos, introduzca **`database-test1`**.

   La página Create database (Crear base de datos) debe ser similar a la siguiente imagen.



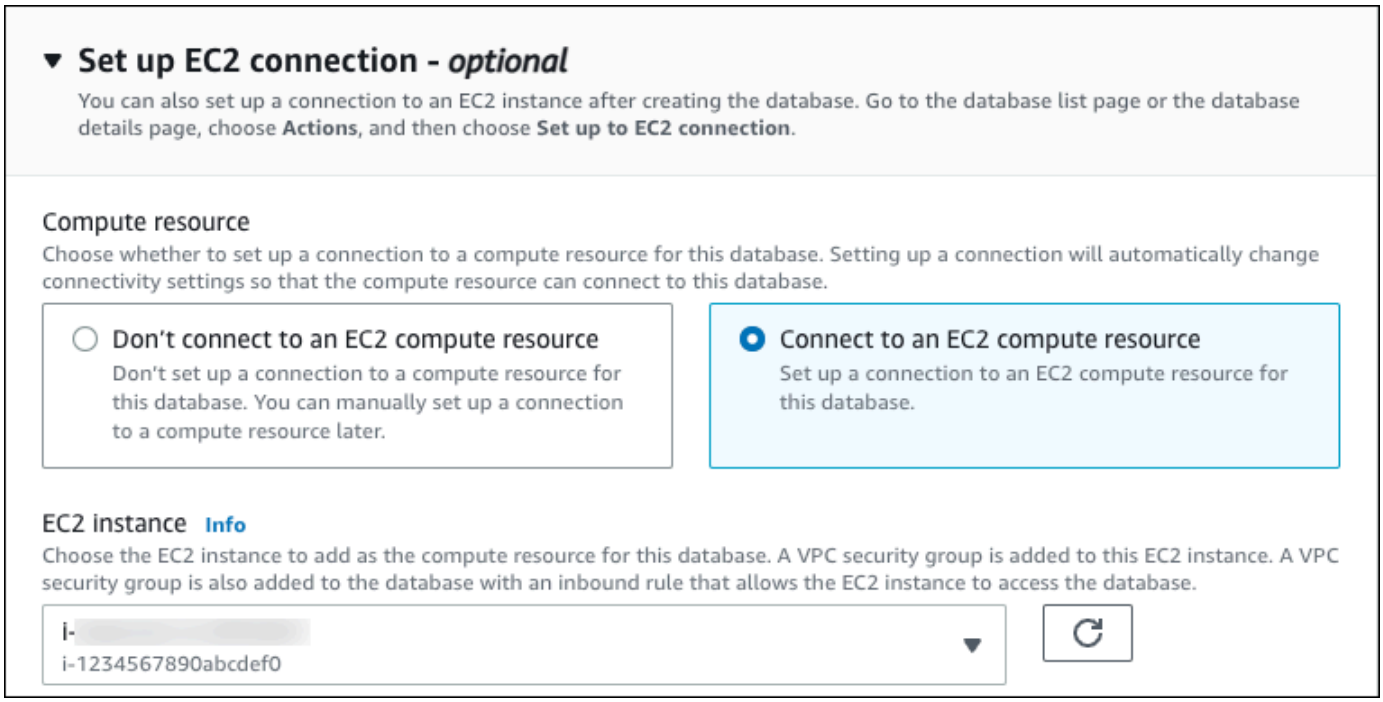
8. En Nombre de usuario maestro, introduzca un nombre para el usuario o deje el nombre predeterminado (**`postgres`**).

9. Para utilizar una contraseña maestra generada automáticamente para el clúster de base de datos, seleccione Generación automática de contraseña.

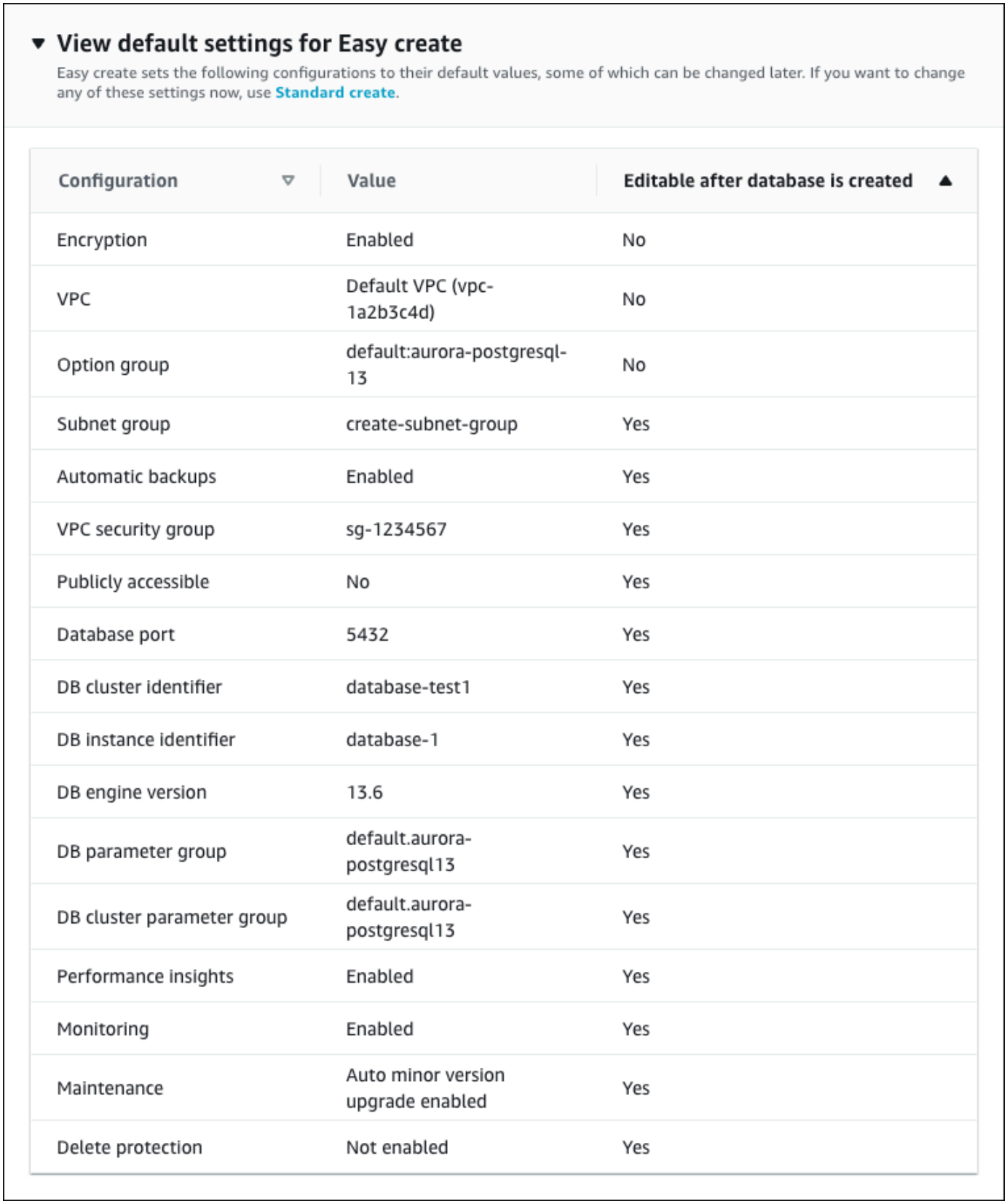
   Para introducir la contraseña maestra, asegúrese de desactivar la casilla Generación automática de contraseña y luego introduzca la misma contraseña en Contraseña maestra y Confirmar contraseña.

10. Para configurar una conexión con la instancia de EC2 que creó anteriormente, abra Configurar conexión a EC2 - (opcional).

    Seleccione Conectarse a un recurso informático de EC2. Elija la instancia de EC2 que ha creado anteriormente.



11. Abra la opción Ver la configuración predeterminada de la creación sencilla.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-postgresql-13	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	5432	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	13.6	Yes
DB parameter group	default.aurora-postgresql13	Yes
DB cluster parameter group	default.aurora-postgresql13	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Puede examinar la configuración predeterminada utilizada con Easy create (Creación sencilla). La columna Editable después de crear la base de datos muestra las opciones que puede cambiar después de crear la base de datos.

- Si una configuración tiene No en esa columna y desea una configuración diferente, puede usar Creación estándar para crear el clúster de base de datos.
- Si una configuración tiene Sí en esa columna y desea una configuración diferente, puede utilizar Creación estándar para crear el clúster de base de datos o modificar el clúster de base de datos después de crearlo para cambiar la configuración.

12. Elija Creación de base de datos.

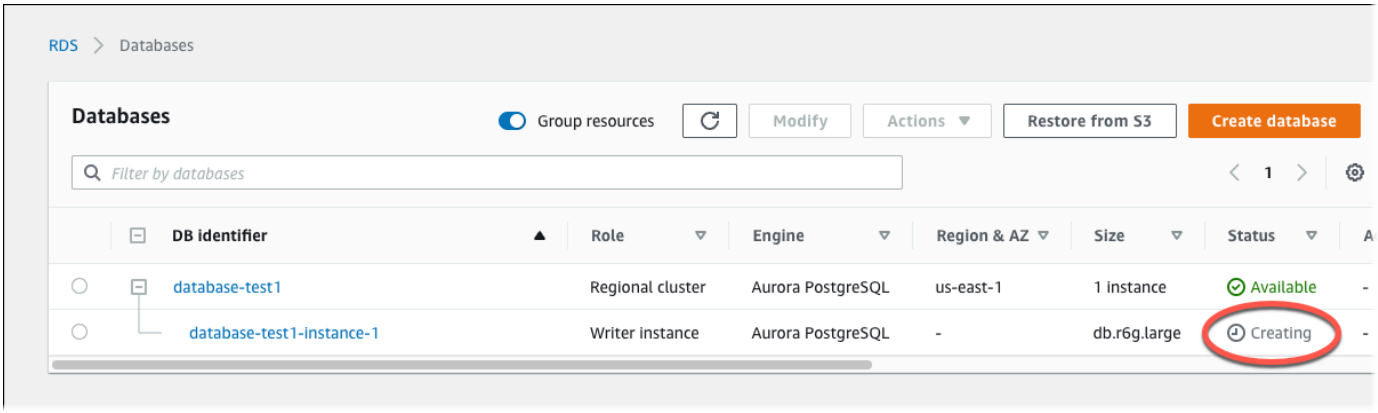
    Para consultar la contraseña y el nombre de usuario maestros del clúster de base de datos, seleccione Ver detalles de credenciales.

    Puede utilizar la contraseña y el nombre de usuario que aparecen para conectarse al clúster de base de datos como el usuario maestro.

    > **Important**
    >
    > No puede ver la contraseña de usuario maestro de nuevo. Si no la registra, es posible que tenga que cambiarla.
    > Si tiene que cambiar la contraseña de usuario maestro después de que el clúster de base de datos esté disponible, puede modificar el clúster de base de datos para ello. Para obtener más información sobre la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

13. En la lista Bases de datos, seleccione el nombre del nuevo clúster de base de datos de PostgreSQL para ver sus detalles.

    La instancia de escritor tiene el estado Creando hasta que el clúster de base de datos está listo para usarse.



Cuando el estado de la instancia de escritor cambie a Disponible, puede conectarse al clúster de la base de datos. Dependiendo de la clase de instancia de base de datos y de la cantidad de almacenamiento, es posible que el nuevo clúster de base de datos tarde hasta 20 minutos en estar disponible.

(Opcional) Crear una VPC, una instancia EC2 y un clúster de Aurora PostgreSQL mediante CloudFormation

En lugar de utilizar la consola para crear la VPC, la instancia de EC2 y el clúster de base de datos de Aurora PostgreSQL, puede utilizar CloudFormation para aprovisionar recursos de AWS tratando la infraestructura como código. Para ayudarle a organizar sus recursos de AWS en unidades más pequeñas y fáciles de administrar, puede utilizar la funcionalidad de pila anidada de CloudFormation. Para obtener más información, consulte Creación de una pila en la consola CloudFormation y Uso de pilas anidadas.

> **Important**
>
> CloudFormation es gratuito, pero los recursos que CloudFormation crea están activos. Se le facturan las tarifas de uso estándar por estos recursos hasta que los finalice. Para obtener más información, consulte Precios de Amazon Aurora.

Para crear sus recursos con la consola CloudFormation, siga estos pasos:

- Paso 1: Descargar la plantilla de CloudFormation
- Paso 2: Configurar los recursos mediante CloudFormation

Descargar la plantilla de CloudFormation

Una plantilla de CloudFormation es un archivo de texto con formato JSON o YAML que contiene la información de configuración de los recursos que desea crear en la pila. Esta plantilla también crea una VPC y un host bastión para usted junto con el clúster de Aurora.

Para descargar el archivo de plantilla, abra el enlace Aurora PostgreSQL CloudFormation template.

En la página de Github, haga clic en el botón Descargar archivo sin procesar para guardar el archivo YAML de la plantilla.

Configurar los recursos mediante CloudFormation

> **Note**
>
> Antes de iniciar este proceso, asegúrese de tener un par de claves para una instancia EC2 en su Cuenta de AWS. Para obtener más información, consulte Pares de claves de Amazon EC2 e instancias Linux.

Al utilizar la plantilla de CloudFormation, debe seleccionar los parámetros correctos para asegurarse de que los recursos se crean correctamente. Siga los pasos que se indican a continuación:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Elija Crear pila.
3. En la sección Especificar la plantilla, seleccione Cargar un archivo de plantilla desde el ordenador y Siguiente.
4. En la página Especificar detalles de la pila, introduzca los siguientes parámetros:
   a. Establezca el nombre de la pila en AurPostgreSQLTestStack.
   b. En Parámetros, defina las zonas de disponibilidad seleccionando dos zonas de disponibilidad.
   c. En Configuración de host bastión de Linux, en Nombre de la clave, seleccione un par de claves para iniciar sesión en su instancia de EC2.
   d. En los ajustes de Configuración de host bastión de Linux, ponga el rango de IP permitido en su dirección IP. Para conectarse a las instancias de EC2 de su VPC mediante Secure Shell (SSH), determine su dirección IP pública mediante el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es 192.0.2.1/32.

> **Warning**
>
> Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias de EC2 públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias de EC2 mediante SSH.

e. En Configuración general de la base de datos, defina la clase de instancia de base de datos en db.t4g.large.

f. Ponga el Nombre de la base de datos en **`database-test1`**.

g. En Nombre de usuario maestro, introduzca un nombre para el usuario maestro.

h. Ponga Administrar contraseña de usuario maestro de base de datos con Secrets Manager en `false` para este tutorial.

i. En Contraseña de base de datos, ponga la contraseña que desee. Recuerde esta contraseña para poder ver los pasos adicionales del tutorial.

j. Ponga Implementación multi-AZ en `false`.

k. Deje el resto de la configuración con los valores predeterminados. Haga clic en Siguiente para continuar.

5. En la página Configurar opciones de pila, deje todas las opciones predeterminadas. Haga clic en Siguiente para continuar.
6. En la página Revisar la pila, seleccione Enviar después de comprobar las opciones de base de datos y de host bastión de Linux.

Una vez finalizado el proceso de creación de la pila, visualice las pilas con los nombres BastionStack y APGNS para anotar la información que necesita para conectarse a la base de datos. Para obtener más información, consulte Viewing CloudFormation stack data and resources on the Consola de administración de AWS.

Paso 3: conectarse a un clúster de base de datos de Aurora PostgreSQL

Puede usar cualquier aplicación cliente PostgreSQL estándar para conectarse al clúster de base de datos. En este ejemplo, se conecta a un clúster de base de datos de PostgreSQL mediante el cliente de línea de comandos psql.

Para conectarse al clúster de base de datos de Aurora PostgreSQL

1. Busque el punto de conexión (nombre de DNS) y el número de puerto de la instancia de escritor de su clúster de base de datos.

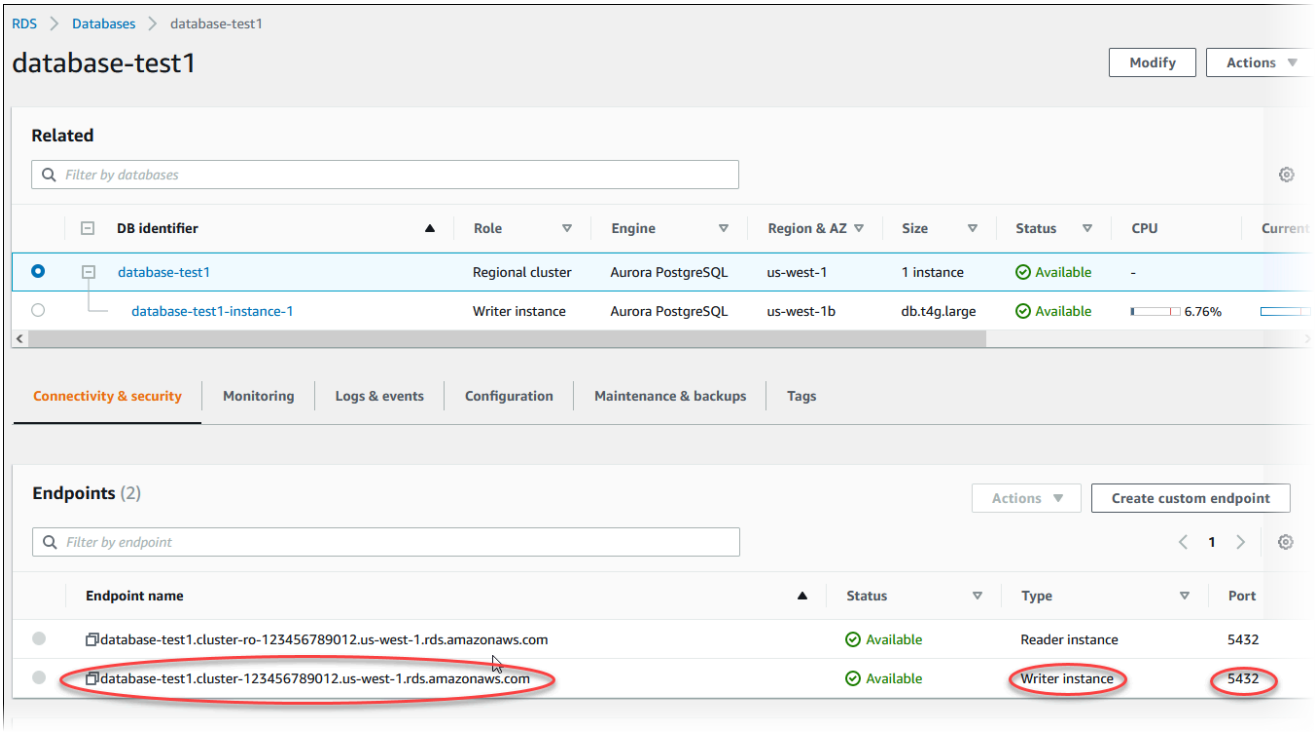
   a. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

   b. En la esquina superior derecha de la consola de Amazon RDS, elija la Región de AWS del clúster de base de datos.

   c. En el panel de navegación, elija Databases (Bases de datos).

   d. Seleccione el nombre del clúster de base de datos de Aurora PostgreSQL para ver sus detalles.

   e. En la pestaña Conectividad y seguridad, copie el punto de conexión de la instancia de escritor. También anote el número de puerto. Necesita el punto de conexión y el número de puerto para conectarse al clúster de base de datos.



2. Conéctese a la instancia de EC2 que ha creado anteriormente siguiendo los pasos que se indican en Conexión con la instancia de Linux en la Guía del usuario de Amazon EC2.

   Le recomendamos que se conecte a la instancia de EC2 mediante SSH. Si la utilidad de cliente SSH está instalada en Windows, Linux o Mac, puede conectarse a la instancia con el siguiente formato de comando:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Por ejemplo, suponga que `ec2-database-connect-key-pair.pem` está almacenado en `/dir1` en Linux y que el DNS IPv4 público de su instancia de EC2 es `ec2-12-345-678-90.compute-1.amazonaws.com`. Su comando SSH tendría el siguiente aspecto:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. Obtenga las correcciones de errores y las actualizaciones de seguridad más recientes actualizando el software en su instancia de EC2. Para ello, utilice el siguiente comando.

> **Note**
>
> La opción `-y` instala las actualizaciones sin necesidad de confirmación. Para examinar las actualizaciones antes de la instalación, omita esta opción.

```
sudo dnf update -y
```

4. Instale el cliente de línea de comandos psql desde PostgreSQL en Amazon Linux 2023 con el siguiente comando:

```
sudo dnf install postgresql15
```

5. Conéctese al clúster de base de datos de Aurora PostgreSQL. Por ejemplo, introduzca el siguiente comando. Esta acción le permite conectarse al clúster de base de datos de Aurora PostgreSQL mediante el cliente psql.

   Sustituya el punto de conexión de la instancia de escritor por *`endpoint`*, sustituya el nombre de la base de datos `--dbname` a la que quiera conectarse por *`postgres`* y sustituya el nombre de usuario maestro que utilizó por *`postgres`*. Proporcione la contraseña maestra que utilizó cuando se le solicite una contraseña.

```
psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
```

Una vez especificada la contraseña del usuario, debería ver un resultado similar al siguiente.

```
psql (14.3, server 14.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

Para obtener más información acerca de la conexión a un clúster de base de datos de Aurora PostgreSQL, consulte Conexión a un clúster de base de datos Amazon Aurora PostgreSQL. Si no puede conectarse al clúster de base de datos, consulte No puede conectarse a la instancia de base de datos de Amazon RDS.

Por motivos de seguridad, se recomienda utilizar conexiones cifradas. Utilice solo una conexión PostgreSQL sin cifrar cuando el cliente y el servidor están en la misma VPC y la red es de confianza. Para obtener información sobre el uso de de conexiones cifradas, consulte . Protección de los datos de Aurora PostgreSQL con SSL/TLS.

6. Ejecutar comandos SQL.

   Por ejemplo, el siguiente comando de SQL muestra la fecha y la hora actuales:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Paso 4: eliminar la instancia de EC2 y el clúster de base de datos

Después de conectarse y explorar la instancia de EC2 de muestra y el clúster de base de datos que creó, elimínelos para que no le sigan cobrando por ellos.

Si ha utilizado CloudFormation para crear recursos, omita este paso y vaya al siguiente.

Para eliminar la instancia de EC2

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En el panel de navegación, seleccione Instances (Instancias).
3. Seleccione la instancia de EC2 y elija Estado de la instancia y Terminar instancia.

4. Cuando se le indique que confirme, elija Terminar.

Para obtener más información sobre la eliminación de una instancia de EC2, consulte Terminar la instancia en la Guía del usuario de Amazon EC2.

Para eliminar un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos asociada al clúster de base de datos.
3. En Actions (Acciones), elija Delete (Eliminar).
4. Elija Eliminar.

   Después de eliminar todas las instancias de base de datos asociadas al clúster de base de datos, se elimina automáticamente el clúster de base de datos .

(Opcional) Eliminar la instancia de EC2 y el clúster de base de datos creados con CloudFormation

Si ha utilizado CloudFormation para crear recursos, elimine la pila de CloudFormation después de conectarse a la instancia de EC2 y el clúster de base de datos de muestra y de explorarlos; de este modo, ya no se le cobrará por ellos.

Para eliminar los recursos de CloudFormation

1. Abra la consola de CloudFormation.
2. En la página Pilas de la consola de CloudFormation, seleccione la pila raíz (la pila sin el nombre VPCStack, BastionStack o APGNS).
3. Elija Eliminar.
4. Cuando se le pida confirmación, seleccione Eliminar pila.

Para obtener información sobre cómo eliminar una pila en CloudFormation, consulte Eliminación de una pila en la consola de CloudFormation, en la Guía del usuario de AWS CloudFormation.

(Opcional) Conecte el clúster de base de datos a una función de Lambda

También puede conectar el clúster de base de datos de Aurora PostgreSQL a un recurso de computación sin servidor de Lambda. Las funciones de Lambda permiten ejecutar código sin aprovisionar ni administrar la infraestructura. Una función de Lambda también permite responder automáticamente a las solicitudes de ejecución de código a cualquier escala, desde una docena de eventos al día hasta cientos de eventos por segundo. Para obtener más información, consulte Conexión automática de una función de Lambda y un clúster de base de datos de Aurora.

Explicación: crear un servidor web y un clúster de base de datos de Amazon Aurora

Este tutorial le ayuda a instalar un servidor web Apache con PHP y a crear una base de datos de MariaDB, MySQL o PostgreSQL. El servidor web se ejecuta en una instancia de Amazon EC2 mediante Amazon Linux 2023 y puede elegir entre un clúster de base de datos de Aurora MySQL o Aurora PostgreSQL. Tanto la instancia de Amazon EC2 como el clúster de base de datos se ejecutan en una nube virtual privada (VPC) basada en el servicio Amazon VPC.

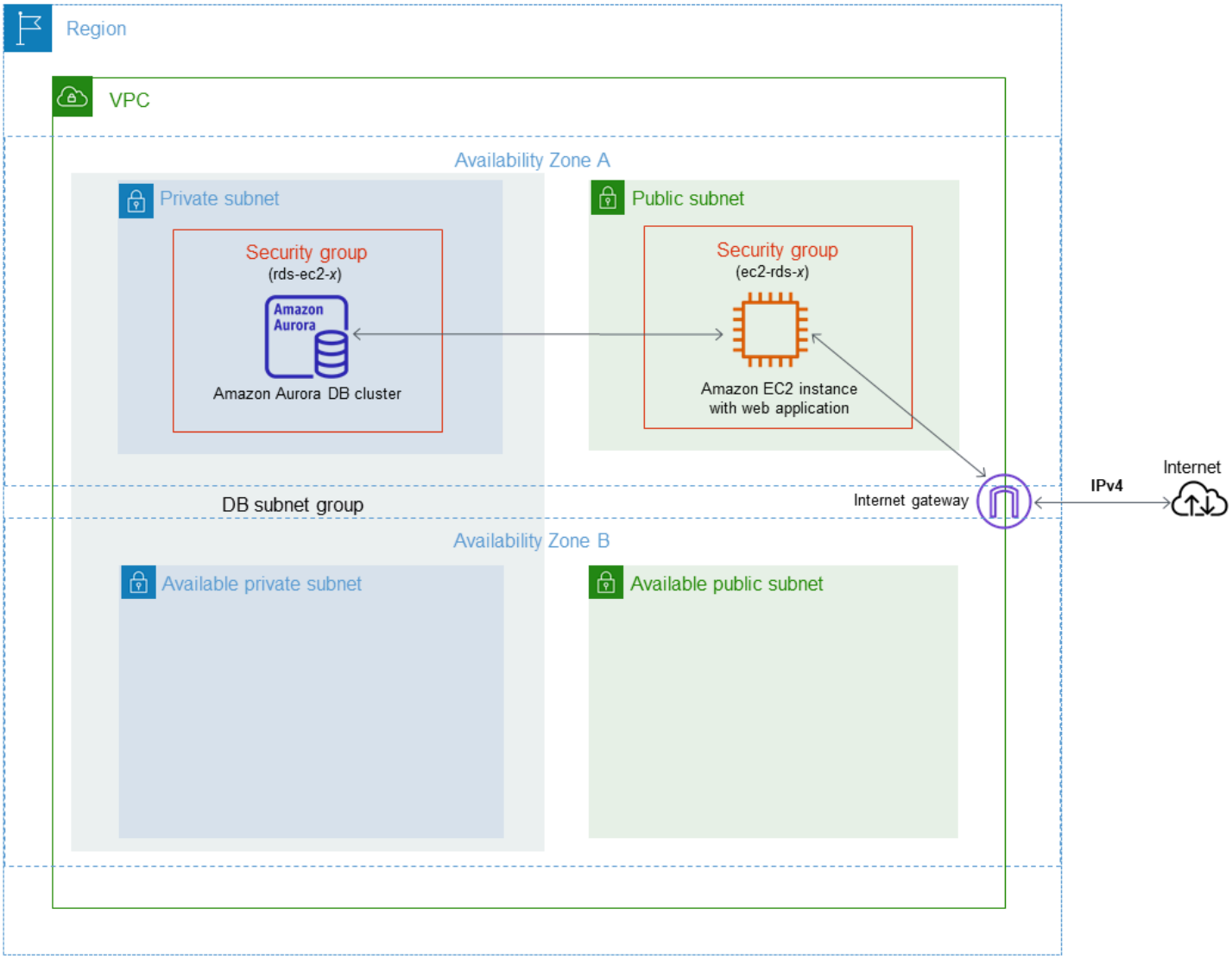
> **Important**
>
> La creación de una cuenta de AWS no supone ningún costo. No obstante, al completar este tutorial, puede incurrir en costos por los recursos de AWS que utilice. Puede eliminar estos recursos después de completar el tutorial si ya no son necesarios.

> **Note**
>
> Este tutorial funciona con Amazon Linux 2023 y podría no funcionar con otras versiones de Linux.

En la siguiente explicación, cree una instancia EC2 que utilice la VPC, subredes y el grupo de seguridad predeterminados para la Cuenta de AWS. En este tutorial, se muestra cómo crear el clúster de base de datos y configurar automáticamente la conectividad con la instancia EC2 que creó. A continuación, el tutorial muestra cómo instalar el servidor web en la instancia EC2. Conecte el servidor web a su clúster de base de datos en la VPC con el punto de conexión del escritor de clúster de base de datos.

1. Lanzamiento de una instancia EC2 para conectarse con el clúster de base de datos
2. Crear un clúster de base de datos de Amazon Aurora
3. Instalación de un servidor web en la instancia de EC2

El siguiente diagrama muestra la configuración cuando el tutorial se completa.



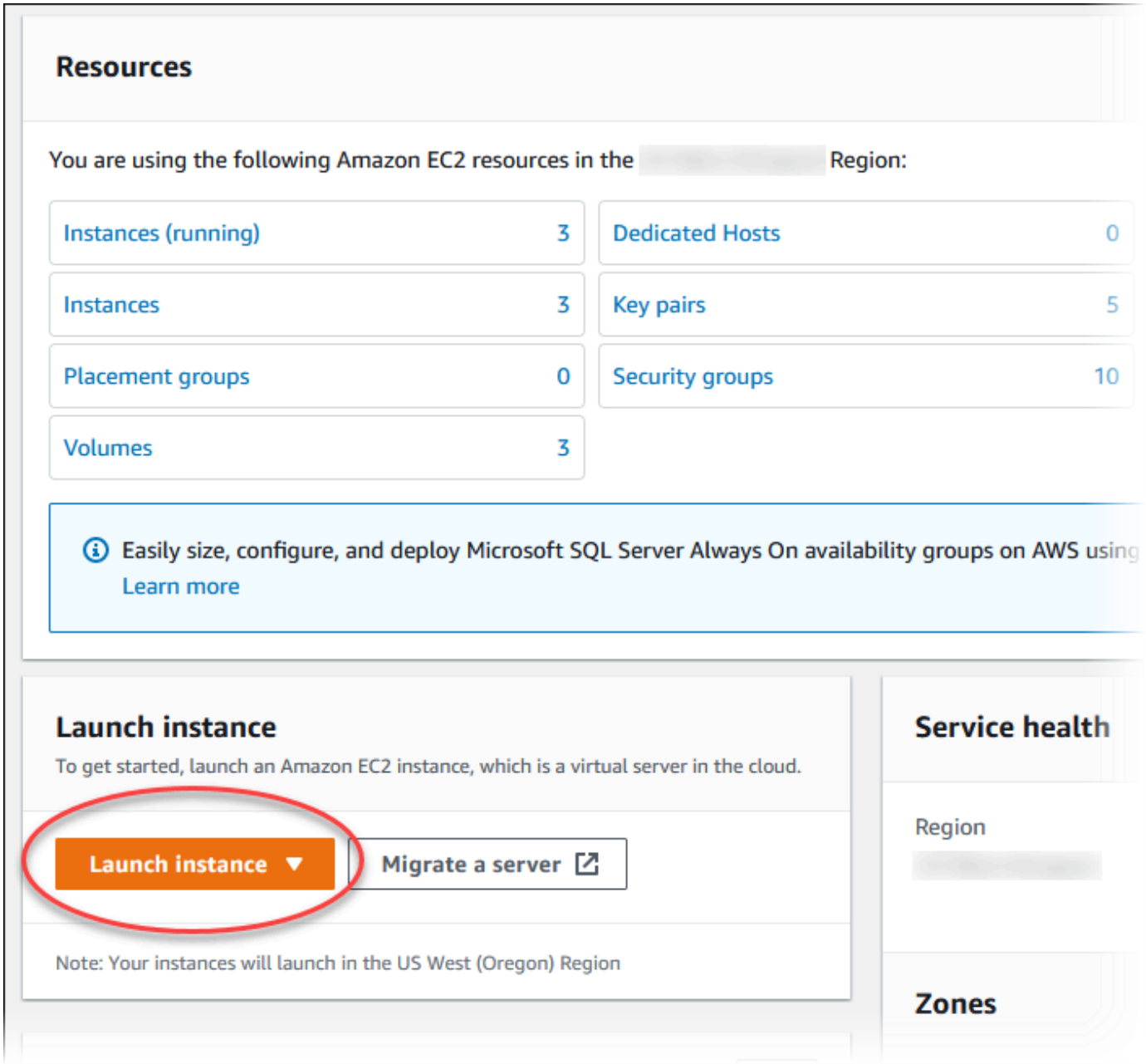
> **Note**
>
> Tras completar el tutorial, habrá una subred pública y una privada en cada zona de disponibilidad de la VPC. En este tutorial, se usa la VPC predeterminada para Cuenta de AWS que configura de forma automática la conectividad entre la instancia EC2 y el clúster de base de datos. Si prefiere configurar una nueva VPC para este escenario, complete las tareas de Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

Lanzamiento de una instancia EC2 para conectarse con el clúster de base de datos

Cree una instancia Amazon EC2 en la subred pública de la VPC.

Para lanzar una instancia de EC2

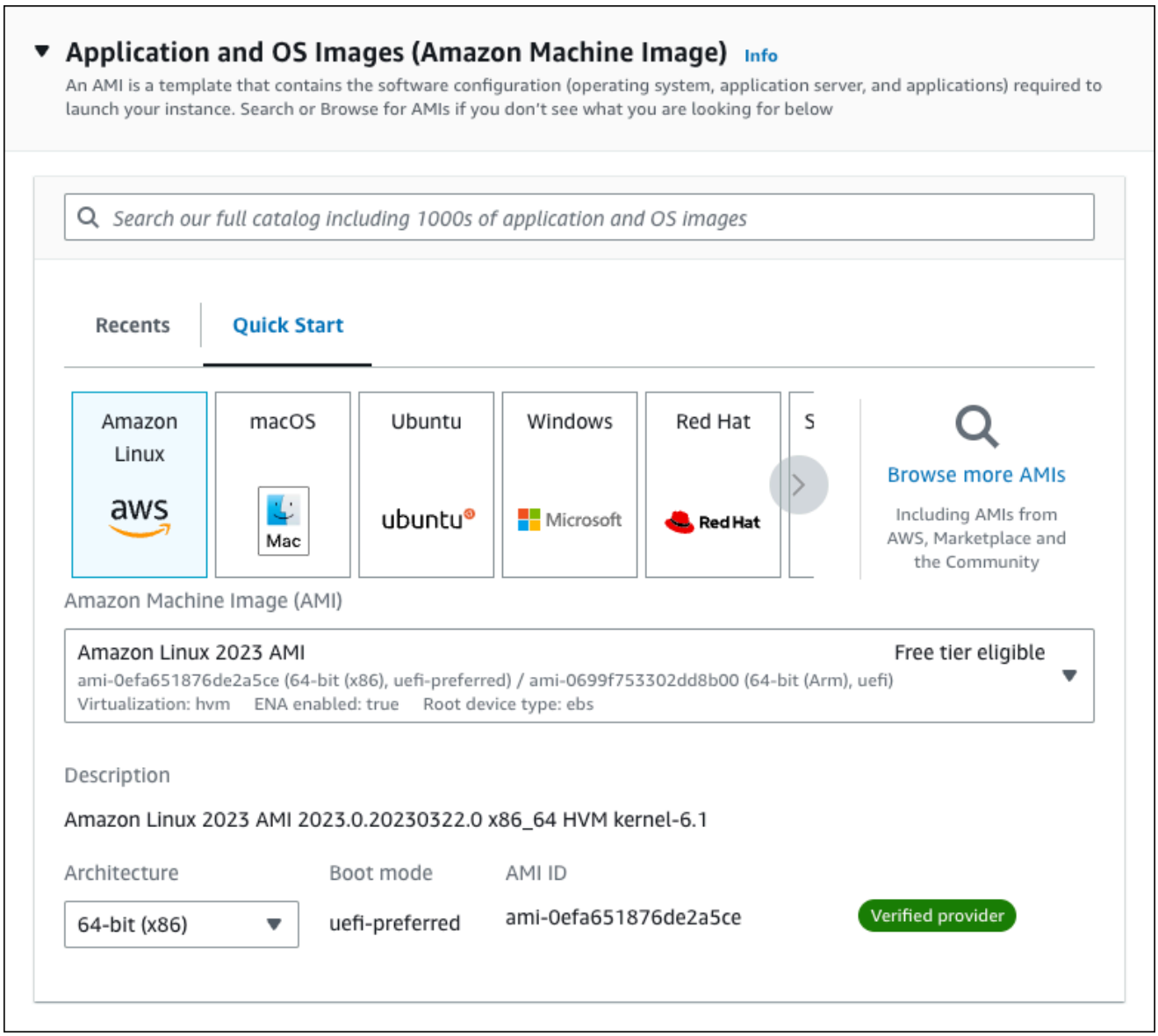
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En la esquina superior derecha de la Consola de administración de AWS, elija la Región de AWS en la que desea crear la instancia de EC2.
3. Elija Panel de EC2 y, a continuación, Lanzar instancia, como se muestra a continuación.



4. Elija los siguientes ajustes en la página Lanzar una instancia.

   a. En Name and tags (Nombre y etiquetas), en Name (Nombre), introduzca **`tutorial-ec2-instance-web-server`**.

   b. En Imágenes de aplicaciones y sistema operativo (Imagen de máquina de Amazon), elija Amazon Linux y, a continuación, AMI de Amazon Linux 2023. Mantenga los valores predeterminados para las demás opciones.



c. En Instance type (Tipo de instancia), elija t2.micro.

d. En Key pair (login) [Par de claves (inicio)], elija Key pair name (Nombre de par de claves) para utilizar un par de claves existente. Para crear un nuevo par de claves para la instancia de Amazon EC2, que se muestra a continuación, elija Create new key pair (Crear nuevo par de claves) y, a continuación, utilice la ventana Create key pair (Crear un par de claves).

   Para obtener más información sobre la creación de un nuevo par de claves, consulte Crear un par de claves en la Guía del usuario de Amazon EC2.

e. En Network settings (Configuración de red), defina estos valores y mantenga los ajustes predeterminados en los otros valores:

- En Allow SSH traffic from (Permitir el tráfico SSH desde), elija el origen de las conexiones SSH a la instancia de EC2.

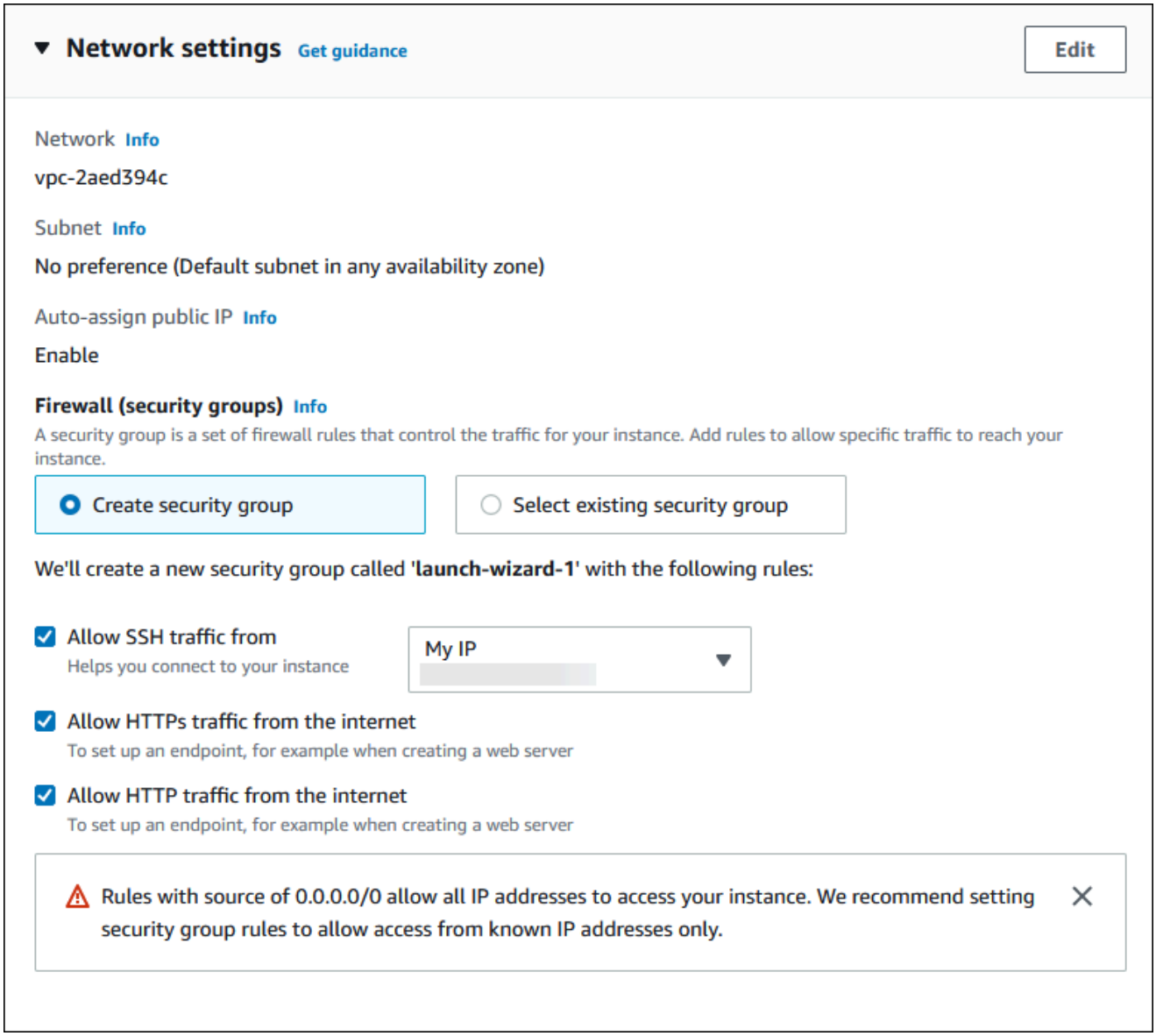
  Puede elegir My IP (Mi IP) si la dirección IP que se muestra es correcta para las conexiones SSH.

  De lo contrario, puede determinar la dirección IP que usará para conectarse a las instancias de EC2 en su VPC mediante Secure Shell (SSH). Para determinar su dirección IP pública, en una ventana o pestaña distinta del navegador, puede utilizar el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es `203.0.113.25/32`.

  En muchos casos, puede conectarse a través de un proveedor de servicios de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, asegúrese de identificar el rango de direcciones IP que utilizan los equipos cliente.

  > **Warning**
  >
  > Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias mediante SSH.

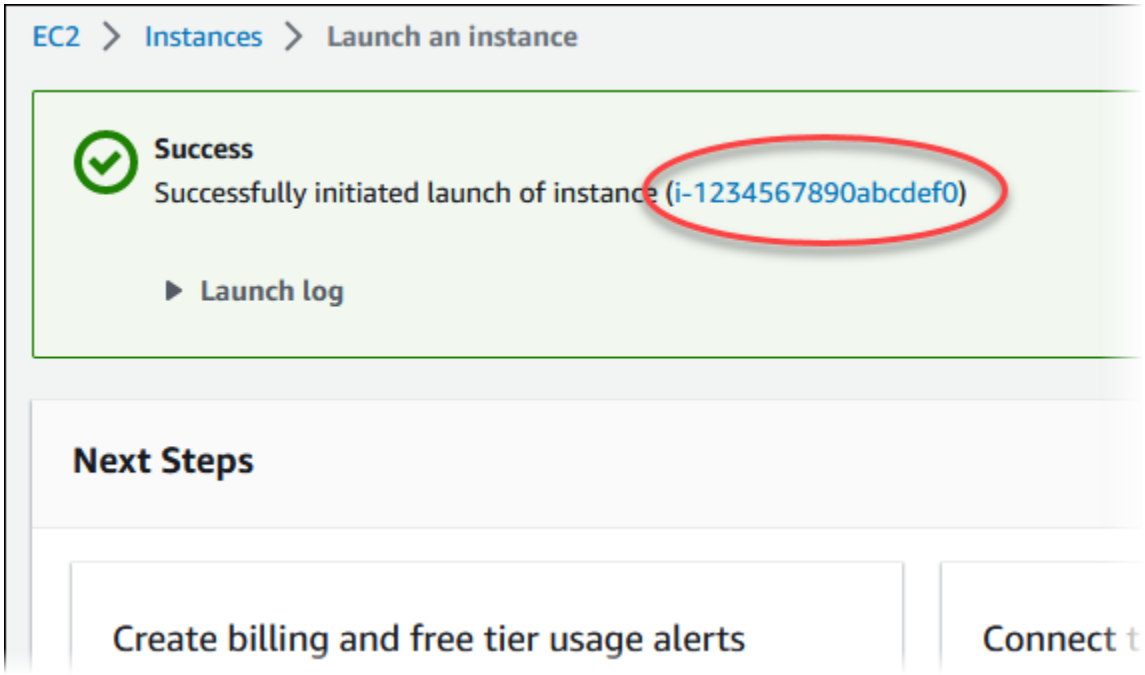
- Active Allow HTTPs traffic from the internet (Permitir el tráfico HTTP desde internet).
- Active Allow HTTPs traffic from the internet (Permitir el tráfico HTTP desde internet).



f. No cambie los valores predeterminados del resto de las secciones.

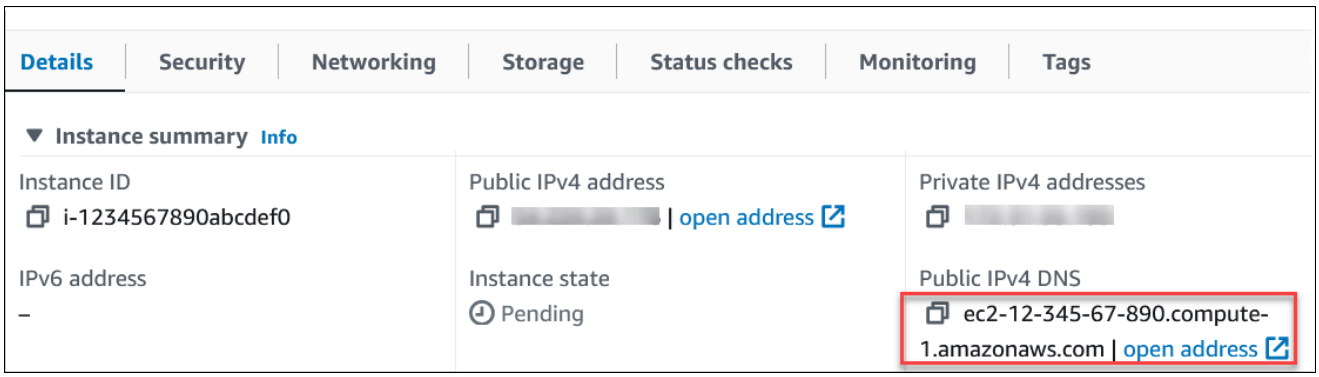
g. Revise un resumen de la configuración de su instancia en el panel Summary (Resumen); cuando haya terminado, elija Launch instance.

5. En la página Launch Status, que se muestra a continuación, anote el identificador de la nueva instancia de EC2, por ejemplo, `i-1234567890abcdef0`.

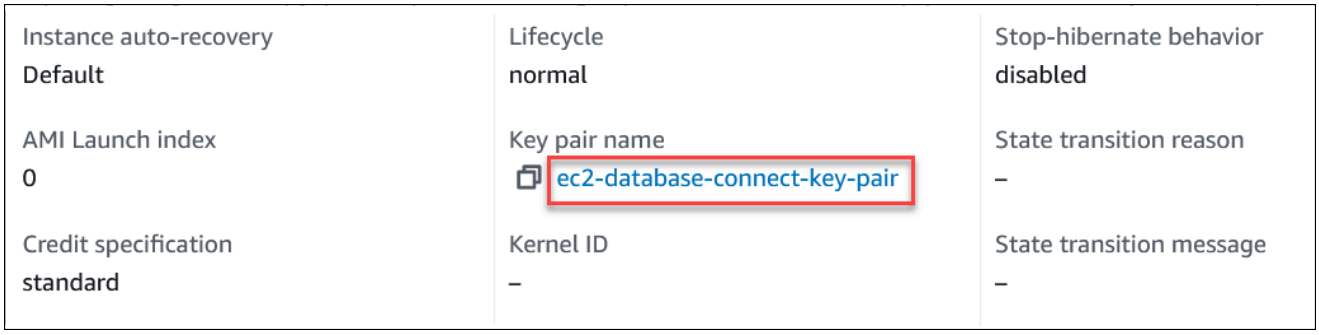


6. Elija el identificador de instancia de EC2 para abrir la lista de instancias de EC2 y, a continuación, seleccione su instancia de EC2.
7. En la pestaña Detalles, anote los siguientes valores, ya que los necesitará cuando se conecte mediante SSH:

   a. En Resumen de la instancia, anote el valor del DNS IPv4 público.



   b. En Detalles de la instancia, anote el valor de Nombre del par de claves.



8. Espere a que Instance Status (Estado de la instancia) se muestre como Running (En ejecución) antes de continuar.
9. Complete la Crear un clúster de base de datos de Amazon Aurora.

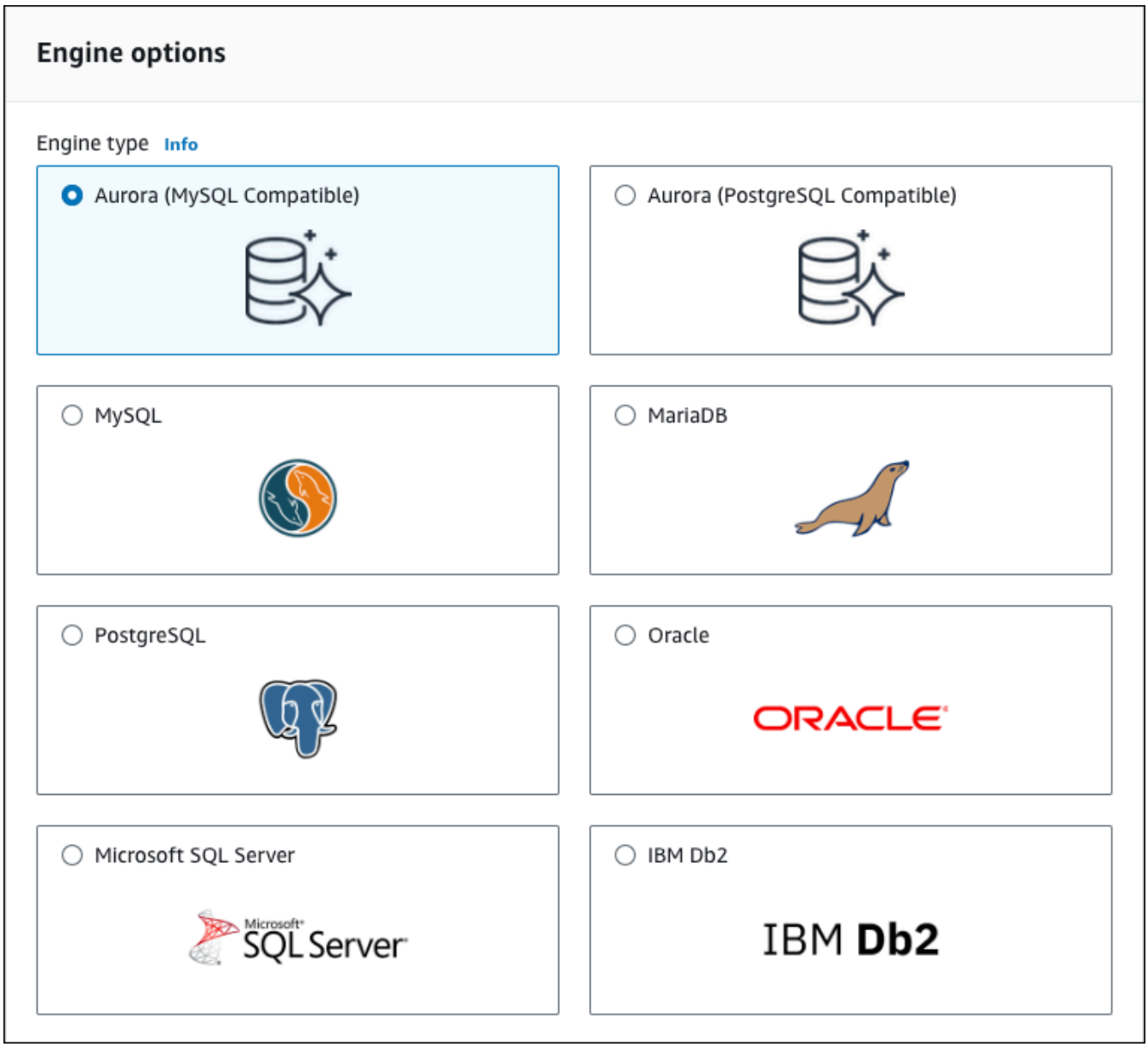
Crear un clúster de base de datos de Amazon Aurora

Cree un clúster de base de datos de Amazon Aurora MySQL o Aurora PostgreSQL que conserve los datos utilizados por una aplicación web.

Aurora MySQL

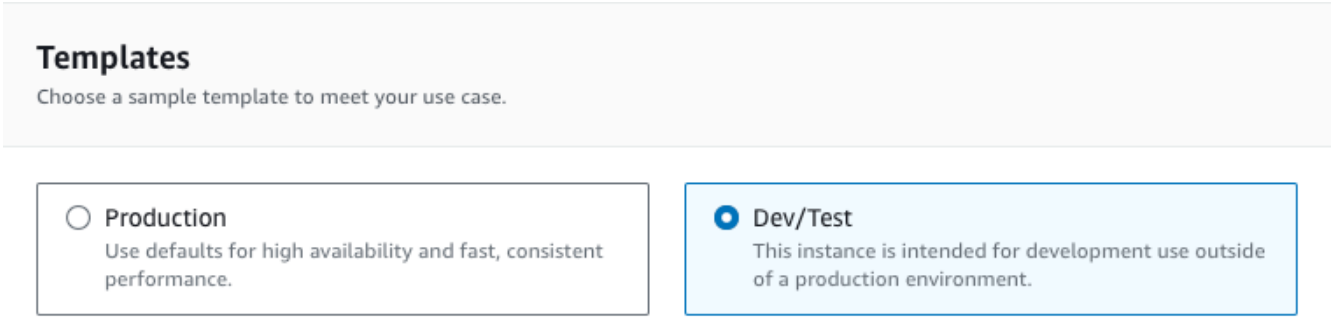
Para crear un clúster de base de datos de Aurora MySQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la Consola de administración de AWS, asegúrese de que la Región de AWS sea la misma que en la que creó la instancia EC2.
3. En el panel de navegación, seleccione Databases (Bases de datos).
4. Elija Create database (Crear base de datos).
5. En la página Crear base de datos, elija Creación estándar.
6. En Opciones del motor, elija Aurora (compatible con MySQL).



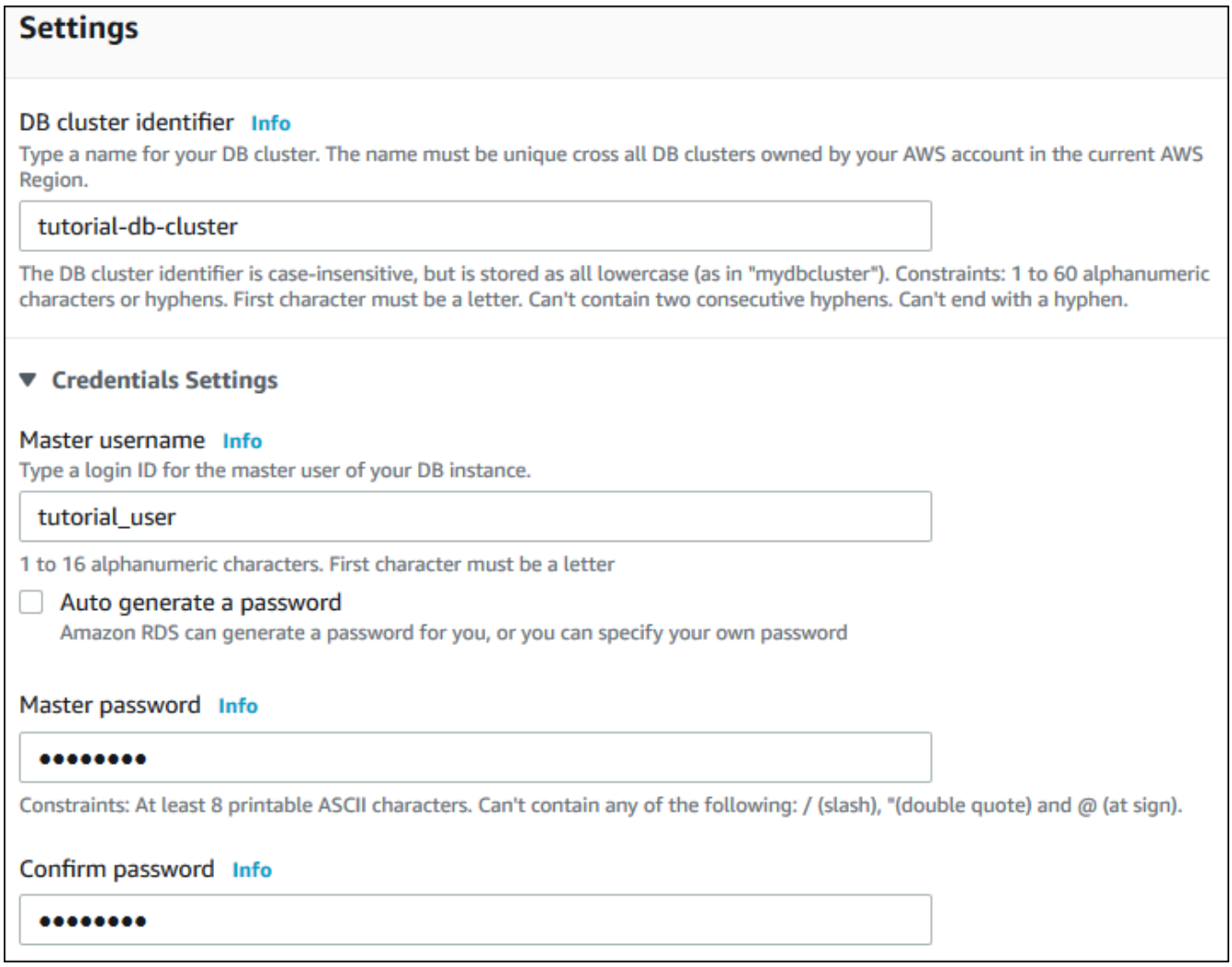
Mantenga los valores predeterminados para la versión y el resto de opciones del motor.

7. En la sección Templates (Plantillas), elija Desarrollo/Prueba.



8. En la sección Settings (Configuración), establezca los siguientes valores:

   - DB cluster identifier (Identificador de clúster de base de datos): escriba **`tutorial-db-cluster`**
   - Master username (Nombre de usuario maestro): escriba **`tutorial_user`**.
   - Auto generate a password (Generar una contraseña de forma automática): deje la opción desactivada.
   - Master password (Contraseña maestra): escriba una contraseña.
   - Confirm password (Confirmar contraseña):– vuelva a introducir la contraseña.

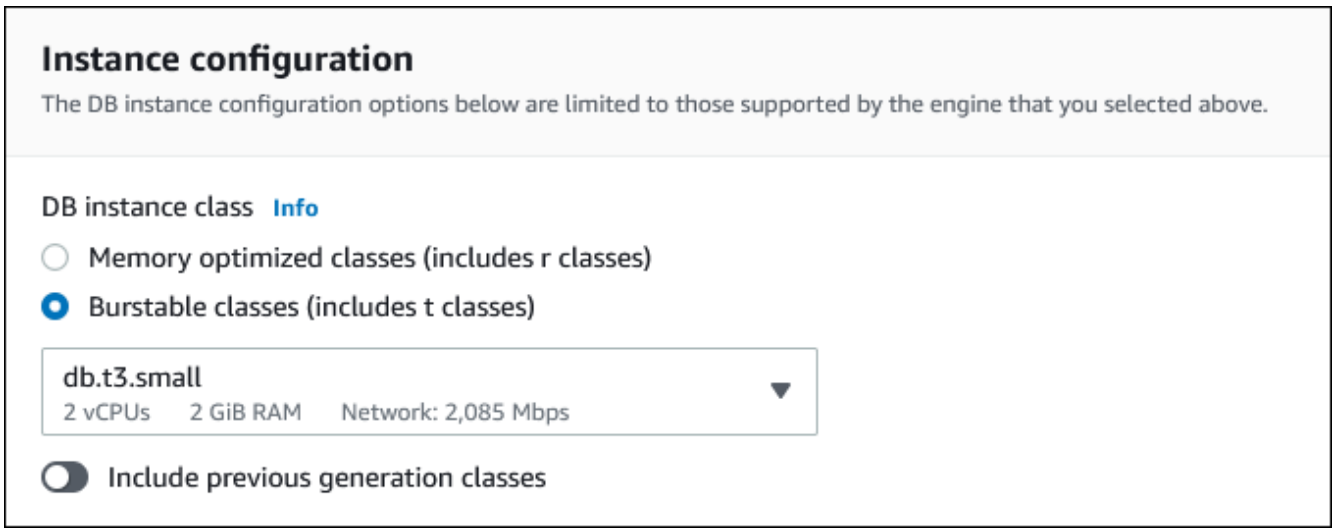


9. En la sección Instance configuration (Configuración de instancia), establezca estos valores:

   - Clases por ráfagas (incluye clases t)

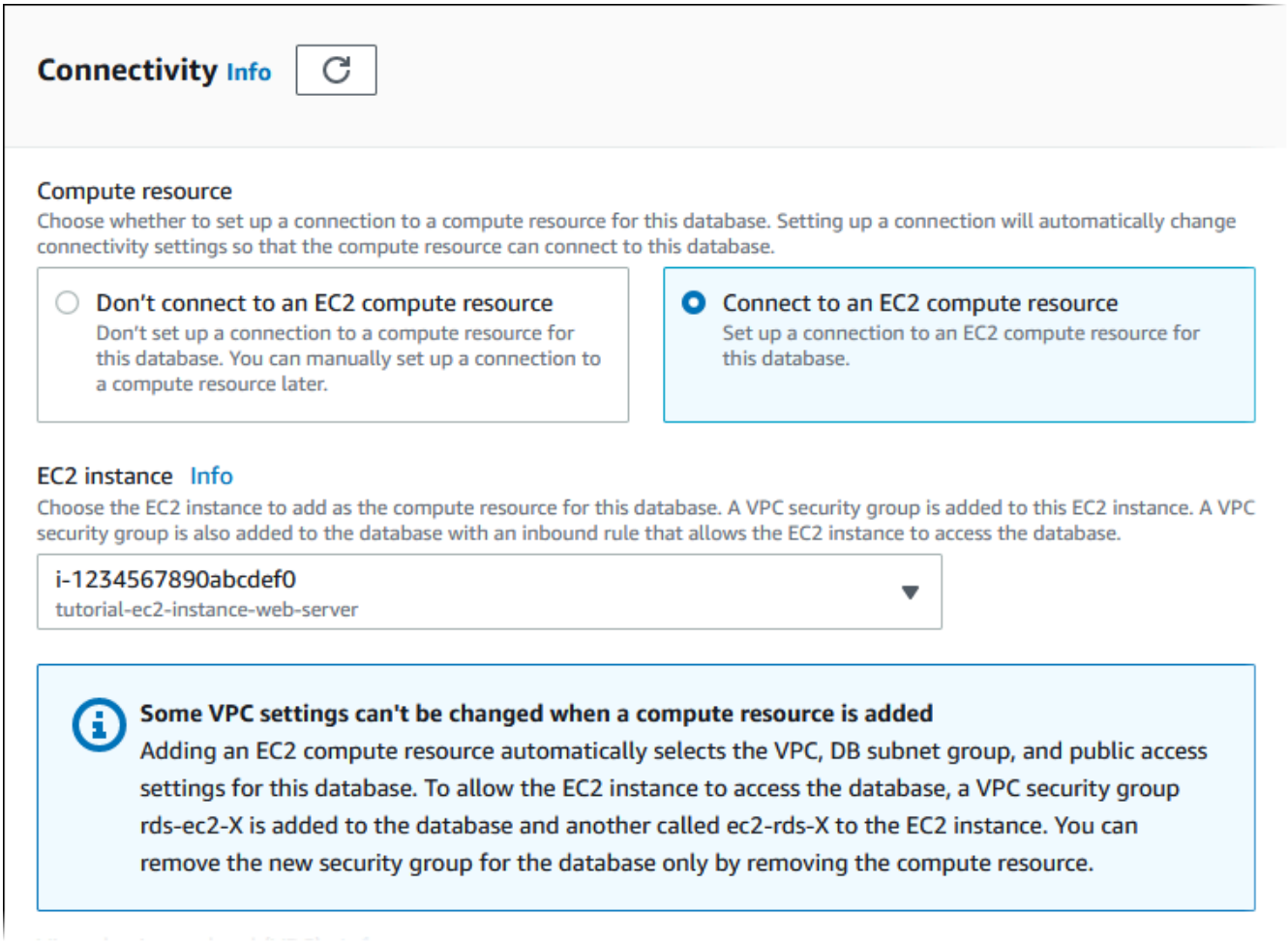
- db.t3.small o db.t3.medium

> **Note**
>
> Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.



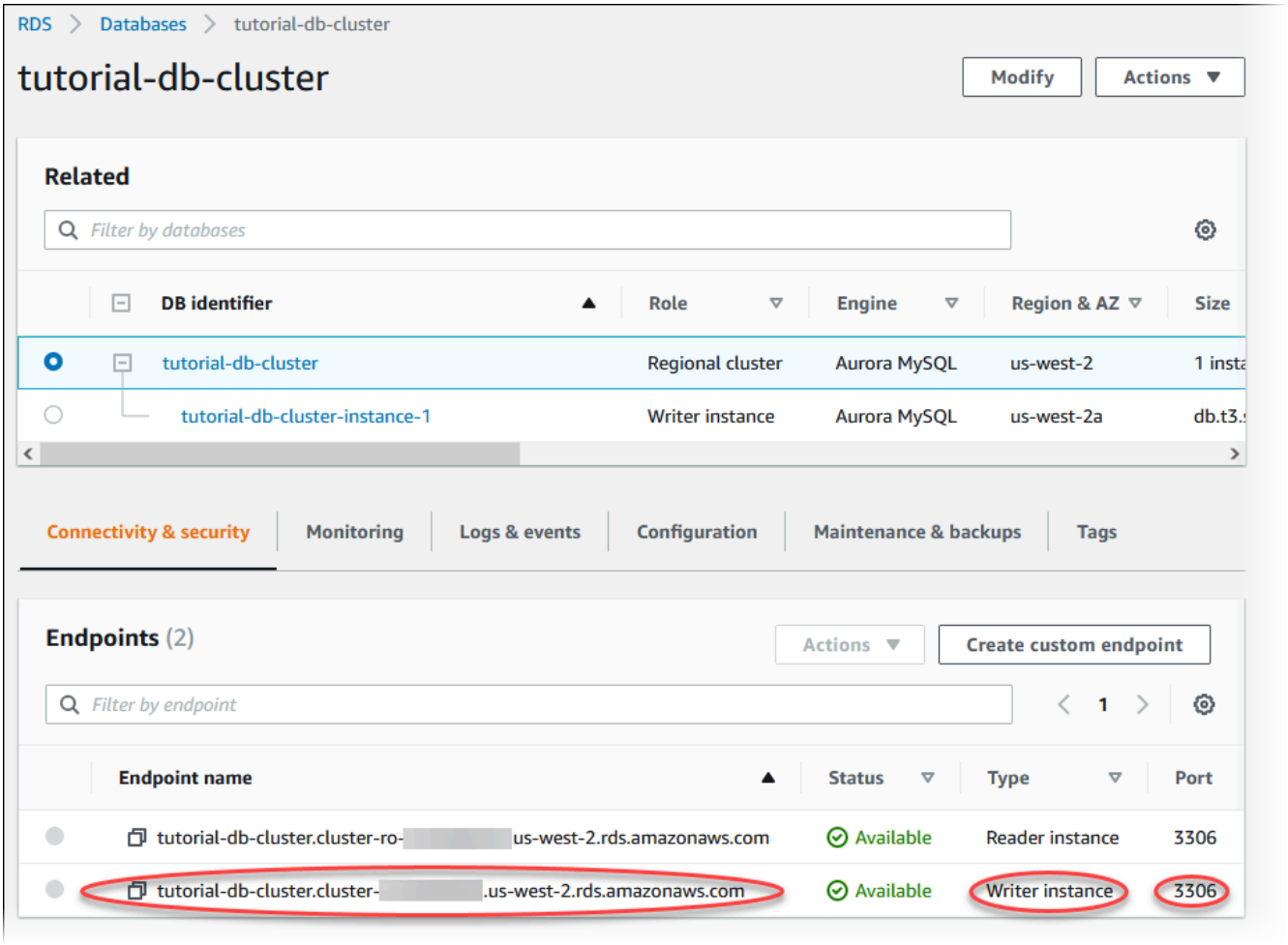
10. En la sección Availability and durability (Disponibilidad y durabilidad), utilice los valores predeterminados.
11. En la sección Connectivity (Conectividad), defina estos valores y mantenga los demás con sus valores predeterminados:

    - En Compute resource (Recurso informático), elija Connect to an EC2 compute resource (Conectar a un recurso informático de EC2).
    - En EC2 instance (Instancia EC2), elija la instancia de EC2 que creó anteriormente, como tutorial-ec2-instance-web-server.



12. Abra la sección Additional configuration (Configuración adicional) e introduzca **`sample`** para Initial database name (Nombre de la base de datos inicial) Mantenga la configuración predeterminada para el resto de las opciones.
13. Para crear el clúster de base de datos de Aurora MySQL, elija Crear base de datos.

    El nuevo clúster de base de datos aparece en la lista Bases de datos con el estado Creándose.
14. Espere a que el Estado de su nuevo clúster de base de datos se muestre como Disponible. A continuación, elija el nombre del clúster de base de datos para mostrar sus detalles.
15. En la sección Conectividad y seguridad, consulte el Punto de enlace y Puerto de la instancia de base de datos del escritor.



Anote el punto de enlace y el puerto de la instancia de base de datos del escritor. Utiliza esta información para conectar su servidor web al clúster de base de datos.

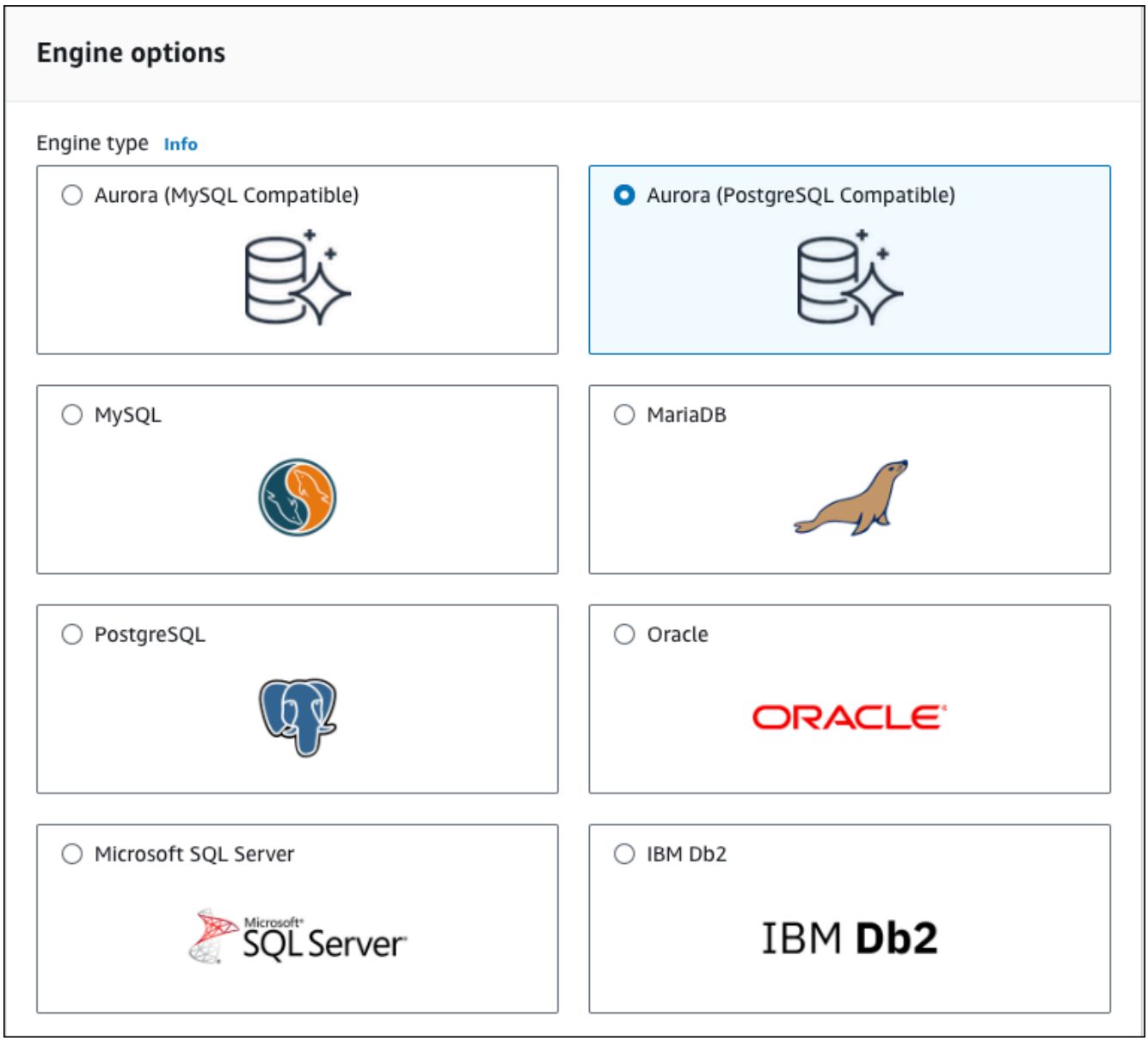
16. complet Instalación de un servidor web en la instancia de EC2.

Aurora PostgreSQL

Para crear un clúster de base de datos de Aurora PostgreSQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la Consola de administración de AWS, asegúrese de que la Región de AWS sea la misma que en la que creó la instancia EC2.
3. En el panel de navegación, seleccione Databases (Bases de datos).
4. Elija Create database (Crear base de datos).

5. En la página Crear base de datos, elija Creación estándar.
6. En Opciones de motor, elija Aurora (compatible con PostgreSQL).

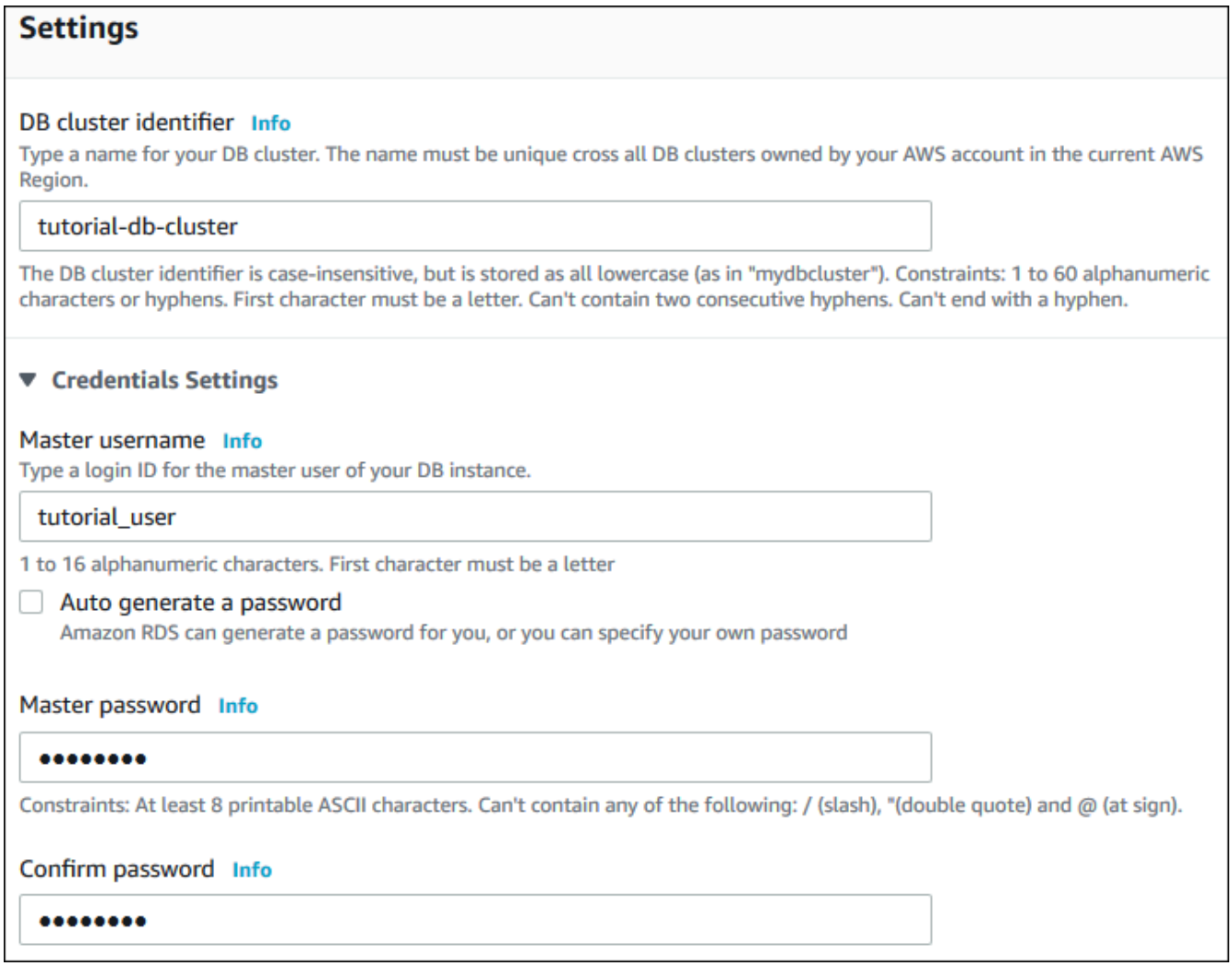


Mantenga los valores predeterminados para la versión y el resto de opciones del motor.

7. En la sección Templates (Plantillas), elija Desarrollo/Prueba.

8. En la sección Settings (Configuración), establezca los siguientes valores:

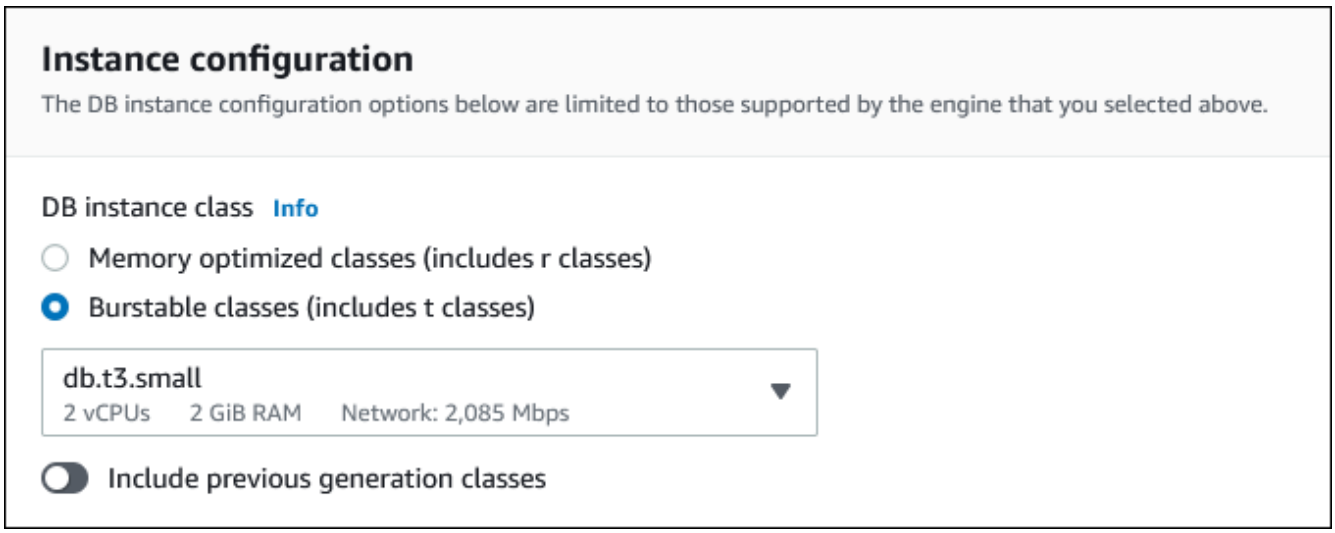
   - DB cluster identifier (Identificador de clúster de base de datos): escriba **`tutorial-db-cluster`**
   - Master username (Nombre de usuario maestro): escriba **`tutorial_user`**.
   - Auto generate a password (Generar una contraseña de forma automática): deje la opción desactivada.
   - Master password (Contraseña maestra): escriba una contraseña.
   - Confirm password (Confirmar contraseña):– vuelva a introducir la contraseña.



9. En la sección Instance configuration (Configuración de instancia), establezca estos valores:

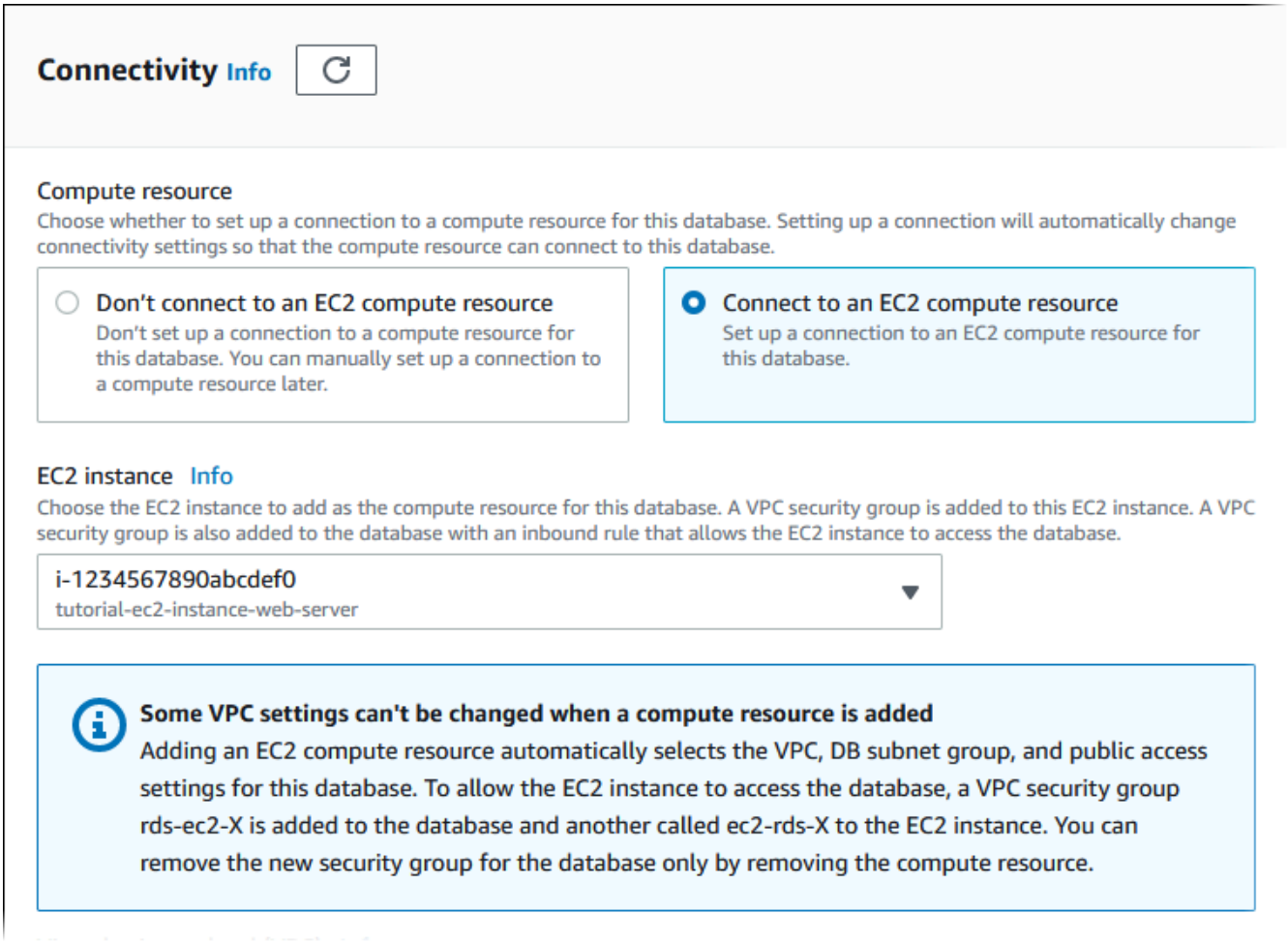
   - Clases por ráfagas (incluye clases t)
   - db.t3.small o db.t3.medium

> **Note**
>
> Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.



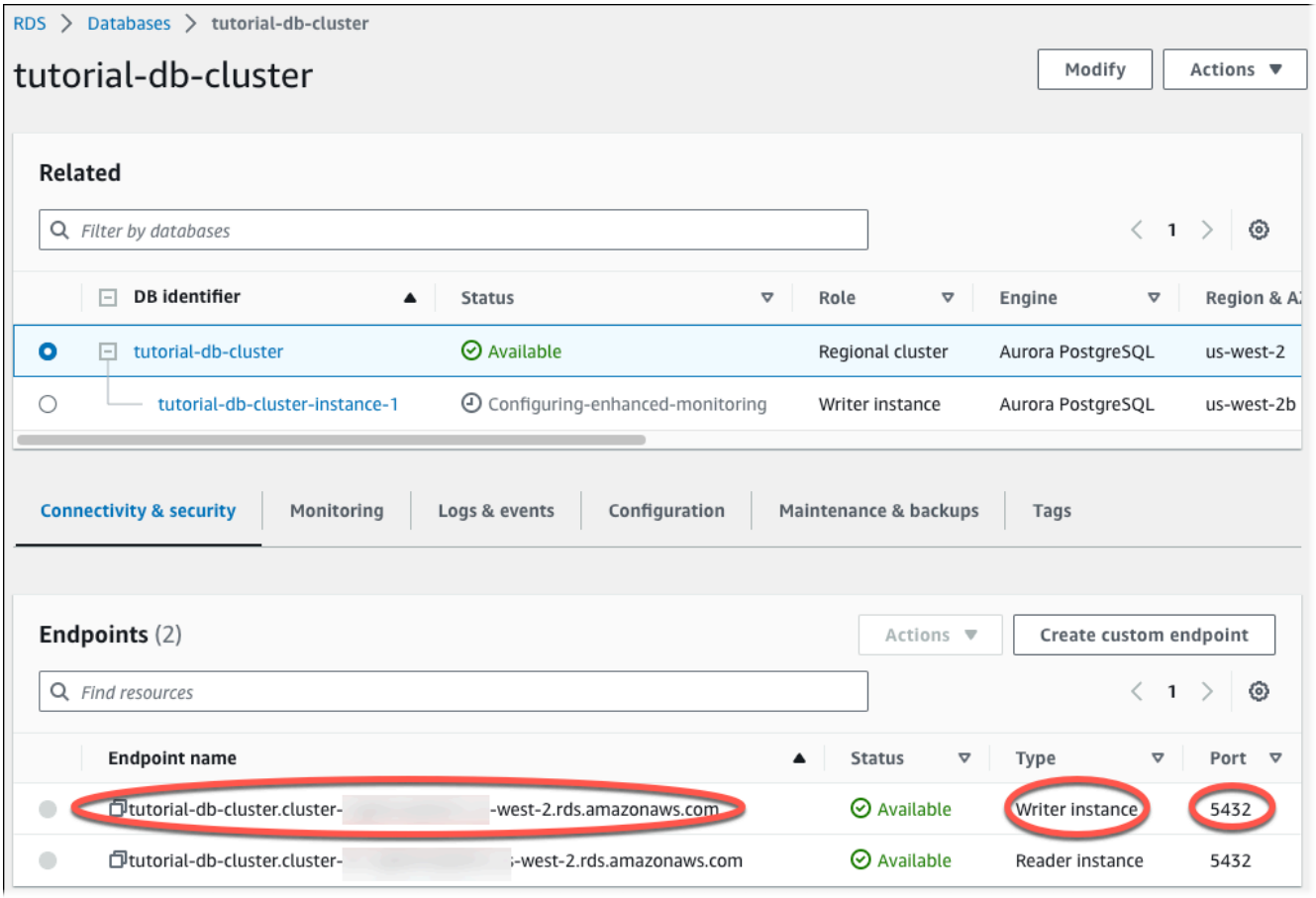
10. En la sección Availability and durability (Disponibilidad y durabilidad), utilice los valores predeterminados.
11. En la sección Connectivity (Conectividad), defina estos valores y mantenga los demás con sus valores predeterminados:

    - En Compute resource (Recurso informático), elija Connect to an EC2 compute resource (Conectar a un recurso informático de EC2).
    - En EC2 instance (Instancia EC2), elija la instancia de EC2 que creó anteriormente, como tutorial-ec2-instance-web-server.



12. Abra la sección Additional configuration (Configuración adicional) e introduzca **`sample`** para Initial database name (Nombre de la base de datos inicial) Mantenga la configuración predeterminada para el resto de las opciones.
13. Para crear el clúster de base de datos de Aurora PostgreSQL, elija Crear base de datos.

    El nuevo clúster de base de datos aparece en la lista Bases de datos con el estado Creándose.
14. Espere a que el Estado de su nuevo clúster de base de datos se muestre como Disponible. A continuación, elija el nombre del clúster de base de datos para mostrar sus detalles.
15. En la sección Conectividad y seguridad, consulte el Punto de enlace y Puerto de la instancia de base de datos del escritor.



Anote el punto de enlace y el puerto de la instancia de base de datos del escritor. Utiliza esta información para conectar su servidor web al clúster de base de datos.

16. complet Instalación de un servidor web en la instancia de EC2.

Instalación de un servidor web en la instancia de EC2

Instale un servidor web en una instancia de EC2 que creó en Lanzamiento de una instancia EC2 para conectarse con el clúster de base de datos. El servidor web se conecta al clúster de base de datos de Amazon Aurora que creó en Crear un clúster de base de datos de Amazon Aurora.

Instalación de un servidor web Apache con PHP y MariaDB

Conéctese a su instancia de EC2 e instale el servidor web.

Para conectarse a la instancia de EC2 e instalar el servidor web Apache con PHP

1. Conéctese a la instancia de EC2 que ha creado anteriormente siguiendo los pasos que se indican en Conexión con la instancia de Linux en la Guía del usuario de Amazon EC2.

Le recomendamos que se conecte a la instancia de EC2 mediante SSH. Si la utilidad de cliente SSH está instalada en Windows, Linux o Mac, puede conectarse a la instancia con el siguiente formato de comando:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Por ejemplo, suponga que `ec2-database-connect-key-pair.pem` está almacenado en `/dir1` en Linux y que el DNS IPv4 público de su instancia de EC2 es `ec2-12-345-678-90.compute-1.amazonaws.com`. Su comando SSH tendría el siguiente aspecto:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

2. Obtenga las correcciones de errores y las actualizaciones de seguridad más recientes actualizando el software en su instancia de EC2. Para ello, utilice el siguiente comando.

> **Note**
>
> La opción `-y` instala las actualizaciones sin necesidad de confirmación. Para examinar las actualizaciones antes de la instalación, omita esta opción.

```
sudo dnf update -y
```

3. Una vez completadas las actualizaciones, instale el servidor web Apache, PHP y el software de MariaDB o PostgreSQL con los siguientes comandos. Este comando instala varios paquetes de software y dependencias relacionadas al mismo tiempo.

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

   PostgreSQL

   ```
   sudo dnf install -y httpd php php-pgsql postgresql15
   ```

Si recibe un error, probablemente la instancia no se ha iniciado con una AMI de Amazon Linux 2023. Es posible que esté usando la AMI Amazon Linux 2 en su lugar. Puede ver la versión de Amazon Linux usando el comando siguiente:

```
cat /etc/system-release
```

Para obtener más información, consulte Actualización del software de instancia.

4. Inicie el servidor web mediante el comando que se muestra a continuación.

```
sudo systemctl start httpd
```

Puede probar que el servidor web esté correctamente instalado e iniciado. Para ello, escriba el nombre público del Sistema de nombres de dominio (DNS) de su instancia de EC2 en la barra de direcciones de un navegador web, por ejemplo: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. Si el servidor web se está ejecutando, se mostrará la página de prueba de Apache.

Si no ve la página de prueba de Apache, compruebe las reglas de entrada para el grupo de seguridad de VPC que creó en Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4). Asegúrese de que las reglas de entrada incluyan una que permita el acceso HTTP (puerto 80) a la dirección IP que utiliza para conectarse al servidor web.

> **Note**
>
> La página de prueba de Apache aparece únicamente cuando el directorio raíz de documentos está vac, `/var/www/html`. Después de añadir contenido al directorio raíz de documentos, el contenido aparece en la dirección DNS pública de la instancia de EC2. Antes de este punto, aparece en la página de prueba de Apache.

5. Configure el servidor web para que se inicie en cada arranque del sistema con el comando `systemctl`.

```
sudo systemctl enable httpd
```

Para permitir a `ec2-user` administrar archivos en el directorio raíz predeterminado del servidor web Apache, modifique los propietarios y los permisos del directorio `/var/www`. Existen muchas formas de realizar esta tarea. En este tutorial se añade el usuario `ec2-user` al grupo `apache`, se otorga al grupo `apache` la propiedad del directorio `/var/www` y se asignan permisos de escritura al grupo.

Para configurar los permisos de archivo para el servidor web Apache

1. Agregue el usuario `ec2-user` al grupo `apache`.

   ```
   sudo usermod -a -G apache ec2-user
   ```

2. Cierre la sesión para actualizar los permisos e incluir el nuevo grupo `apache`.

   ```
   exit
   ```

3. Inicie sesión nuevamente y compruebe que existe el grupo `apache` mediante el comando `groups`.

   ```
   groups
   ```

   El resultado tiene un aspecto similar al siguiente:

   ```
   ec2-user adm wheel apache systemd-journal
   ```

4. Cambie la propiedad de grupo del directorio `/var/www` y su contenido al grupo `apache`.

   ```
   sudo chown -R ec2-user:apache /var/www
   ```

5. Cambie los permisos del directorio `/var/www` y sus subdirectorios para añadir permisos de escritura de grupo y establecer el ID de grupo en los subdirectorios que se creen en el futuro.

   ```
   sudo chmod 2775 /var/www
   find /var/www -type d -exec sudo chmod 2775 {} \;
   ```

6. Cambie recursivamente los permisos de los archivos del directorio `/var/www` y sus subdirectorios para añadir permisos de escritura de grupo.

   ```
   find /var/www -type f -exec sudo chmod 0664 {} \;
   ```

Ahora `ec2-user` (y cualquier miembro futuro del grupo de `apache`) puede añadir, eliminar y editar archivos en la raíz de documentos de Apache. Esto le permite añadir contenido, como un sitio web estático o una aplicación PHP.

> **Note**
>
> Un servidor web que ejecuta el protocolo HTTP no proporciona seguridad de transporte de los datos que envía o recibe. Cuando se conecta a un servidor HTTP utilizando un navegador web, la mayor parte de la información es visible a cualquier acceso no autorizado en la ruta de la red. Esta información incluye las URL que visita, el contenido de las páginas web que recibe y el contenido (incluidas las contraseñas) de cualquier formulario HTML.
> La práctica recomendada para proteger el servidor web es instalar soporte para HTTPS (HTTP seguro). Este protocolo protege los datos con el cifrado SSL/TLS. Para obtener más información, consulte Tutorial: Configurar SSL/TLS con la AMI de Amazon Linux en la Guía del usuario deAmazon EC2.

Conecte el servidor web Apache con el clúster de base de datos.

A continuación, agregue contenido al servidor web Apache que se conecta a su clúster de base de datos de Amazon Aurora.

Para agregar contenido al servidor web Apache que se conecta a su clúster de base de datos

1. Mientras está conectado a la instancia de EC2, cambie el directorio a `/var/www` y cree un subdirectorio nuevo denominado `inc`.

   ```
   cd /var/www
   mkdir inc
   cd inc
   ```

2. Cree un archivo en el directorio `inc`, denominado `dbinfo.inc` y, a continuación, edite el archivo mediante nano (o a cualquier otro editor de su elección).

   ```
   >dbinfo.inc
   nano dbinfo.inc
   ```

3. Añada el siguiente contenido al archivo `dbinfo.inc`. Aquí, *db_instance_endpoint* es punto de conexión de escritor de clúster de base de datos, sin el puerto, para su clúster de base de datos.

   > **Note**
   >
   > Recomendamos colocar la información del nombre de usuario y la contraseña en una carpeta que no forme parte de la raíz del documento del servidor web. Al hacerlo, se reduce la posibilidad de que se exponga la información de seguridad.
   > Asegúrese de cambiar la `master password` a una contraseña adecuada en su aplicación.

   ```
   <?php

   define('DB_SERVER', 'db_cluster_writer_endpoint');
   define('DB_USERNAME', 'tutorial_user');
   define('DB_PASSWORD', 'master password');
   define('DB_DATABASE', 'sample');
   ?>
   ```

4. Guarde y cierre el archivo `dbinfo.inc`. Si utiliza nano, guarde y cierre el archivo con Ctrl+S y Ctrl+X.
5. Cambie el directorio a `/var/www/html`.

   ```
   cd /var/www/html
   ```

6. Cree un archivo en el directorio `html`, denominado `SamplePage.php` y, a continuación, edite el archivo mediante nano (o a cualquier otro editor de su elección).

   ```
   >SamplePage.php
   nano SamplePage.php
   ```

7. Añada el siguiente contenido al archivo `SamplePage.php`:

   MariaDB & MySQL

   ```
   <?php include "../inc/dbinfo.inc"; ?>
   <html>
   <body>
   ```

```
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
```

```
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";
```

```
   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
<?php
```

```
/* Connect to PostgreSQL and select the database. */
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
```

```
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
?>
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}
```

```
/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;

}
?>
```

8. Guarde y cierre el archivo `SamplePage.php`.
9. Compruebe que el servidor web se conecta correctamente a su clúster de base de datos abriendo un navegador web y navegando a `http://`*`EC2 instance endpoint`*`/SamplePage.php`, por ejemplo: `http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php`.

Puede utilizar `SamplePage.php` para agregar datos a su clúster de base de datos. Los datos que añada se mostrarán en la página. Para verificar que los datos se insertaron en la tabla, puede instalar el cliente MySQL en la instancia de Amazon EC2. A continuación, se conectará a el clúster de base de datos y ejecutará una consulta en la tabla.

Para obtener más información acerca de la conexión al clúster de base de datos, consulte Conexión a un clúster de base de datos Amazon Aurora.

Para asegurarse de que su clúster de base de datos es lo más seguro posible, compruebe que los orígenes fuera de la VPC no se pueden conectar a su clúster de base de datos.

Una vez que haya terminado de probar su servidor web y su base de datos, debe eliminar el clúster de base de datos y la instancia Amazon EC2.

- Para eliminar un clúster de base de datos, siga las instrucciones en Eliminación de clústeres e instancias de base de datos de Aurora. No es necesario crear una instantánea final.
- Para finalizar una instancia Amazon EC2, siga las instrucciones de Terminar su instancia en la Guía del usuario de Amazon EC2.

Tutoriales de Amazon Aurora y código de muestra

La documentación de AWS incluye varios tutoriales que lo guiarán a través de los casos de uso comunes de Amazon Aurora. Muchos de estos tutoriales muestran cómo utilizar Amazon Aurora con otros servicios de AWS. Además, puede acceder al código de ejemplo en .

> **Note**
>
> Puede encontrar más tutoriales en el Blog de AWS Database. Para obtener información acerca de la capacitación, consulte AWS Training and Certification.

Temas

- Tutoriales en esta guía
- Tutoriales en otras AWS guías
- Tutoriales y código de muestra en GitHub
- Manual de instrucciones de bases de datos de AWS
- Portal de contenido de laboratorios y talleres de AWS para Amazon Aurora PostgreSQL
- Portal de contenido de laboratorios y talleres de AWS para Amazon Aurora MySQL
- Cómo utilizar este servicio con un AWS SDK

Tutoriales en esta guía

En los siguientes tutoriales aprenderá a realizar tareas comunes con Amazon Aurora:

- Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4)

  Aprenda a incluir un clúster de BD en una nube privada virtual (VPC) basada en el servicio Amazon VPC. En este caso, la VPC comparte datos con un servidor web que se ejecuta en una instancia de Amazon EC2 en la misma VPC.
- Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble)

  Aprenda a incluir un clúster de BD en una nube privada virtual (VPC) basada en el servicio Amazon VPC. En este caso, la VPC comparte datos con una instancia de Amazon EC2 en la

misma VPC. En este tutorial, creará la VPC para este escenario que funciona con una base de datos que se ejecuta en modo de pila doble.

- Explicación: crear un servidor web y un clúster de base de datos de Amazon Aurora

  Obtenga información sobre cómo instalar un servidor web Apache con PHP y crear una base de datos MySQL. El servidor web se ejecuta en una instancia Amazon EC2 mediante Amazon Linux y la base de datos de MySQL es , un clúster de base de datos de Aurora MySQL. Tanto la instancia de Amazon EC2 y el clúster de de base de datos se ejecutan en una Amazon VPC.

- Tutorial: Restaurar un clúster de base de datos de Amazon Aurora desde una instantánea de clúster de base de datos

  Obtenga información sobre cómo restaurar un clúster de base de datos desde una instantánea de clúster de base de datos.

- Tutorial: Uso de etiquetas para especificar qué clústeres de base de datos de Aurora se deben detener

  Obtenga información sobre cómo usar etiquetas para especificar qué clústeres de base de datos de Aurora detener.

- Tutorial: Registrar el estado de una instancia de base de datos con Amazon EventBridge

  Obtenga información para registrar un cambio de estado de instancia de base de datos mediante Amazon EventBridge y AWS Lambda.

Tutoriales en otras AWS guías

Los siguientes tutoriales de otras guías de AWS muestran cómo realizar tareas comunes con Amazon Aurora:

> **Note**
>
> Algunos de los tutoriales utilizan instancias de base de datos de Amazon RDS, pero se pueden adaptar para usar clústeres de base de datos de Aurora.

- Tutorial: Aurora Serverless en la guía para desarrolladores de AWS AppSync

  Obtenga información sobre cómo usar AWS AppSync para proporcionar un origen de datos para ejecutar comandos SQL en clústeres de base de datos de Aurora Serverless con la API de datos

habilitada. Puede utilizar los solucionadores de AWS AppSync para ejecutar instrucciones SQL con respecto a la API de datos con consultas, mutaciones y suscripciones de GraphQL.

- Tutorial: Rotación de un secreto para una base de datos de AWS en la guía del usuario de AWS Secrets Manager

  Aprenda a crear un secreto para una AWS base de datos y configurar el secreto a fin de rotar según una programación. Puede activar una rotación manualmente y después confirmar que la nueva versión del secreto continúa proporcionando acceso.

- Tutoriales y ejemplos en la guía para desarrolladores de AWS Elastic Beanstalk

  Obtenga información para implementar aplicaciones que utilizan bases de datos de Amazon RDS con AWS Elastic Beanstalk.

- Uso de datos de una base de datos de Amazon RDS para crear un origen de datos de Amazon ML en la Amazon Machine Learning Developer Guide

  Obtenga información sobre cómo crear un objeto de origen de datos de Amazon Machine Learning (Amazon ML) a partir de datos almacenados en una instancia de base de datos MySQL.

- Habilitación manual del acceso a una instancia de Amazon RDS en una VPC en la Guía del usuario de Amazon Quick Suite

  Obtenga información sobre cómo habilitar el acceso de Quick Suite a una instancia de base de datos de Amazon RDS en una VPC.

Tutoriales y código de muestra en GitHub

En los siguientes tutoriales y en el código de muestra, aprenderá a realizar tareas comunes con Amazon Aurora:

- Creación de una biblioteca de préstamos de Aurora Serverless v2

  Aprenda a crear una aplicación de biblioteca de préstamos en la que los usuarios puedan pedir prestados libros y devolverlos. El ejemplo usa Aurora Serverless v2 y AWS SDK para Python (Boto3)..

- Creación de una aplicación de seguimiento de artículos de Amazon Aurora con una API REST de Spring que consulta datos de Aurora Serverless v2 mediante SDK for Java 2.x

Aprenda a crear una API de REST de Spring que realice consultas de datos de Aurora Serverless v2. Es para que lo utilice una aplicación de React mediante SDK para Java 2.x.

- Creación de una aplicación de seguimiento de artículos de Amazon Aurora que consulta datos de Aurora Serverless v2 mediante AWS SDK para PHP

  Aprenda a crear una aplicación que utilice `RdsDataClient` de la API de datos y Aurora Serverless v2 para realizar un seguimiento e informar sobre los elementos de trabajo. El ejemplo utiliza AWS SDK para PHP.

- Creación de una aplicación de seguimiento de artículos de Amazon Aurora que consulta datos de Aurora Serverless v2 mediante AWS SDK para Python (Boto3)

  Aprenda a crear una aplicación que utilice `RdsDataClient` de la API de datos y Aurora Serverless v2 para realizar un seguimiento e informar sobre los elementos de trabajo. El ejemplo utiliza AWS SDK para Python (Boto3).

Manual de instrucciones de bases de datos de AWS

El manual de instrucciones de base de datos de AWS es una guía completa de bases de datos que le enseña cómo crear, implementar y administrar soluciones de bases de datos rentables y de alto rendimiento en AWS. Los tutoriales paso a paso le guiarán a través de la creación de aplicaciones listas para la producción y la implementación de las aplicaciones con plantillas de CloudFormation. Obtendrá información sobre los servicios de AWS esenciales a medida que cree infraestructuras, implemente redes, desarrolle arquitecturas sin servidor, administre bases de datos e integre la IA generativa. Obtenga información sobre las prácticas recomendadas de AWS que le ayudarán a crear soluciones seguras y escalables y, al mismo tiempo, a optimizar los costos. Si es nuevo en AWS o si es un profesional experimentado, el manual de instrucciones de bases de datos de AWS le ayuda a desarrollar destrezas para resolver los desafíos más comunes en materia de bases de datos e implementar soluciones preparadas para la empresa. El manual de instrucciones incluye las siguientes secciones:

- Introducción a AWS para las aplicaciones de bases de datos: obtenga información sobre aspectos básicos de AWS, como configurar la cuenta y el entorno del cuaderno de Jupyter.
- Conceptos básicos de bases de datos: explore los conceptos básicos de las bases de datos y compare los servicios de bases de datos de AWS para elegir la solución adecuada para las cargas de trabajo.

- Aplicación web sin servidor con Amazon Aurora: cree una aplicación integral para minoristas con Amazon Aurora PostgreSQL que gestione el inventario, los pedidos y los datos de los clientes.
- Supervisión y observabilidad: configure el seguimiento del rendimiento y las alertas para identificar posibles problemas con las bases de datos antes de que afecten a las aplicaciones.
- Escalación con Amazon Aurora: obtenga información sobre cómo crear implementaciones resilientes multirregionales con Aurora DSQL y a escalar las bases de datos para obtener más potencia de procesamiento o distribuirlas en varias instancias para aumentar la capacidad.
- Optimización del rendimiento y el costo: optimice el rendimiento de la base de datos y reduzca los costos con estrategias de ajuste probadas.
- Proceso hacia las bases de datos personalizadas de AWS: cree una infraestructura segura y fiable que amplíe las soluciones de IA generativa y las aplicaciones basadas en datos, desde el prototipo hasta la implementación empresarial.
- Aplicaciones de GenAI con RAG: cree un sistema de búsqueda inteligente de documentos médicos y de seguros que utilice la generación aumentada por recuperación (RAG) para ofrecer resultados precisos y contextuales.

Portal de contenido de laboratorios y talleres de AWS para Amazon Aurora PostgreSQL

La siguiente colección de talleres y otros contenidos prácticos le ayudan a conocer las características y capacidades de Amazon Aurora PostgreSQL:

- Creación de un nuevo clúster de Aurora manualmente

  Aprenda a crear un clúster de Amazon Aurora PostgreSQL de forma manual.
- Configuración de Cloud9 e inicialización de la base de datos

  Aprenda a configurar Cloud9 e inicializar la base de datos de PostgreSQL.
- Clonación rápida

  Aprenda a crear un clon rápido de Aurora.
- Administración de planes de consultas

  Aprenda a controlar los planes de ejecución para un conjunto de instrucciones mediante la administración de planes de consultas.

- Administración de la caché de clústeres

  Obtenga información sobre la característica de administración de la caché de clústeres en Aurora PostgreSQL.

- Transmisiones de actividades de la base de datos

  Aprenda a supervisar y auditar la actividad de la base de datos con esta característica.

- Uso de Información sobre rendimiento

  Aprenda a supervisar y ajustar su instancia de base de datos mediante Información sobre rendimiento.

- Supervisión del rendimiento con las herramientas de RDS

  Aprenda a utilizar las herramientas de AWS y Postgres (Cloudwatch, Monitorización mejorada, Slow Query Logs, Información sobre rendimiento, PostgreSQL Catalog Views) para comprender los problemas de rendimiento e identificar formas de mejorar el rendimiento de su base de datos.

- Réplicas de lectura de escalado automático

  Descubra cómo funciona el escalado automático de las réplicas de lectura de Aurora en la práctica mediante un script generador de carga.

- Prueba de la tolerancia a errores

  Descubra cómo un clúster de base de datos puede tolerar un error.

- Base de datos global de Aurora

  Obtenga información sobre la base de datos global de Aurora.

- Uso de machine learning

  Obtenga información sobre Aurora Machine Learning.

- Aurora Serverless v2

  Obtenga información sobre Aurora Serverless v2.

- Extensiones de lenguaje de confianza para Aurora PostgreSQL

  Aprenda a crear extensiones de alto rendimiento que se ejecuten de forma segura en Aurora PostgreSQL.

Portal de contenido de laboratorios y talleres de AWS para Amazon Aurora MySQL

La siguiente colección de talleres y otros contenidos prácticos le ayudan a conocer las características y capacidades de Amazon Aurora MySQL:

- Creación de un clúster de Aurora

  Aprenda a crear un clúster de Amazon Aurora MySQL de forma manual.
- Creación de un entorno IDE basado en la nube de Cloud9 para conectarse a su base de datos

  Aprenda a configurar Cloud9 e inicializar la base de datos MySQL.
- Clonación rápida

  Aprenda a crear un clon rápido de Aurora.
- Realización de una búsqueda de datos anteriores en un clúster

  Aprenda a realizar una búsqueda de datos anteriores en un clúster de base de datos.
- Uso de Información sobre rendimiento

  Aprenda a supervisar y ajustar su instancia de base de datos mediante Información sobre rendimiento.
- Supervisión del rendimiento con las herramientas de RDS

  Aprenda a utilizar las herramientas de AWS y SQL para conocer los problemas de rendimiento e identificar formas de mejorar el rendimiento de su base de datos.
- Análisis del rendimiento de las consultas

  Aprenda a solucionar problemas relacionados con el rendimiento de SQL con diferentes herramientas.
- Réplicas de lectura de escalado automático

  Descubra cómo funcionan las réplicas de lectura con el escalado automático.
- Prueba de la tolerancia a errores

  Obtenga información sobre las características de alta disponibilidad y tolerancia a errores de Aurora MySQL.

- Base de datos global de Aurora

  Obtenga información sobre la base de datos global de Aurora.

- Aurora Serverless v2

  Obtenga información sobre Aurora Serverless v2.

- Uso de machine learning

  Obtenga información sobre Aurora Machine Learning.

Cómo utilizar este servicio con un AWS SDK

Los kits de desarrollo de software (SDK) de AWS se encuentran disponibles en muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ Ejemplos de código de la
AWS CLI	AWS CLI Ejemplos de código de la
AWS SDK para Go	AWS SDK para Go Ejemplos de código de la
AWS SDK para Java	AWS SDK para Java Ejemplos de código de la
AWS SDK para JavaScript	AWS SDK para JavaScript Ejemplos de código de la
AWS SDK para Kotlin	AWS SDK para Kotlin Ejemplos de código de la
AWS SDK para .NET	AWS SDK para .NET Ejemplos de código de la
AWS SDK para PHP	AWS SDK para PHP Ejemplos de código de la
Herramientas de AWS para PowerShell	Herramientas de AWS para PowerShell Ejemplos de código de la

Documentación de SDK	Ejemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) Ejemplos de código de la
AWS SDK para Ruby	AWS SDK para Ruby Ejemplos de código de la
AWS SDK para Rust	AWS SDK para Rust Ejemplos de código de la
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP Ejemplos de código de la
AWS SDK para Swift	AWS SDK para Swift Ejemplos de código de la

Para obtener ejemplos específicos de este servicio, consulte Ejemplos de código de Aurora con SDK de AWS.

Ejemplo de disponibilidad

¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

Acceso mediante programación a Amazon Aurora

Amazon RDS le proporciona las siguientes herramientas para administrar los recursos de Amazon Aurora mediante programación.

AWS Command Line Interface (AWS CLI)

Puede crear y administrar los recursos de Amazon Aurora mediante el intérprete de comandos de la línea de comandos de la AWS CLI. La AWS CLI ofrece acceso directo a las API para servicios de AWS, por ejemplo, Amazon RDS. Para obtener la sintaxis y ejemplos de los comandos de Amazon RDS, consulte rds en la Referencia de comandos de AWS CLI.

AWS CloudFormation

Con la herramienta Infraestructura como código (IaC) de AWS, puede crear plantillas que describan todos los recursos de Amazon Aurora que desee, y AWS CloudFormation aprovisiona y configura esos recursos. Para obtener más información, consulte the section called “Creación de recursos con AWS CloudFormation”.

Kits de desarrollo de software (SDK) de AWS

AWS proporciona SDK para muchas tecnologías y lenguajes de programación conocidos. Podrá comenzar a llamar de forma más sencilla a los servicios de AWS desde sus aplicaciones en ese lenguaje o tecnología. Para obtener más información sobre estos SDK, consulte Tools for developing and managing applications on AWS.

API de Amazon RDS

Esta API es la interfaz de protocolo de Amazon RDS. Al utilizar esta API, debe formatear correctamente todas las solicitudes de HTTPS y añadir una firma digital válida a cada solicitud. Para obtener más información, consulte Referencia de la API de Amazon RDS.

Console-to-Code

Con esta herramienta, puede generar código para las acciones que realiza en la consola de Amazon RDS y usar ese mismo código en otras herramientas, como, por ejemplo, AWS CloudFormation. Para obtener más información, consulte the section called “Console-to-Code”.

API de datos de RDS

La API de datos de RDS (API de datos) proporciona acceso mediante programación a los datos de las bases de datos de Amazon Aurora sin servidor. Con la API de datos, puede ejecutar un

punto de conexión de HTTP seguro para ejecutar instrucciones de SQL sin tener que administrar las conexiones. Para obtener más información, consulte Uso de la API de datos de Amazon RDS.

Use Console-to-Code para generar código para las acciones de la consola de Amazon Aurora.

La consola proporciona una ruta guiada para crear recursos y probar prototipos. Si desea crear los mismos recursos a escala, necesitará un código de automatización. Console-to-Code es una característica de Amazon Q Developer que puede ayudarlo a empezar a usar el código de automatización. Console-to-Code registra las acciones de la consola, incluidos los valores predeterminados y los valores de parámetros que indique. A continuación, utiliza la IA generativa para sugerir código en el lenguaje y formato que prefiera para las acciones que elija. Como el flujo de trabajo de la consola garantiza que los valores de los parámetros que especifique sean válidos juntos, el código que genere mediante el uso de Console-to-Code tiene valores de parámetros compatibles. Puede usar el código como punto de partida y luego personalizarlo para que esté listo para producción en función de su caso de uso específico.

Por ejemplo, con Console-to-Code, puede registrar la creación de un clúster de base de datos de Aurora y optar por generar código en el formato JSON de AWS CloudFormation. Luego, puede copiar ese código y personalizarlo para usarlo en su plantilla AWS CloudFormation.

Actualmente, Console-to-Code puede generar infraestructura como código (IaC) en los siguientes formatos e idiomas:

- Java de CDK
- Python de CDK
- TypeScript de CDK
- JSON de CloudFormation
- YAML de CloudFormation

Para obtener más información e instrucciones sobre cómo utilizar Console-to-Code, consulte Automatización de servicios AWS con Console-to-Code de Amazon Q Developer en la Guía del usuario de Amazon Q Developer.

Configuración de un clúster de base de datos de Amazon Aurora

En esta sección se muestra cómo configurar un clúster de base de datos Aurora. Antes de crear un clúster de base de datos Aurora, decida qué clase de instancia de base de datos ejecutará el clúster de base de datos. Además, decida dónde se ejecutará el clúster de base de datos al seleccionar una AWS Región. A continuación, cree el clúster de base de datos. Si tiene datos fuera de Aurora, puede migrar los datos a un clúster de base de datos Aurora.

Temas

- Creación de un clúster de base de datos de Amazon Aurora
- Creación de recursos de Amazon Aurora con AWS CloudFormation
- Conexión a un clúster de base de datos Amazon Aurora
- Grupos de parámetros para Amazon Aurora
- Migración de datos a un clúster de base de datos de Amazon Aurora
- Creación de una caché de Amazon ElastiCache mediante el uso de ajustes del clúster de base de datos de Aurora
- Migración automática de bases de datos de EC2 a Amazon Aurora mediante AWS Database Migration Service
- Tutorial: Creación de un clúster de bases de datos de MySQL con un grupo de parámetros personalizados

Creación de un clúster de base de datos de Amazon Aurora

Un clúster de base de datos Amazon Aurora se compone de una instancia de base de datos compatible con MySQL o PostgreSQL y un volumen de clúster que contiene los datos para el clúster de base de datos, copiados en tres zonas de disponibilidad como un único volumen virtual. De forma predeterminada, un clúster de base de datos de Aurora contiene una instancia de base de datos principal que realiza lecturas y escrituras y, opcionalmente, hasta 15 réplicas de Aurora (instancias de base de datos de lectura). Para obtener más información acerca de los clústeres de base de datos Aurora, consulte Clústeres de base de datos de Amazon Aurora.

Aurora tiene dos tipos principales de clústeres de bases de datos:

- Aurora aprovisionada: puede elegir la clase de instancia de base de datos para las instancias de escritor y lector en función de la carga de trabajo prevista. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora. Aurora aprovisionada tiene varias opciones, incluidas las bases de datos globales de Aurora. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.
- Aurora Serverless: Aurora Serverless v2 es una configuración de escalado automático bajo demanda para Aurora. La capacidad se ajusta automáticamente en función de la demanda de la aplicación. Solo se le cobrará por los recursos que consuman los clústeres de base de datos. Esta automatización es especialmente útil para entornos con cargas de trabajo muy variables e impredecibles. Para obtener más información, consulte Uso de Aurora Serverless v2.

A continuación, puede averiguar cómo crear un clúster de base de datos de Aurora. Para comenzar, primero vea Requisitos previos de clúster de base de datos.

Para obtener instrucciones para conectarse al clúster de base de datos de Aurora, consulte Conexión a un clúster de base de datos Amazon Aurora.

Contenido

- Requisitos previos de clúster de base de datos
  - Configurar la red para el clúster de base de datos
    - Configurar la conectividad de red automática con una instancia de EC2
    - Configurar la red manualmente
  - Requisitos previos adicionales
- Creación de un clúster de base de datos

- Creación de una instancia de base de datos principal (escritora)
- Configuración de clústeres de bases de datos de Aurora
- Configuración que no se aplica a los clústeres de base de datos de Amazon Aurora
- Configuración que no se aplica a las instancias de base de datos de Amazon Aurora

Requisitos previos de clúster de base de datos

> **Important**
>
> Para poder crear un clúster de base de datos de Aurora, debe completar las tareas de Configuración del entorno para Amazon Aurora.

A continuación se describen los requisitos previos para crear un clúster de base de datos.

Temas

- Configurar la red para el clúster de base de datos
- Requisitos previos adicionales

Configurar la red para el clúster de base de datos

Solo es posible crear un clúster de base de datos de Amazon Aurora en una nube virtual privada (VPC) basada en el servicio de Amazon VPC, en una Región de AWS que tenga al menos dos zonas de disponibilidad. El grupo de subred de base de datos que elija para el clúster de base de datos debe abarcar al menos dos zonas de disponibilidad. Esta configuración garantiza que su clúster de base de datos siempre tenga al menos una instancia de base de datos disponible ante una conmutación por error, en el caso improbable de que se produzca un error en una zona de disponibilidad.

Si tiene pensado configurar la conectividad entre el nuevo clúster de base de datos y una instancia de EC2 en la misma VPC, puede hacerlo durante la creación del clúster de base de datos. Si tiene pensado conectarse a su clúster de base de datos desde recursos que no sean instancias de EC2 en la misma VPC, puede configurar las conexiones de red manualmente.

Temas

- Configurar la conectividad de red automática con una instancia de EC2

- Configurar la red manualmente

Configurar la conectividad de red automática con una instancia de EC2

Al crear un clúster de base de datos de Aurora, puede usar la Consola de administración de AWS para configurar la conectividad entre una instancia de Amazon EC2 y el nuevo clúster de base de datos. Al hacerlo, RDS configura automáticamente los ajustes de red y VPC. El clúster de base de datos se crea en la misma VPC que la instancia de EC2 para que la instancia de EC2 pueda acceder al clúster de base de datos.

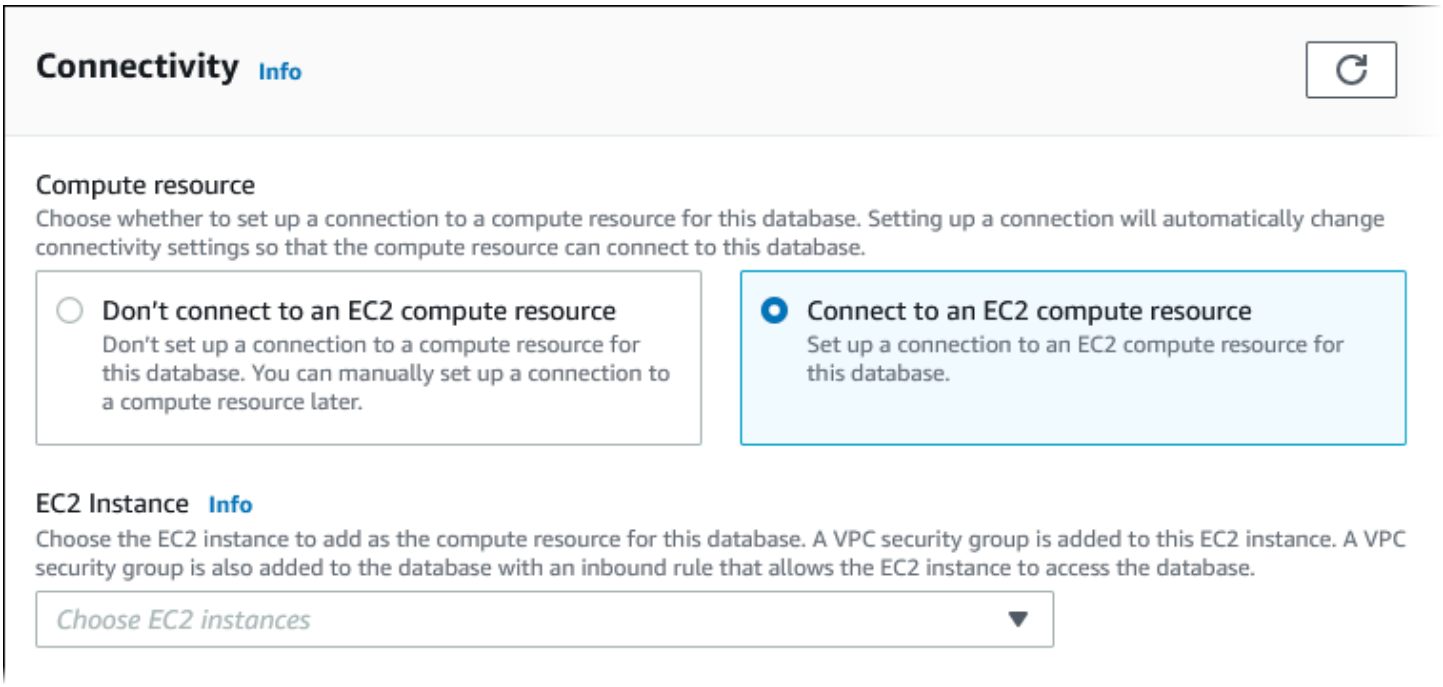
Estos son los requisitos para conectar una instancia de EC2 al clúster de base de datos:

- La instancia de EC2 debe existir en la Región de AWS antes de crear el clúster de base de datos.

  Si no existen instancias de EC2 en la Región de AWS, la consola proporciona un enlace para crear una.
- Actualmente, el clúster de base de datos no puede ser un clúster de base de datos de Aurora Serverless ni parte de una base de datos de Aurora.
- El usuario que crea la instancia de base de datos debe tener permisos para realizar las siguientes operaciones:
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Esta opción permite crear un clúster de base de datos privado. El clúster de base de datos usa un grupo de subredes de base de datos con solo subredes privadas para restringir el acceso a los recursos dentro de la VPC.

Para conectar una instancia de EC2 al clúster de base de datos, seleccione Connect to an EC2 compute resource (Conectarse a un recurso de computación de EC2) en la sección Connectivity (Conectividad) de la página Create database (Crear base de datos).



Cuando elige Connect to an EC2 compute resource (Conectarse a un recurso de computación de EC2), RDS establece las siguientes opciones automáticamente. No puede cambiar esta configuración a menos que decida no configurar la conectividad con una instancia de EC2 seleccionando Don't connect to an EC2 compute resource (No conectarse a un recurso de computación de EC2).

Opción de la consola	Configuración automática
Tipo de red	RDS establece el tipo de red en IPv4. Actualmente, el modo de doble pila no se admite cuando se configura una conexión entre una instancia de EC2 y la instancia de base de datos.
Virtual Private Cloud (VPC) (Nube virtual privada)	RDS establece la VPC en la asociada a la instancia de EC2.

Opción de la consola	Configuración automática
Grupo de subredes de base de datos	RDS necesita un grupo de subredes de base de datos con una subred privada en la misma zona de disponibilidad que la instancia de EC2. Si existe un grupo de subredes de base de datos que cumpla este requisito, RDS utiliza el grupo de subredes de base de datos existente. De forma predeterminada, esta opción está configurada en Automatic setup (Configuración automática). Si selecciona Automatic setup (Configuración automática) y no hay ningún grupo de subredes de base de datos que cumpla este requisito, se produce lo siguiente. RDS usa tres subredes privadas disponibles en tres zonas de disponibilidad, donde una de las zonas de disponibilidad es la misma que la instancia de EC2. Si una subred privada no está disponible en una zona de disponibilidad, RDS crea una subred privada en la zona de disponibilidad. Luego RDS crea el grupo de subredes de base de datos. Cuando hay una subred privada disponible, RDS usa la tabla de enrutamiento asociada a ella y añade cualquier subred que cree a esta tabla de enrutamiento. Cuando no hay ninguna subred privada disponible, RDS crea una tabla de enrutamiento sin acceso a la puerta de enlace de Internet y añade las subredes que crea a la tabla de enrutamiento. RDS también permite utilizar los grupos de subredes de base de datos existentes. Seleccione Choose existing (Elegir existente) si desea utilizar un grupo de subredes de base de datos existente de su elección.
Acceso público	RDS elige No para que no se pueda acceder al clúster de base de datos de forma pública. Por motivos de seguridad, se recomienda mantener la base de datos privada y asegurarse de que no se pueda acceder a ella desde Internet.

Opción de la consola	Configuración automática
Grupo de seguridad de VPC (firewall)	RDS crea un nuevo grupo de seguridad que se asocia al clúster de base de datos. El grupo de seguridad se denomina `rds-ec2-`*n*, donde *n* es un número. Este grupo de seguridad incluye una regla de entrada con el grupo de seguridad de VPC de EC2 (firewall) como origen. Este grupo de seguridad que está asociado al clúster de base de datos permite que la instancia de EC2 acceda al clúster de base de datos. RDS también crea un nuevo grupo de seguridad que se asocia a la instancia de EC2. El grupo de seguridad se denomina `ec2-rds-`*n*, donde *n* es un número. Este grupo de seguridad incluye una regla de salida con el grupo de seguridad de VPC del clúster de base de datos como origen. Este grupo de seguridad permite que la instancia de EC2 envíe tráfico al clúster de base de datos. Para agregar otro grupo de seguridad nuevo, elija Create new (Crear nuevo) y escriba el nombre del nuevo grupo de seguridad . Para añadir grupos de seguridad existentes, elija Choose existing (Elegir existentes) y seleccione los grupos de seguridad que desea añadir.

Opción de la consola	Configuración automática
Zona de disponibilidad	Si no crea una réplica de Aurora en Availability & durability (Disponibilidad y durabilidad) durante la creación del clúster de base de datos (implementación Single-AZ), RDS elige la zona de disponibilidad de la instancia de EC2. Al crear una réplica de Aurora durante la creación del clúster de base de datos (implementación Multi-AZ), RDS elige la zona de disponibilidad de la instancia de EC2 para una instancia de base de datos del clúster de base de datos. RDS elige aleatoria mente una zona de disponibilidad diferente para la otra instancia de base de datos en el clúster de base de datos. La instancia de base de datos primaria o la réplica de Aurora se crean en la misma zona de disponibilidad que la instancia de EC2. Existe la posibilidad de incurrir en costes entre zonas de disponibilidad si la instancia de base de datos y la instancia de EC2 están en zonas de disponibilidad diferentes.

Para obtener más información sobre estas opciones, consulte Configuración de clústeres de bases de datos de Aurora.

Si realiza algún cambio en esta configuración después de crear el clúster de base de datos, los cambios pueden afectar a la conexión entre la instancia de EC2 y el clúster de base de datos.

Configurar la red manualmente

Si tiene pensado conectarse a su clúster de base de datos desde recursos que no sean instancias de EC2 en la misma VPC, puede configurar las conexiones de red manualmente. Si utiliza la Consola de administración de AWS para crear un clúster de base de datos, puede hacer que Amazon RDS cree automáticamente una VPC. O puede usar una VPC ya existente o crear una nueva VPC para su clúster de base de datos de Aurora. Independientemente del enfoque que adopte, la VPC debe tener al menos una subred en cada una de las dos zonas de disponibilidad para usarla con un clúster de base de datos de Amazon Aurora.

De forma predeterminada, Amazon RDS crea automáticamente la instancia de base de datos principal y la réplica de Aurora en las zonas de disponibilidad. Para elegir una zona de disponibilidad específica, debe cambiar la configuración de implementación Multi-AZ de disponibilidad y durabilidad

a Don't create an Aurora Replica (No crear una réplica de Aurora). Hacerlo habilita una opción de Availability Zone (Zona de disponibilidad) que le permite elegir entre las zonas de disponibilidad de la VPC. Sin embargo, le recomendamos encarecidamente que mantenga la configuración predeterminada y deje que Amazon RDS cree una implementación Multi-AZ y elija las zonas de disponibilidad por usted. Al hacerlo, el clúster de base de datos de Aurora se crea con características de conmutación por error rápida y alta disponibilidad que son dos de los beneficios clave de Aurora.

Si no tiene una VPC predeterminada o no ha creado una VPC, puede hacer que Amazon RDS cree automáticamente una VPC cuando se cree un clúster de base de datos de Aurora usando la consola. De lo contrario, tendrá que hacer lo siguiente:

- Cree una VPC que tenga como mínimo una subred en al menos dos de las zonas de disponibilidad de la región de Región de AWS en la que desea implementar su clúster de base de datos. Para obtener más información, consulte Uso de una clúster de base de datos en una VPC y Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).
- Especifique un grupo de seguridad de VPC que autorice las conexiones con su clúster de base de datos de . Para obtener más información, consulte Proporcionar acceso al clúster de base de datos en la VPC mediante la creación de un grupo de seguridad y Control de acceso con grupos de seguridad.
- Especifique un grupo de subredes de base de datos de RDS que defina al menos dos subredes de la VPC que pueda usar el clúster de base de datos de . Para obtener más información, consulte Uso de los grupos de subredes de base de datos.

Para obtener información acerca de las VPC, consulte VPC de Amazon y Amazon Aurora. Para ver un tutorial donde se configura la red para un clúster de base de datos privada, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

Si desea conectarse a un recurso que no está en la misma VPC que el clúster de base de datos de Aurora, consulte los escenarios adecuados en Escenarios de acceso a un clúster de base de datos en una VPC.

Requisitos previos adicionales

Antes de crear el clúster de base de datos, tenga en cuenta los siguientes requisitos previos adicionales:

- Si se conecta a AWS con credenciales de AWS Identity and Access Management (IAM), la cuenta de AWS debe tener políticas de IAM que concedan los permisos necesarios para realizar

operaciones de Amazon RDS. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.

Si usa IAM para acceder a la consola de Amazon RDS, primero tiene que iniciar sesión en la Consola de administración de AWS con sus credenciales de usuario. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

- Si desea adaptar los parámetros de configuración para su clúster de base de datos, debe especificar un grupo de parámetros de clúster de base de datos y un grupo de parámetros de base de datos con la configuración de parámetros requerida. Para obtener información acerca de cómo crear o modificar un grupo de parámetros de clúster de base de datos o un grupo de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora.
- Determine el número de puerto de TCP/IP que quiera especificar para el clúster de base de datos. Los firewalls de algunas compañías bloquean las conexiones a los puertos predeterminados (3306 para MySQL, 5432 para PostgreSQL) de Aurora. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el clúster de base de datos. Todas las instancias de un clúster de base de datos usan el mismo puerto.
- Si la versión principal del motor de su base de datos ha alcanzado la fecha de finalización del soporte estándar de RDS, deberá utilizar la opción CLI de soporte extendido o el parámetro API de RDS. Para obtener más información, consulte el uso del soporte extendido de RDS en Configuración de clústeres de bases de datos de Aurora.

Creación de un clúster de base de datos

Puede crear un clúster de base de datos Aurora mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Puede crear una cluster de base de datos mediante la Consola de administración de AWS con Creación sencilla habilitada o deshabilitada. Con Easy create (Creación sencilla) habilitada, únicamente debe especificar el tipo de motor de base de datos, el tamaño de la instancia de base de datos y el identificador de instancias de bases de datos. Easy create (Creación sencilla) utiliza la configuración predeterminada para otras opciones de configuración. Con Easy create (Creación sencilla) deshabilitada, se especifican más opciones de configuración al crear una base de datos, incluidas las de disponibilidad, seguridad, copias de seguridad y mantenimiento.

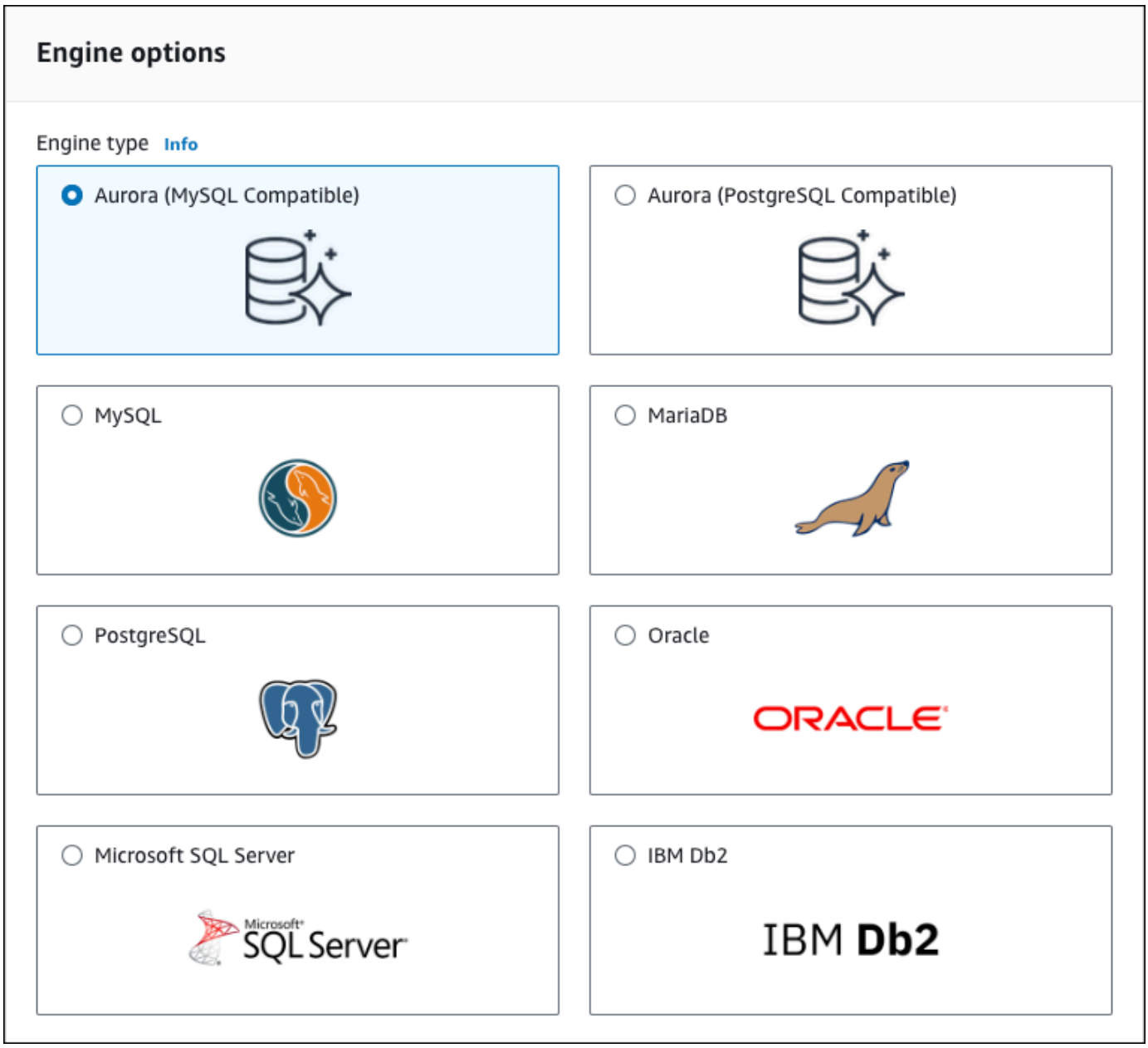
> **Note**
>
> En este ejemplo, la opción Standard Create (Creación estándar) está habilitada y la opción Easy Create (Creación sencilla) no está habilitada. Para obtener más información acerca de cómo crear un clúster de base de datos con Creación sencilla habilitada, consulte Introducción a Amazon Aurora.

Para crear un clúster de base de datos de Aurora con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la Consola de administración de AWS, elija la región de AWS en la que desea crear el clúster de base de datos.

   Aurora no está disponible en todas las regiones de AWS. Para obtener una lista de las regiones de AWS en las que Aurora está disponible, consulte Disponibilidad por región.
3. En el panel de navegación, elija Databases (Bases de datos).
4. Elija Create database (Creación de base de datos).
5. En Elegir un método de creación de base de datos, elija Creación estándar.
6. En Tipo de motor, seleccione una de las opciones siguientes:

   - Aurora (compatible con MySQL)
   - Aurora (compatible con PostgreSQL)



7. Elija la Versión del motor.

   Para obtener más información, consulte Versiones de Amazon Aurora. Puede usar los filtros para elegir versiones que sean compatibles con las características que desee, como Aurora Serverless v2. Para obtener más información, consulte Uso de Aurora Serverless v2.

8. En Templates (Plantillas), elija la plantilla que coincida con su caso de uso.

9. Para ingresar la contraseña maestra, proceda del modo siguiente:

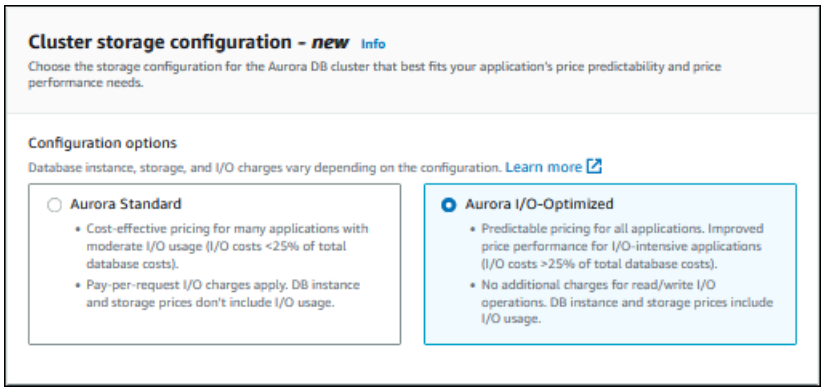
   a. En la sección Configuración, expanda Configuración de credenciales.

b. Desmarque la casilla Auto generate a password (Generar automáticamente una contraseña).

c. (Opcional) Cambie el valor de Master username (Nombre de usuario maestro) e introduzca la misma contraseña en Master password (Contraseña maestra) y Confirm password (Confirmar contraseña).

De forma predeterminada, la nueva instancia de base de datos utiliza una contraseña generada automáticamente para el usuario maestro.

10. En la sección Conectividad del Grupo de seguridad de VPC (firewall), si selecciona Crear nuevo, se crea un grupo de seguridad de VPC con una regla de entrada que permite que la dirección IP del equipo local acceda a la base de datos.
11. Para Configuración del almacenamiento en clústeres, elija Aurora I/O-Optimized o Aurora Standard. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.



12. (Opcional) Configure una conexión a un recurso de computación para este clúster de base de datos.

    Puede configurar la conectividad entre una instancia de Amazon EC2 y el nuevo clúster de base de datos durante la creación del clúster de base de datos. Para obtener más información, consulte Configurar la conectividad de red automática con una instancia de EC2.
13. En el resto de secciones, especifique los ajustes de configuración del clúster de base de datos. Para obtener más información acerca de cada ajuste, consulte Configuración de clústeres de bases de datos de Aurora.
14. Elija Create database (Crear base de datos).

    Si decide utilizar una contraseña generada automáticamente, el botón View credential details (Ver detalles de credenciales) aparece en la página Databases (Bases de datos).

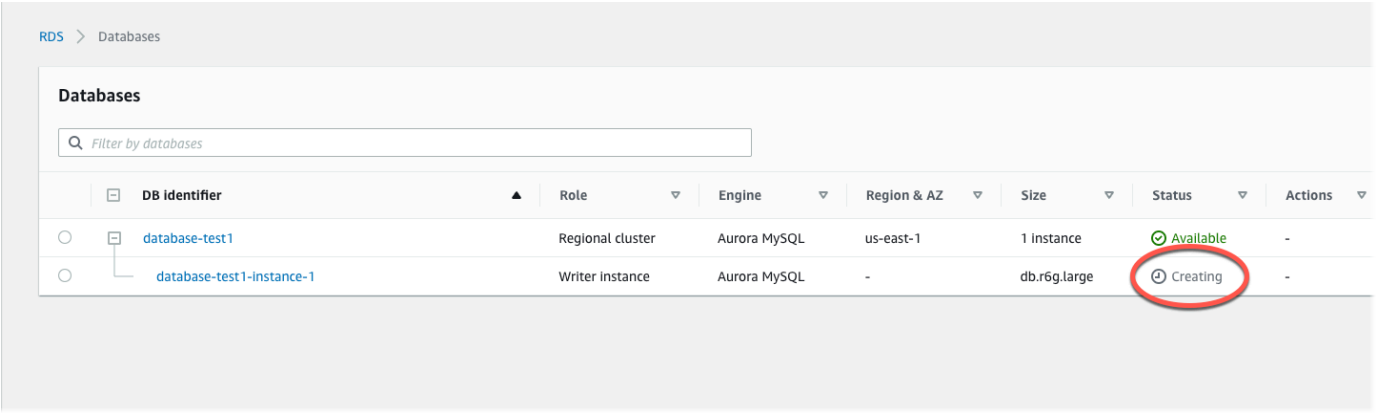
Para consultar la contraseña y el nombre de usuario maestros del clúster de base de datos, seleccione View credential details (Ver detalles de credenciales).

Para conectarse a la instancia de base de datos como usuario maestro, utilice el nombre de usuario y la contraseña que aparecen.

> **Important**
>
> No puede ver la contraseña de usuario maestro de nuevo. Si no la registra, es posible que tenga que cambiarla. Si tiene que cambiar la contraseña de usuario maestro después de que la instancia de base de datos esté disponible, puede modificar la instancia de base de datos para ello. Para obtener más información acerca de la modificación de una instancia de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

15. En Databases (Bases de datos), seleccione el nombre del nuevo clúster de base de datos de Aurora.

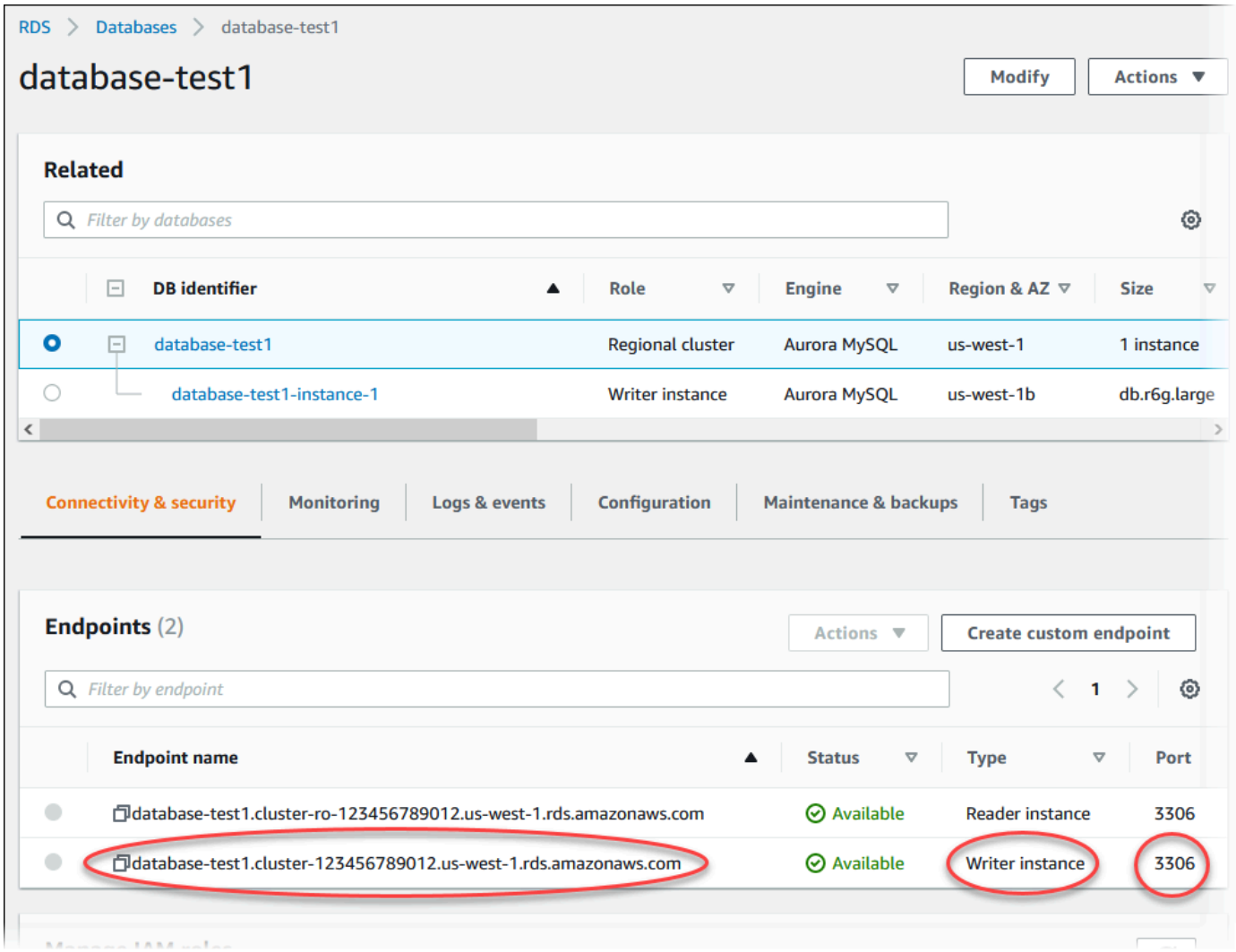
Los detalles del nuevo clúster de base de datos aparecen en la consola de RDS. El clúster de base de datos y su instancia de base de datos tienen el estado creating (creándose) hasta que el clúster de base de datos esté listo para su uso.



Cuando el estado cambie a available (disponible) en ambos, podrá conectarse al clúster de base de datos. Dependiendo de la clase de instancia de base de datos y de la cantidad de almacenamiento, es posible que el nuevo clúster de base de datos tarde hasta 20 minutos en estar disponible.

Para ver el clúster recién creado, elija Databases (Bases de datos) en el panel de navegación de la consola de Amazon RDS. A continuación, seleccione el clúster de base de datos para mostrar

los detalles de dicho clúster. Para obtener más información, consulte Visualización de un clúster de base de datos de Amazon Aurora.



En la pestaña Connectivity & security (Conectividad y seguridad), anote el puerto y el punto de enlace de la instancia de base de datos de escritor. Utilice el punto de enlace y el puerto del clúster en sus cadenas de conexión JDBC y ODBC para cualquier aplicación que realice operaciones de lectura o escritura.

AWS CLI

> **Note**
>
> Para poder crear un clúster de base de datos Aurora a través de la AWS CLI, debe cumplir los requisitos previos, como crear una VPC y un grupo de subred de base de datos de RDS. Para obtener más información, consulte Requisitos previos de clúster de base de datos.

Puede utilizar la AWS CLI para crear un clúster de base de datos de Aurora MySQL o un clúster de base de datos de Aurora PostgreSQL.

Para crear un clúster de base de datos de Aurora MySQL mediante la AWS CLI

Al crear un clúster o una instancia de base de datos Aurora MySQL 8.0 o 5.7, debe especificar `aurora-mysql` para la opción `--engine`.

Realice los pasos siguientes:

1. Identifique el identificador del grupo de subred de base de datos y el grupo de seguridad de la VPC para el nuevo clúster de base de datos y, a continuación, llame al comando create-db-cluster de la AWS CLI para crear el clúster de base de datos de Aurora MySQL.

   Por ejemplo, el comando siguiente crea un nuevo clúster de base de datos compatible con MySQL 8.0 llamado `sample-cluster`. El clúster usa la versión del motor predeterminada y el tipo de almacenamiento Aurora I/O-Optimized.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-mysql --engine-version 8.0 \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Para Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-mysql --engine-version 8.0 ^
       --storage-type aurora-iopt1 ^
   ```

```
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

El comando siguiente crea un nuevo clúster de base de datos compatible con MySQL 5.7 llamado `sample-cluster`. El clúster usa la versión del motor predeterminada y el tipo de almacenamiento Aurora Standard.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster  \
    --engine aurora-mysql --engine-version 5.7 \
    --storage-type aurora \
    --master-username user-name --manage-master-user-password \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster sample-cluster  ^
    --engine aurora-mysql --engine-version 5.7 ^
    --storage-type aurora ^
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Si usa la consola para crear un clúster de base de datos, Amazon RDS crea automáticamente la instancia principal (de escritura) del clúster de base de datos. Si usa la AWS CLI para crear un clúster de base de datos, debe crear expresamente la instancia principal del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos. Los puntos de conexión del clúster de base de datos permanecen con el estado `Creating` hasta que cree la instancia de base de datos principal.

   Llame al comando create-db-instance de la AWS CLI para crear la instancia principal del clúster de base de datos. Incluya el nombre del clúster de base de datos como valor de la opción `--db-cluster-identifier`.

   > **Note**
   >
   > No puede establecer la opción `--storage-type` para las instancias de bases de datos. Puede configurarla solo para clústeres de base de datos.

Por ejemplo, el comando siguiente crea una nueva instancia de base de datos compatible con MySQL 5.7 o MySQL 8.0 llamada `sample-instance`.

Para Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Para crear un clúster de base de datos de Aurora PostgreSQL mediante la AWS CLI

1. Identifique el identificador del grupo de subred de base de datos y el grupo de seguridad de la VPC para el nuevo clúster de base de datos y, a continuación, llame al comando create-db-cluster de la AWS CLI para crear el clúster de base de datos de Aurora PostgreSQL.

   Por ejemplo, el comando siguiente crea un nuevo clúster de base de datos llamado `sample-cluster`. El clúster usa la versión del motor predeterminada y el tipo de almacenamiento Aurora I/O-Optimized.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-postgresql \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Para Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-postgresql ^
       --storage-type aurora-iopt1 ^
   ```

```
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Si usa la consola para crear un clúster de base de datos, Amazon RDS crea automáticamente la instancia principal (de escritura) del clúster de base de datos. Si usa la AWS CLI para crear un clúster de base de datos, debe crear expresamente la instancia principal del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos. Los puntos de conexión del clúster de base de datos permanecen con el estado `Creating` hasta que cree la instancia de base de datos principal.

   Llame al comando create-db-instance de la AWS CLI para crear la instancia principal del clúster de base de datos. Incluya el nombre del clúster de base de datos como valor de la opción `--db-cluster-identifier`.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance \
        --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
   instance-class db.r5.large
   ```

   Para Windows:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance ^
        --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
   instance-class db.r5.large
   ```

Estos ejemplos especifican la opción `--manage-master-user-password` para generar la contraseña del usuario maestro y administrarla en Secrets Manager. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager. También puede utilizar la opción `--master-password` para especificar y administrar la contraseña usted mismo.

API de RDS

> **Note**
>
> Para poder crear un clúster de base de datos de Aurora a través de la AWS CLI, debe cumplir los requisitos previos, como crear una VPC y un grupo de subred de base de datos

de RDS. Para obtener más información, consulte Requisitos previos de clúster de base de datos.

Identifique el grupo de subred de base de datos y el ID del grupo de seguridad de la VPC para el nuevo clúster de base de datos y, a continuación, llame a la operación CreateDBInstance para crear el clúster de base de datos.

Al crear un clúster o una instancia de base de datos de la versión 2 o 3 de Aurora MySQL, especifique `aurora-mysql` para el parámetro `Engine`.

Al crear un clúster o una instancia de base de datos Aurora PostgreSQL, especifique `aurora-postgresql` para el parámetro `Engine`.

Si usa la consola para crear un clúster de base de datos, Amazon RDS crea automáticamente la instancia principal (de escritura) del clúster de base de datos. Si usa la API de RDS para crear un clúster de base de datos, debe crear explícitamente la instancia principal del clúster de base de datos con CreateDBInstance. La instancia principal es la primera instancia que se crea en un clúster de bases de datos. Los puntos de conexión del clúster de base de datos permanecen con el estado `Creating` hasta que cree la instancia de base de datos principal.

Creación de una instancia de base de datos principal (escritora)

Si usa la Consola de administración de AWS para crear un clúster de base de datos, Amazon RDS crea automáticamente la instancia principal (de escritura) del clúster de base de datos. Si usa la AWS CLI o la API de RDS para crear un clúster de base de datos, debe crear expresamente la instancia principal del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos. Los puntos de conexión del clúster de base de datos permanecen con el estado `Creating` hasta que cree la instancia de base de datos principal.

Para obtener más información, consulte Creación de un clúster de base de datos.

> **Note**
>
> Si tiene un clúster de base de datos sin una instancia de base de datos escritora, también denominado clúster headless, no puede usar la consola para crear una instancia de escritura. Debe utilizar la AWS CLI o la API de RDS.

En el siguiente ejemplo, se utiliza el comando de la AWS CLI create-db-instance para crear una instancia de escritura para un clúster de base de datos de Aurora PostgreSQL denominado `headless-test`.

```
aws rds create-db-instance \
    --db-instance-identifier no-longer-headless \
    --db-cluster-identifier headless-test \
    --engine aurora-postgresql \
    --db-instance-class db.t4g.medium
```

Configuración de clústeres de bases de datos de Aurora

La siguiente tabla contiene detalles sobre los ajustes de configuración que se eligen al crear un clúster de base de datos de Aurora.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Actualización automática de versiones secundarias	Elija Enable auto minor version upgrade (Habilitar actualización automática de versiones secundarias) si desea habilitar su clúster de base de datos de Aurora para recibir actualizaciones de las versiones secundarias preferidas del motor de base de datos automáticamente cuando estén disponibles. El ajuste Auto minor version upgrade (Actualización automática a versiones secundarias) solo se aplica a los clústeres de base de datos de Aurora PostgreSQL y Aurora MySQL. Para obtener más información acerca de las actualizaciones de	Establezca este valor para cada instancia de base de datos del clúster de Aurora. Si alguna instancia de base de datos del clúster tiene esta configuración desactivada, el clúster no se actualiza automáticamente. Con la AWS CLI, ejecute `create-db-instance` y establezca la opción `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade`. Mediante la API de RDS, llame a `CreateDBInstance` y establezca el parámetro `AutoMinorVersionUpgrade`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
	motor de Aurora PostgreSQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL. Para obtener más información acerca de las actualizaciones de motor de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.	
AWS KMS key	Solo está disponible si Encryption (Cifrado) se establece en Enable encryption (Habilitar cifrado). Elija la AWS KMS key que se va a utilizar para el cifrado de este clúster de base de datos. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--kms-key-id` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `KmsKeyId`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Backtrack	Se aplica solo a Aurora MySQL. Seleccione Enable Backtrack (Habilitar Backtrack) para habilitar la búsqueda de datos anteriores o Disable Backtrack (Deshabilitar Backtrack) para deshabilitarla. Utilizando la búsqueda de datos anteriores puede rebobinar un clúster de base de datos a un momento específico, sin crear un nuevo clúster de base de datos. Está deshabilitado de forma predeterminada. Si habilita la búsqueda de datos anteriores, especifique también la cantidad de tiempo que desea poder realizar la búsqueda de datos anteriores en su clúster de base de datos (la ventana de búsqueda de datos anteriores de destino). Para obtener más información, consulte Búsqueda de datos anteriores de un clúster de base de datos de Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--backtrack-window` . Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `BacktrackWindow` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Certificate authority (Autoridad de certificado)	La entidad de certificación (CA) del certificado de servidor que utilizan las instancias de base de datos en el clúster de base de datos. Para obtener más información, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.	Con la AWS CLI, ejecute `create-db-instance` y establezca la opción `--ca-certificate-identifier`. Mediante la API de RDS, realice una llamada a `CreateDBInstance` y establezca el parámetro `CACertificateIdentifier`.
Configuración de almacenamiento en clústeres	El tipo de almacenamiento para el clúster de base de datos: Aurora I/O-Optimized o Aurora Standard. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--storage-type`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `StorageType`.
Copy tags to snapshots (Copiar etiquetas en instantáneas)	Seleccione esta opción para que, al crear una instantánea de base de datos, se copien en ellas las etiquetas de las instancias de base de datos. Para obtener más información, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--copy-tags-to-snapshot | --no-copy-tags-to-snapshot`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `CopyTagsToSnapshot`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Database authentication (Autenticación de bases de datos)	La autenticación de bases de datos que desea usar. Para MySQL: • Elija Password authentication (Autenticación de contraseña) para autenticar solo a los usuarios de la base de datos con contraseñas de base de datos. • Elija Password and IAM database authentication (Contraseña y autenticación de base de datos de IAM) para autenticar a los usuarios de la base de datos con contraseñas y credenciales de usuario a través de usuarios y roles de IAM. Para obtener más información, consulte Autenticación de bases de datos de IAM . Para PostgreSQL: • Elija IAM database authentication (Autenticación de base de datos de IAM) para autenticar a los usuarios de la base de datos con contraseñas y credenciales de la base de datos a través de usuarios y roles. Para obtener más información, consulte	Para usar la autenticación de la base de datos IAM con el AWS CLI, ejecute `create-db-cluster` y establezca la opción `--enable-iam-database-authentication | --no-enable-iam-database-authentication` . Para usar la autenticación de la base de datos de IAM con la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `EnableIAMDatabaseAuthentication` . Para usar la autenticación de Kerberos con la AWS CLI, ejecute `create-db-cluster` y establezca las opciones `--domain` y `--domain-iam-role-name` . Para usar la autenticación Kerberos con la API de RDS, llame a `CreateDBCluster` y establezca los parámetros `Domain` y `DomainIAMRoleName` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
	Autenticación de bases de datos de IAM . • Elija Kerberos authentication (Autenticación Kerberos) para autenticar contraseñas de base de datos y credenciales de usuario mediante la autentica ción Kerberos. Para obtener más información, consulte Uso de la autenticación Kerberos con Aurora PostgreSQL.	
Database port (Puerto de base de datos)	Especifique el puerto que deben usar las aplicaciones y las utilidades para obtener acceso a la base de datos. Los clústeres de base de datos Aurora MySQL usan de forma predeterminada el puerto MySQL, 3306 y los clústeres de base de datos Aurora PostgreSQL usan de forma predeterminada el puerto PostgreSQL, 5432. Los firewalls de algunas compañías bloquean las conexiones a estos puertos predeterminados. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el nuevo clúster de base de datos.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--port`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `Port`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
DB cluster identifier (Identificador de clúster de base de datos)	Introduzca un nombre para el clúster de base de datos que sea exclusivo para su cuenta en la región de AWS que seleccionó. Este identificador se utiliza en la dirección del punto de enlace del clúster para su clúster de base de datos. Para obtener información acerca del punto de enlace del clúster, consulte Conexiones de puntos de conexión de Amazon Aurora. El identificador del clúster de base de datos tiene las siguientes limitaciones: • Debe contener de 1 a 63 caracteres alfanuméricos o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser único para todos los clústeres de base de datos por cada cuenta de AWS y por cada región de AWS.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--db-cluster-identifier` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `DBClusterIdentifier` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Grupo de parámetros de clúster de base de datos	Elija un grupo de parámetros de clúster de base de datos Aurora cuenta con un grupo de parámetros de clúster de base de datos predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros de clúster de base de datos. Para obtener más información acerca de los grupos de parámetros de clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--db-cluster-parameter-group-name`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `DBClusterParameterGroupName`.
DB instance class (Clase de instancia de base de datos)	Solo se aplica al tipo de capacidad aprovisionada. Elija una clase de instancia de base de datos que defina los requisitos de procesamiento y memoria para cada instancia en el clúster de base de datos. Para obtener más información acerca de las clases de instancias de bases de datos, consulte Clases de instancia de base de datos de Amazon Aurora.	Establezca este valor para cada instancia de base de datos del clúster de Aurora. Con la AWS CLI, ejecute `create-db-instance` y establezca la opción `--db-instance-class`. Mediante la API de RDS, llame a `CreateDBInstance` y establezca el parámetro `DBInstanceClass`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Grupo de parámetros de base de datos	Elija un grupo de parámetro s. Aurora tiene un grupo de parámetros de predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.	Establezca este valor para cada instancia de base de datos del clúster de Aurora. Con la AWS CLI, ejecute `create-db-instance` y establezca la opción `--db-parameter-group-name` . Mediante la API de RDS, realice una llamada a `CreateDBInstance` y establezca el parámetro `DBParameterGroupName` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Grupo de subredes de base de datos	El grupo de subredes de base de datos que desea utilizar para el clúster de base de datos. Seleccione Choose existing (Elegir existente) para utilizar un grupo de subredes de base de datos existente. A continuación, elija el grupo de subredes requerido en la lista desplegable Existing DB subnet groups (Grupos de subredes de base de datos existentes). Elija Automatic setup (Configuración automática) para permitir que RDS seleccione un grupo de subredes de base de datos compatible. Si no existe ninguno, RDS crea un nuevo grupo de subredes para el clúster. Para obtener más información, consulte Requisitos previos de clúster de base de datos.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--db-subnet-group-name` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `DBSubnetGroupName` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Enable deletion protection (Habilitar la protección contra la eliminación	Seleccione Enable deletion protection (Habilitar la protección contra la eliminación) para evitar que se elimine el clúster de base de datos. Si crea un clúster de base de datos de producción con la consola, se habilita de forma predeterminada la protección contra la eliminación.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--deletion-protection | --no-deletion-protection`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `DeletionProtection`.
Enable encryption	Elija `Enable encryption` (Sí) para habilitar el cifrado en reposo para este clúster de base de datos. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--storage-encrypted | --no-storage-encrypted`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `StorageEncrypted`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Enable Enhanced Monitoring	Elija Enable enhanced monitoring (Habilitar monitorización mejorada) a fin de habilitar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta su clúster de base de datos. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.	Establezca estos valores para cada instancia de base de datos del clúster de Aurora. Con la AWS CLI, ejecute `create-db-instance` y establezca las opciones `--monitoring-interval` y `--monitoring-role-arn`. Mediante la API de RDS, realice una llamada a `CreateDBInstance` y establezca los parámetros `MonitoringInterval` y `MonitoringRoleArn`.
Habilitación de la API de datos de RDS	Seleccione Habilitar la API de datos de RDS para habilitar la API de datos de RDS (API de datos). La API de datos proporciona un punto de conexión HTTP seguro para ejecutar instrucciones SQL sin administrar las conexiones. Para obtener más información, consulte Uso de la API de datos de Amazon RDS.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--enable-http-endpoint | --no-enable-http-endpoint`. Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `EnableHttpEndpoint`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Tipo de motor	El nombre del motor de base de datos que se debe utilizar para este clúster de base de datos.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--engine`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `Engine`.
Engine version (Versión del motor)	Solo se aplica al tipo de capacidad aprovisionada. Elija el número de versión del motor de base de datos.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--engine-version` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `EngineVersion` .
Failover priority (Prioridad de conmutación por error)	Elija una prioridad de conmutación por error para la instancia. Si no elije un valor, el ajuste predeterminado es tier-1. Esta prioridad determina el orden en que se promueven las réplicas de Aurora cuando el sistema se recupera de un error en la instancia principal . Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.	Establezca este valor para cada instancia de base de datos del clúster de Aurora. Con la AWS CLI, ejecute `create-db-instance` y establezca la opción `--promotion-tier` . Mediante la API de RDS, llame a `CreateDBInstance` y establezca el parámetro `PromotionTier` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Initial database name (Nombre inicial de la base de datos)	Escriba un nombre para la base de datos predeterminada. Si no se proporciona un nombre para un clúster de base de datos Aurora MySQL, Amazon RDS no crea una base de datos en el clúster de base de datos que se está creando. Si no proporcio na un nombre para un clúster de Aurora PostgreSQL base de datos, Amazon RDS crea una base de datos denominada `postgres`. Para Aurora MySQL, el nombre de base de datos predeterminado tiene las restricciones siguientes: • Debe tener entre 1 y 64 caracteres alfanuméricos. • No puede ser una palabra reservada por el motor de base de datos. Para Aurora PostgreSQL, el nombre de base de datos predeterminado tiene las restricci ones siguientes: • Debe tener entre 1 y 63 caracteres alfanuméricos. • Deben comenzar por una letra. Los caracteres posteriores pueden ser letras, guiones bajos o dígitos (de 0 a 9).	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--database-name` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `DatabaseName` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
	• No puede ser una palabra reservada por el motor de base de datos. Para crear otras bases de datos, conéctese al clúster de base de datos y use el comando SQL CREATE DATABASE. Para obtener más información acerca de la conexión al clúster de base de datos, consulte Conexión a un clúster de base de datos Amazon Aurora.	
Log exports (Exportaciones de registros)	En la sección Logs exports (Exportaciones de registros), elija los registros que desea comenzar a publicar en Amazon CloudWatch Logs. Para obtener más información sobre la publicación de registros de Aurora MySQL en CloudWatch Logs, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs. Para obtener más información sobre la publicación de registros de Aurora PostgreSQL en CloudWatch Logs, consulte Publicación de registros de Aurora PostgreSQL en Amazon CloudWatch Logs.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--enable-cloudwatch-logs-exports`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `EnableCloudwatchLogsExports`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Periodo de mantenimiento	Seleccione Select window (Seleccionar periodo) y especifique el intervalo de tiempo semanal durante el que se puede llevar a cabo el mantenimiento del sistema. O bien elija No preference (Sin preferencia) para que Amazon RDS asigne un período aleatoriamente.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--preferred-maintenance-window`. Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `PreferredMaintenanceWindow`.
Administrar las credenciales maestras en AWS Secrets Manager	Seleccione Manage master credentials in AWS Secrets Manager (Administrar credenciales maestras en AWS Secrets Manager) para administrar la contraseña del usuario maestro en un secreto en Secrets Manager. De forma opcional, elija la clave KMS para proteger el secreto. Elija entre las claves de KMS de su cuenta o bien introduzca la clave de otra cuenta. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.	Con la AWS CLI, ejecute `create-db-cluster` y establezca las opciones `--manage-master-user-password | --no-manage-master-user-password` y `--master-user-secret-kms-key-id`. Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca los parámetros `ManageMasterUserPassword` y `MasterUserSecretKmsKeyId`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Master password (Contraseña maestra)	Escriba una contraseña para iniciar sesión en su clúster de base de datos: • Para Aurora MySQL, la contraseña debe tener entre 8 y 41 caracteres ASCII imprimibles. • Para Aurora PostgreSQL, debe tener entre 8 y 99 caracteres ASCII imprimibles. • No puede contener `/`, `"`, `@` o un espacio.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--master-user-password`. Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `MasterUserPassword`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Master Username (Nombre de usuario maestro)	Escriba un nombre para usarlo como nombre de usuario maestro para iniciar sesión en su clúster de base de datos. • Para Aurora MySQL. el nombre debe tener entre 1 y 16 caracteres alfanuméricos. • Para Aurora PostgreSQL, debe tener entre 1 y 63 caracteres alfanuméricos. • El primer carácter debe ser una letra. • El nombre no puede ser una palabra reservada por el motor de base de datos. Después de crear el clúster de base de datos, no se puede cambiar el nombre de usuario maestro.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--master-username` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `MasterUsername` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Multi-AZ deployment (Implementación Multi-AZ)	Solo se aplica al tipo de capacidad aprovisionada. Determine si desea crear réplicas de Aurora en otras zonas de disponibilidad para permitir la conmutación por error. Si selecciona Create Replica in Different Zone (Crear réplica en otra zona), Amazon RDS crea una réplica de Aurora en su clúster de base de datos en una zona de disponibilidad diferente de la zona de disponibilidad de la instancia principal del clúster de base de datos. Para obtener más información acerca del uso de varias zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--availability-zones`. Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `AvailabilityZones`.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Tipo de red	Protocolos de direccionamiento IP admitidos por el clúster de base de datos. IPv4 para especificar que los recursos se pueden comunicar con el clúster de base de datos solo a través del protocolo de direcciones IPv4. Modo de pila dual para especific ar que los recursos se pueden comunicar con el clúster de base de datos mediante IPv4, IPv6 o ambos. Utilice el modo de pila doble si tiene recursos que deben comunicarse con su clúster de base de datos a través del protocolo de direccion amiento IPv6. Para usar el modo de doble pila, asegúrese de que haya al menos dos subredes que abarquen dos zonas de disponibi lidad que admitan los protocolo s de red IPv4 e IPv6. Además, asegúrese de asociar un bloque CIDR IPv6 a todas las subredes del grupo de subredes de base de datos que especifique. Para obtener más información, consulte Direccionamiento IP de Amazon Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `-network-type` . Mediante la API de RDS, llame a `CreateDBCluster` y establezca el parámetro `NetworkType` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Public access (Acceso público)	Elija Publicly accessible (Accesible públicamente) para dar al clúster de base de datos una dirección IP pública o elija Not publicly accessible (No accesible públicamente). Las instancias del clúster de base de datos pueden ser una combinación de instancias de base de datos públicas y privadas. Para obtener más información acerca del modo de ocultar instancias al acceso público, consulte Cómo ocultar un clúster de base de datos en una VPC desde Internet.. Para conectarse a una instancia de base de datos desde fuera de su Amazon VPC, la instancia de base de datos debe ser accesible públicamente, el acceso debe concederse mediante las reglas entrantes del grupo de seguridad de la instancia de base de datos y deben cumplirse otros requisitos. Para obtener más información, consulte No puede conectarse a la instancia de base de datos de Amazon RDS. Si su instancia de base de datos no es accesible públicamente, también puede usar una conexión Site-to-site VPN AWS	Establezca este valor para cada instancia de base de datos del clúster de Aurora. Con la AWS CLI, ejecute `create-db-instance` y establezca la opción `--publicly-accessible | --no-publicly-accessible` . Mediante la API de RDS, realice una llamada a `CreateDBInstance` y establezca el parámetro `PubliclyAccessible` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
	o una conexión a Direct Connect para acceder a ella desde una red privada. Para obtener más información, consulte Privacidad del tráfico entre redes.	
Soporte extendido de RDS	Seleccione Habilitar soporte extendido de RDS para permitir que las principales versiones del motor compatibles sigan funcionando después de la fecha de fin de soporte estándar de Aurora. Al crear un clúster de base de datos, Amazon Aurora utiliza el Soporte extendido de RDS de forma predeterminada. Para evitar la creación de un nuevo clúster de base de datos después de la fecha del fin del soporte estándar de Aurora y evitar cargos por el soporte extendido de RDS, deshabilite esta configuración. Sus clústeres de bases de datos existentes no incurrirán en cargos hasta la fecha de inicio de los precios del Soporte extendido de RDS. Para obtener más información, consulte Soporte extendido de Amazon RDS con Amazon Aurora.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--engine-lifecycle-support` . Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `EngineLifecycleSupport` .

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
RDS Proxy	Elija Create an RDS Proxy (Crear un RDS Proxy) para crear un proxy para el clúster de base de datos. Amazon RDS crea automáticamente un rol de IAM y un secreto de Secrets Manager para el proxy. Para obtener más información, consulte Amazon RDS Proxy para Aurora.	No está disponible al crear un clúster de base de datos.
Período de retención	Seleccione el tiempo (entre 1 y 35 días) durante el que Aurora conserva las copias de seguridad de la base de datos. Los backups se pueden utilizar para las restauraciones a un momento dado (PITR) de la base de datos con una precisión de segundos.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--backup-retention-period`. Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `BackupRetentionPeriod`.
Turn on DevOps Guru (Activar DevOps Guru)	Elija Turn on DevOps Guru (Activar DevOps Guru) para activar Amazon DevOps Guru para la base de datos de Aurora. Para que DevOps Guru para RDS proporcione un análisis detallado de las anomalías de rendimiento, es necesario activar Performance Insights. Para obtener más información, consulte Configuración de DevOps Guru for RDS.	Puede activar DevOps Guru para RDS desde la consola de RDS, pero no mediante la API de RDS ni la CLI. Para obtener más información sobre la activación de DevOps Guru, consulte la Guía del usuario de Amazon DevOps Guru.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Turn on Performance Insights (Activar Performance Insights)	Elija Turn on Performance Insights (Activar Performance Insights) para activar Amazon RDS Performance Insights. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.	Establezca estos valores para cada instancia de base de datos del clúster de Aurora. Con la AWS CLI, ejecute `create-db-instance` y configure las opciones `--enable-performance-insights` | `--no-enable-performance-insights`, `--performance-insights-kms-key-id`, y `--performance-insights-retention-period`. Con la API de RDS, llame a `CreateDBInstance` y establezca los parámetros `EnablePerformanceInsights`, `PerformanceInsightsKMSKeyId` y `PerformanceInsightsRetentionPeriod`.
Virtual Private Cloud (VPC) (Nube virtual privada)	Elija la VPC que alojará el clúster de base de datos. Seleccione Create a New VPC (Crear una VPC) para que Amazon RDS cree una VPC automáticamente. Para obtener más información, consulte Requisitos previos de clúster de base de datos.	Para la AWS CLI y la API, especifique los ID de grupo de seguridad de la VPC.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
Grupo de seguridad de VPC (firewall)	Seleccione Create new (Crear nuevo) para que Amazon RDS cree un grupo de seguridad de VPC automáticamente. O bien elija Choose existing (Seleccionar existente) y especifique uno o varios grupos de seguridad de VPC para proteger el acceso de red al clúster de base de datos. Al seleccionar Create new (Crear nuevo) en la consola de RDS, se crea un nuevo grupo de seguridad con una regla de entrada que permite el acceso a la instancia de base de datos desde la dirección IP detectada en su navegador. Para obtener más información, consulte Requisitos previos de clúster de base de datos.	Con la AWS CLI, ejecute `create-db-cluster` y establezca la opción `--vpc-security-group-ids`. Mediante la API de RDS, realice una llamada a `CreateDBCluster` y establezca el parámetro `VpcSecurityGroupIds`.

Configuración que no se aplica a los clústeres de base de datos de Amazon Aurora

Las siguientes configuraciones en el comando `create-db-cluster` de la AWS CLI y la operación `CreateDBCluster` de la API de RDS no se aplican a los clústeres de base de datos de Amazon Aurora.

> **Note**
>
> Consola de administración de AWS no muestra esta configuración para los clústeres de base de datos Aurora.

Configuración de la AWS CLI	Configuración de la API de RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`
`--publicly-accessible | --no-publicly-accessible`	`PubliclyAccessible`

Configuración que no se aplica a las instancias de base de datos de Amazon Aurora

Las siguientes configuraciones en el comando `create-db-instance` de la AWS CLI y la operación `CreateDBInstance` de la API de RDS no se aplican a Amazon Aurora.

> **Note**
>
> Consola de administración de AWS no muestra esta configuración para las instancias de base de datos Aurora.

Configuración de la AWS CLI	Configuración de la API de RDS
`--allocated-storage`	`AllocatedStorage`
`--availability-zone`	`AvailabilityZone`
`--backup-retention-period`	`BackupRetentionPeriod`
`--backup-target`	`BackupTarget`
`--character-set-name`	`CharacterSetName`
`--character-set-name`	`CharacterSetName`
`--custom-iam-instance-profile`	`CustomIamInstanceProfile`
`--db-security-groups`	`DBSecurityGroups`
`--deletion-protection | --no-deletion-protection`	`DeletionProtection`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--enable-cloudwatch-logs-exports`	`EnableCloudwatchLogsExports`
`--enable-customer-owned-ip | --no-enable-customer-owned-ip`	`EnableCustomerOwnedIp`
`--enable-iam-database-authentication | --no-enable-iam-database-authentication`	`EnableIAMDatabaseAuthentication`

Configuración de la AWS CLI	Configuración de la API de RDS
`--engine-version`	`EngineVersion`
`--iops`	`Iops`
`--kms-key-id`	`KmsKeyId`
`--master-username`	`MasterUsername`
`--master-user-password`	`MasterUserPassword`
`--max-allocated-storage`	`MaxAllocatedStorage`
`--multi-az | --no-multi-az`	`MultiAZ`
`--nchar-character-set-name`	`NcharCharacterSetName`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--preferred-backup-window`	`PreferredBackupWindow`
`--processor-features`	`ProcessorFeatures`
`--storage-encrypted | --no-storage-encrypted`	`StorageEncrypted`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--timezone`	`Timezone`
`--vpc-security-group-ids`	`VpcSecurityGroupIds`

Creación de recursos de Amazon Aurora con AWS CloudFormation

Amazon Aurora está integrado con AWS CloudFormation, un servicio que lo ayuda a modelar y configurar sus recursos de AWS para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Cree una plantilla que describa todos los recursos de AWS que desee (como clústeres de base de datos y grupos de parámetros de clúster de base de datos), y CloudFormation aprovisiona y configura esos recursos para usted.

Cuando utiliza CloudFormation, puede volver a usar la plantilla para configurar sus recursos de Aurora de forma coherente y repetida. Solo tiene que describir los recursos una vez y luego aprovisionar los mismos recursos una y otra vez en varias cuentas y regiones de AWS.

Aurora y plantillas de CloudFormation

Las plantillas de CloudFormation son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas de CloudFormation. Si no está familiarizado con JSON o YAML, puede utilizar Designer de CloudFormation para comenzar a utilizar las plantillas de CloudFormation. Para obtener más información, consulte ¿Qué es un diseñador de CloudFormation? en la guía del usuario de AWS CloudFormation.

Aurora admite la creación de recursos en CloudFormation. Para obtener más información, incluidos ejemplos de plantillas JSON y YAML para estos recursos, consulte la referencia del tipo de recurso de RDS en la guía del usuario de AWS CloudFormation.

Más información sobre CloudFormation

Para conocer más información acerca de CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- Guía del usuario de AWS CloudFormation
- Referencia de la API de CloudFormation
- Guía del usuario de la interfaz de la línea de comandos de AWS CloudFormation

Conexión a un clúster de base de datos Amazon Aurora

Puede conectarse a un clúster de base de datos de Aurora con las mismas herramientas que utiliza para conectarse a una base de datos de MySQL o PostgreSQL. Especifica una cadena de conexión con cualquier script, utilidad o aplicación conectada a una instancia de base de datos de PostgreSQL o MySQL. Utiliza la misma clave pública para conexiones de Capa de conexión segura (SSL).

En la cadena de conexión, habitualmente utiliza la información del puerto y el host de terminales especiales asociados con el clúster de base de datos. Con estos terminales, puede utilizar los mismos parámetros de conexión independientemente del número de instancias de base de datos que haya en el clúster. También usa información del host y del puerto de la instancia de base de datos específica en su clúster de base de datos de Aurora, como resolución de problemas.

> **Note**
>
> Para los clústeres de Aurora Serverless base de datos, se conecta al punto de enlace de la base de datos en lugar de a la instancia de base de datos. Puede encontrar el punto de enlace de la base de datos para un clúster de base de datos de Aurora Serverless en la pestaña Conectividad y seguridad de Consola de administración de AWS. Para obtener más información, consulte Uso de Amazon Aurora Serverless v1.

Independientemente del motor de base de datos Aurora y de las herramientas específicas que utilice para trabajar con el clúster o instancia de base de datos, el punto de enlace debe ser accesible. Un clúster de base de datos de Aurora puede crearse únicamente en una nube virtual privada (VPC) basada en el servicio de Amazon VPC. Esto significa que puede acceder al punto de enlace desde dentro de la VPC o fuera de la VPC utilizando uno de los siguientes enfoques.

- Acceda al clúster de base de datos Amazon Aurora dentro de la VPC: habilite el acceso al clúster de base de datos Aurora a través de la VPC. Para ello, edite las reglas entrantes en el grupo Seguridad de la VPC para permitir el acceso al clúster de Aurora base de datos específico. Para obtener más información, incluido cómo configurar la VPC para diferentes escenarios de clúster de Aurora base de datos, consulte Amazon nube virtual privada VPC y Amazon Aurora.
- Acceda al clúster de Amazon Aurora base de datos fuera de la VPC: para tener acceso a un clúster de base de datos Aurora desde fuera de la VPC, utilice la dirección pública del punto de conexión del clúster de base de datos.

Para obtener más información, consulte Solución de errores de conexión de Aurora.

Contenido

- Conexión a clústeres de bases de datos Aurora con los controladores de AWS
- Conexión a un clúster de base de datos Amazon Aurora MySQL
  - Utilidades de conexión para Aurora MySQL
  - Conexión a Aurora MySQL con la utilidad MySQL
  - Conexión a Aurora MySQL con el controlador JDBC de Amazon Web Services (AWS)
  - Conexión a Aurora MySQL con el controlador de Python de Amazon Web Services (AWS)
  - Conexión a Aurora MySQL con el controlador ODBC de Amazon Web Services (AWS) para MySQL
  - Conexión a Aurora MySQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)
  - Conexión a Aurora MySQL mediante SSL
- Conexión a un clúster de base de datos Amazon Aurora PostgreSQL
  - Utilidades de conexión para Aurora PostgreSQL
  - Conexión a Aurora PostgreSQL con el controlador JDBC de Amazon Web Services (AWS)
  - Conexión a Aurora PostgreSQL con el controlador de Python de Amazon Web Services (AWS)
  - Conexión a Aurora PostgreSQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)
- Solución de errores de conexión de Aurora

Conexión a clústeres de bases de datos Aurora con los controladores de AWS

El conjunto de controladores de AWS se han diseñado para permitir tiempos de transición y conmutación por error más rápidos y autenticarse con AWS Secrets Manager, AWS Identity and Access Management (IAM) e identidad federada. Los controladores de AWS se basan en la supervisión del estado del clúster de base de datos y en el conocimiento de la topología del clúster para determinar quién es el nuevo escritor. Este enfoque reduce los tiempos de transición y conmutación por error a segundos de un solo dígito, en comparación con las decenas de segundos de los controladores de código abierto.

En la tabla siguiente se enumeran las características admitidas para cada uno de los controladores. A medida que se introducen nuevas características de servicio, el objetivo del conjunto de controladores de AWS es contar con soporte integrado para estas características de servicio.

Característica	Controlador JDBC de AWS	Controlador de Python de AWS	Controlador ODBC de AWS para MySQL	Contenedor de NodeJS avanzado de AWS
Soporte de conmutación por error	Sí	Sí	Sí	Sí
Supervisión mejorada de conmutación por error	Sí	Sí	Sí	Sí
División de lectura y escritura	Sí	Sí	No	Sí
Rastreador de conexiones de Aurora	Sí	Sí	No	Sí
Conexión de metadatos del controlador	Sí	N/A	N/A	N/A
Telemetría	Sí	Sí	No	Sí
Secrets Manager	Sí	Sí	Sí	Sí
Autenticación de IAM	Sí	Sí	Sí	Sí

Característica	Controlador JDBC de AWS	Controlador de Python de AWS	Controlador ODBC de AWS para MySQL	Contenedor de NodeJS avanzado de AWS
Identidad federada (AD FS)	Sí	Sí	No	Sí
Identidad federada (Okta)	Sí	Sí	Sí	Sí
Base de datos ilimitada de Aurora PostgreSQL	Sí (solo Aurora PostgreSQL)	No	No	Sí (solo Aurora PostgreSQL)

Para obtener más información sobre los controladores de AWS, consulte el controlador de idioma correspondiente a su clúster de base de datos de Aurora MySQL o Aurora PostgreSQL.

Conexión a un clúster de base de datos Amazon Aurora MySQL

Para autenticarse en el clúster de base de datos de Aurora MySQL, puede utilizar la autenticación con nombre de usuario y contraseña de MySQL o la autenticación de base de datos AWS Identity and Access Management (IAM). Para obtener más información sobre el uso de la autenticación con nombre de usuario y contraseña de MySQL, consulte Control de acceso y administración de cuentas en la documentación de MySQL. Para obtener más información sobre cómo usar la autenticación de base de datos de IAM, consulte Autenticación de bases de datos de IAM .

Cuando tenga una conexión a su clúster de base de datos Amazon Aurora con compatibilidad con la versión 8.0 de MySQL, podrá ejecutar comandos de SQL que sean compatibles con la versión 8.0 de MySQL. La versión mínima compatible es MySQL 8.0.23. Para obtener más información sobre la sintaxis SQL de MySQL 8.0, consulte MySQL 8.0 Reference Manual. Para obtener información sobre las limitaciones que se aplican a Aurora MySQL 3, consulte Comparación de Aurora MySQL versión 3 y MySQL 8.0 Community Edition.

Cuando tenga una conexión a su clúster de base de datos Amazon Aurora con compatibilidad con la versión 5.7 de MySQL, podrá ejecutar comandos de SQL que sean compatibles con la versión 5.7

de MySQL. Para obtener más información sobre la sintaxis SQL de MySQL 5.7, consulte MySQL 5.7 Reference Manual. Para obtener información sobre las limitaciones que se aplican a Aurora MySQL 5.7, consulte Aurora MySQL versión 2 compatible con MySQL 5.7.

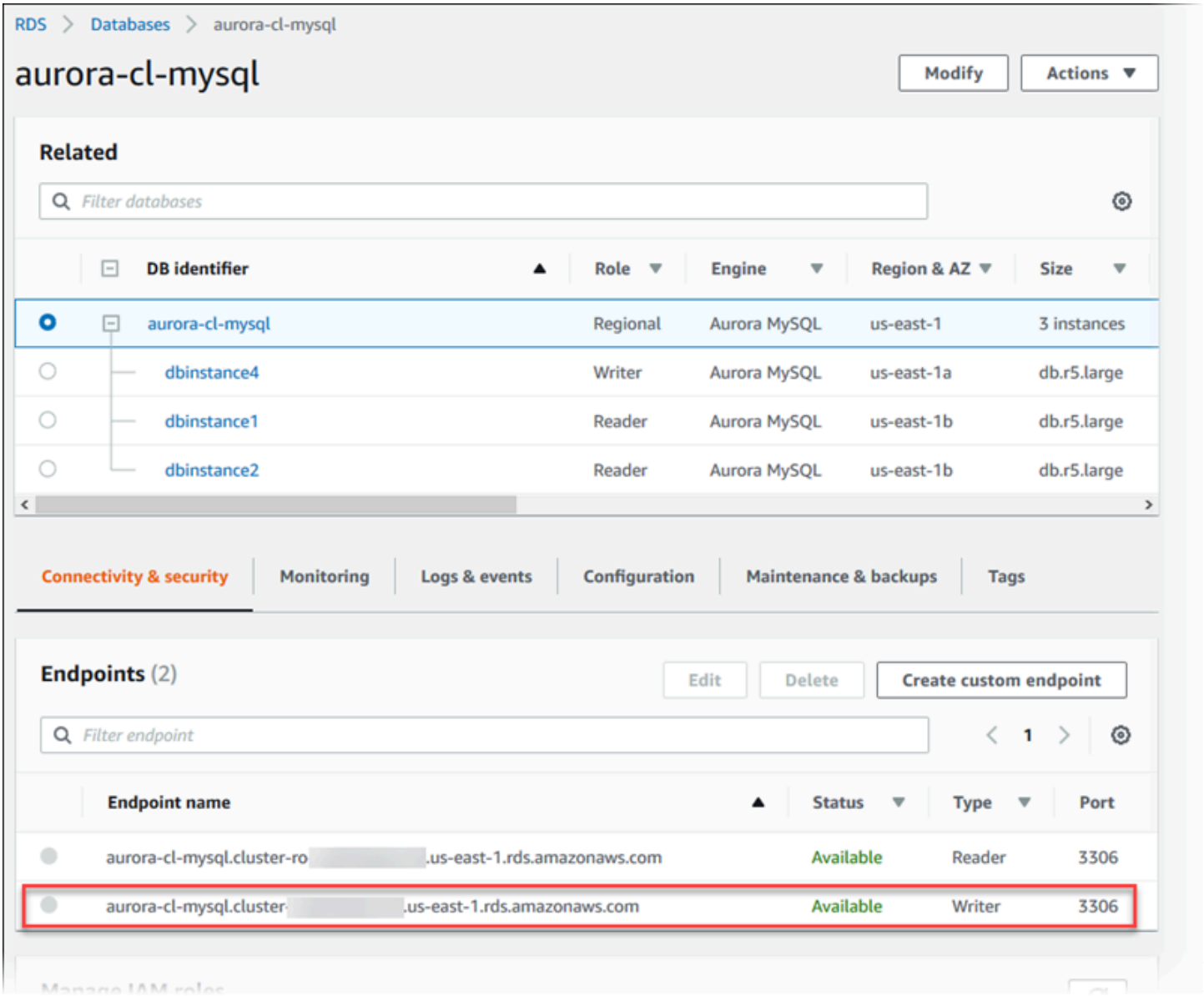
> **Note**
>
> Para obtener una guía detallada y práctica sobre la conexión a un clúster de base de datos de Amazon Aurora MySQL puede consultar el manual Aurora Connection Management.

En la vista de detalles del clúster de base de datos, puede encontrar el punto de enlace de clúster, que se puede usar en la cadena de conexión de MySQL. El punto de enlace se compone del nombre de dominio y el puerto del clúster de base de datos. Por ejemplo, si un valor de punto de enlace es `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, debe especificar los siguientes valores en una cadena de conexión de MySQL:

- Para el host o el nombre del host, especifique `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Para el puerto, especifique `3306` o el valor del puerto que utilizó al crear el clúster de base de datos

El punto de enlace de clúster le conecta a la instancia principal del clúster de base de datos. Puede realizar operaciones de lectura y escritura con el punto de enlace de clúster. El clúster de base de datos también puede tener hasta 15 réplicas de Aurora que admiten acceso de solo lectura a los datos de su clúster de base de datos. La instancia principal y cada réplica de Aurora tienen un punto de enlace único que es independiente del punto de enlace del clúster y que permite establecer conexión directamente con una instancia de base de datos concreta del clúster. El punto de conexión del clúster apunta siempre a la instancia principal. Si se produce un error en la instancia principal y se reemplaza, el punto de enlace del clúster apunta a la nueva instancia principal.

Para ver el punto de enlace del clúster (punto de enlace de escritor), elija Databases (Bases de datos) en la consola de Amazon RDS y elija el nombre del clúster de base de datos para mostrar sus detalles.



Temas

- Utilidades de conexión para Aurora MySQL
- Conexión a Aurora MySQL con la utilidad MySQL
- Conexión a Aurora MySQL con el controlador JDBC de Amazon Web Services (AWS)
- Conexión a Aurora MySQL con el controlador de Python de Amazon Web Services (AWS)
- Conexión a Aurora MySQL con el controlador ODBC de Amazon Web Services (AWS) para MySQL
- Conexión a Aurora MySQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)

- Conexión a Aurora MySQL mediante SSL

Utilidades de conexión para Aurora MySQL

A continuación se indican algunas de las utilidades de conexión que puede usar:

- Línea de comando: puede conectarse a un clúster de base de datos Amazon Aurora usando herramientas como la utilidad de línea de comando de MySQL. Para obtener más información acerca del uso de la utilidad de MySQL, consulte mysql — the MySQL command-line client (mysql: el cliente de línea de comandos de MySQL) en la documentación de MySQL.
- Interfaz gráfica de usuario (GUI): puede usar la utilidad MySQL Workbench para conectarse por medio de una interfaz de usuario. Para obtener más información, consulte la página Download MySQL Workbench.
- Controladores de:AWS
  - Conexión a Aurora MySQL con el controlador JDBC de Amazon Web Services (AWS)
  - Conexión a Aurora MySQL con el controlador de Python de Amazon Web Services (AWS)
  - Conexión a Aurora MySQL con el controlador ODBC de Amazon Web Services (AWS) para MySQL
  - Conexión a Aurora MySQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)

Conexión a Aurora MySQL con la utilidad MySQL

Utilice el siguiente procedimiento: Supone que ha configurado el clúster de base de datos en una subred privada de su VPC. Se conecta mediante una instancia de Amazon EC2 que ha configurado según los tutoriales de Explicación: crear un servidor web y un clúster de base de datos de Amazon Aurora.

> **Note**
>
> Este procedimiento no requiere instalar el servidor web en el tutorial, pero sí instalar MariaDB 10.5.

Para conectarse a un clúster de base de datos mediante la utilidad MySQL

1. Inicie sesión en la instancia de EC2 que usa para conectarse a su clúster de datos.

   Debería ver un resultado similar a este.

   ```
   Last login: Thu Jun 23 13:32:52 2022 from xxx.xxx.xxx.xxx

          __|  __|_  )
          _|  (     /   Amazon Linux 2 AMI
         ___|\___|___|

   https://aws.amazon.com/amazon-linux-2/
   [ec2-user@ip-10-0-xxx.xxx ~]$
   ```

2. Escriba el siguiente comando en el símbolo del sistema para conectarse a la instancia de base de datos principal del clúster de base de datos.

   Para el parámetro `-h`, escriba el nombre de DNS del punto de enlace de la instancia principal. Para el parámetro `-u`, sustituya el ID de usuario de una cuenta de usuario de base de datos.

   ```
   mysql -h primary-instance-endpoint.AWS_account.AWS_Region.rds.amazonaws.com -P 3306
    -u database_user -p
   ```

   Por ejemplo:

   ```
   mysql -h my-aurora-cluster-instance.c1xy5example.123456789012.eu-
   central-1.rds.amazonaws.com -P 3306 -u admin -p
   ```

3. Ingrese la contraseña del usuario de la base de datos.

   Debería ver un resultado similar a este.

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MySQL connection id is 1770
   Server version: 8.0.23 Source distribution

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
   ```

```
MySQL [(none)]>
```

4. Ingrese sus comandos SQL.

Conexión a Aurora MySQL con el controlador JDBC de Amazon Web Services (AWS)

El controlador JDBC de Amazon Web Services (AWS) se ha diseñado como un contenedor JDBC avanzado. Este contenedor complementa y amplía la funcionalidad de un controlador JDBC existente para ayudar a las aplicaciones a aprovechar las características de las bases de datos en clúster, como Aurora MySQL. El controlador se admite con el controlador Connector/J de la comunidad MySQL y el controlador Connector/J de la comunidad MariaDB.

Para instalar el controlador JDBC de AWS, añada el archivo .jar del controlador JDBC de AWS (ubicado en la aplicación `CLASSPATH`) y conserve las referencias al controlador de la comunidad correspondiente. Actualice el prefijo de la URL de conexión correspondiente de la siguiente manera:

- De `jdbc:mysql://` a `jdbc:aws-wrapper:mysql://`
- De `jdbc:mariadb://` a `jdbc:aws-wrapper:mariadb://`

Para obtener más información sobre el controlador JDBC de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del controlador JDBC de Amazon Web Services (AWS).

> **Note**
>
> La versión 3.0.3 de la utilidad MariaDB Connector/J deja de ser compatible con clústeres de base de datos de Aurora, por lo que recomendamos encarecidamente pasar al controlador JDBC de AWS.

Conexión a Aurora MySQL con el controlador de Python de Amazon Web Services (AWS)

El controlador de Python de Amazon Web Services (AWS) se ha diseñado como un contenedor de Python avanzado. Este contenedor complementa y amplía la funcionalidad del controlador de Psycopg de código abierto. El controlador de Python de AWS se admite con las versiones 3.8 y posteriores de Python. Puede instalar el paquete de `aws-advanced-python-wrapper` mediante el comando `pip`, junto con los paquetes de código abierto de `psycopg`.

Para obtener más información sobre el controlador de Python de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del controlador de Python de Amazon Web Services (AWS).

Conexión a Aurora MySQL con el controlador ODBC de Amazon Web Services (AWS) para MySQL

El controlador ODBC de AWS para MySQL es un controlador de cliente diseñado para la alta disponibilidad de Aurora MySQL. El controlador puede existir junto con el conector MySQL/controlador ODBC y es compatible con los mismos flujos de trabajo.

Para obtener más información sobre el controlador ODBC de AWS para MySQL e instrucciones completas para instalarlo y utilizarlo, consulte el repositorio GitHub del controlador ODBC de Amazon Web Services (AWS) para MySQL.

Conexión a Aurora MySQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)

El contenedor de NodeJS avanzado de AWS complementa y amplía la funcionalidad de un controlador NodeJS existente. Ayuda a las aplicaciones a aprovechar las características de las bases de datos agrupadas en clústeres como Aurora MySQL.

Para obtener más información sobre el contenedor de NodeJS avanzado de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del contenedor de NodeJS avanzado de Amazon Web Services (AWS).

Conexión a Aurora MySQL mediante SSL

Puede utilizar políticas el cifrado SSL en las conexiones a una instancia de base de datos de Aurora MySQL. Para obtener información, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

Para conectarse con SSL, use la utilidad MySQL como se describe en el siguiente procedimiento. Si utiliza la autenticación de base de datos de IAM, debe usar una conexión de SSL. Para obtener información, consulte Autenticación de bases de datos de IAM .

> **Note**
>
> Para conectarse al punto de enlace del clúster mediante SSL, la utilidad de conexión del cliente debe ser compatible con los nombres alternativos de firmante (SAN). Si la utilidad

de conexión del cliente no admite SAN, puede conectarse directamente a las instancias del clúster de base de datos de Aurora. Para obtener más información acerca de los puntos de enlace de Aurora, consulte Conexiones de puntos de conexión de Amazon Aurora.

Para conectarse a un clúster de base de datos con SSL a través de la utilidad MySQL

1. Descargue la clave pública para el certificado de firma de Amazon RDS.

   Para obtener más información acerca de cómo descargar certificados, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

2. Escriba el siguiente comando en un símbolo del sistema para conectarse a la instancia principal de un clúster de base de datos con SSL a través de la utilidad MySQL. Para el parámetro `-h`, escriba el nombre de DNS del punto de enlace de la instancia principal. Para el parámetro `-u`, sustituya el ID de usuario de una cuenta de usuario de base de datos. Para el parámetro `--ssl-ca`, utilice el nombre del archivo de certificado de SSL que desee. Escriba la contraseña del usuario maestro cuando se le pida.

   ```
   mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com -u admin_user -p --ssl-ca=[full path]global-bundle.pem --ssl-verify-server-cert
   ```

Debería ver un resultado similar a este.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 8.0.26-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Para obtener instrucciones generales sobre la construcción de cadenas de conexión de RDS for MySQL y encontrar la clave pública para las conexiones SSL, consulte Conexión a una instancia de base de datos que ejecuta el motor de base de datos de MySQL.

Conexión a un clúster de base de datos Amazon Aurora PostgreSQL

Puede conectarse a una instancia de base de datos del clúster de base de datos de Amazon Aurora PostgreSQL con las mismas herramientas que utiliza para conectarse a una base de datos de PostgreSQL. Como parte de este proceso, utiliza la misma clave pública para conexiones de Capa de conexión segura (SSL). Puede usar el punto de enlace y la información de puerto de la instancia principal o las réplicas de Aurora del clúster de base de datos Aurora PostgreSQL en la cadena de conexión de cualquier script, utilidad o aplicación que se conecte a una instancia de base de datos PostgreSQL. En la cadena de conexión, especifique la dirección DNS del punto de enlace de la instancia principal o la réplica de Aurora como parámetro del host. Especifique el número de puerto del punto de enlace como parámetro del puerto.

Cuando tenga una conexión a una instancia de base de datos de su clúster de base de datos de Amazon Aurora PostgreSQL, podrá ejecutar cualquier comando de SQL que sea compatible con PostgreSQL.

En la vista de detalles del clúster de base de datos de Aurora PostgreSQL puede encontrar el nombre del punto de conexión del clúster, el estado, el tipo y el número de puerto. Puede utilizar el punto de conexión y número de puerto en su cadena de conexión de PostgreSQL. Por ejemplo, si un valor de punto de conexión es `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`, debe especificar los siguientes valores en una cadena de conexión de PostgreSQL:

- Para el host o el nombre del host, especifique `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Para el puerto, especifique `5432` o el valor del puerto que utilizó al crear el clúster de base de datos

El punto de enlace de clúster le conecta a la instancia principal del clúster de base de datos. Puede realizar operaciones de lectura y escritura con el punto de enlace de clúster. El clúster de base de datos también puede tener hasta 15 réplicas de Aurora que admiten acceso de solo lectura a los datos de su clúster de base de datos. Cada instancia de base de datos en el clúster de Aurora (esto es, la instancia principal y cada réplica de Aurora) tiene un punto de enlace único independiente del punto de enlace del clúster. Este punto de conexión le permite conectarse a una instancia de base de datos específica directamente en el clúster. El punto de conexión del clúster apunta siempre a la instancia principal. Si se produce un error en la instancia principal y se reemplaza, el punto de conexión del clúster apunta a la nueva instancia principal.

Para ver el punto de enlace del clúster (punto de enlace de escritor), elija Databases (Bases de datos) en la consola de Amazon RDS y elija el nombre del clúster de base de datos para mostrar sus detalles.
Punto de conexión de escritura de Aurora PostgreSQL.

Utilidades de conexión para Aurora PostgreSQL

A continuación se indican algunas de las utilidades de conexión que puede usar:

- Línea de comando: puede conectarse a clústeres de base de datos Aurora PostgreSQL usando herramientas como psql, el terminal interactivo de PostgreSQL. Para obtener más información acerca del uso del terminal interactivo de PostgreSQL, consulte psql en la documentación de PostgreSQL.
- Interfaz gráfica de usuario (GUI): puede usar la utilidad pgAdmin para conectarse a clústeres de base de datos Aurora PostgreSQL por medio de una interfaz de usuario. Para obtener más información, consulte la página Download en el sitio web de pgAdmin.
- Controladores de:AWS
  - Conexión a Aurora PostgreSQL con el controlador JDBC de Amazon Web Services (AWS)
  - Conexión a Aurora PostgreSQL con el controlador de Python de Amazon Web Services (AWS)
  - Conexión a Aurora PostgreSQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)

Conexión a Aurora PostgreSQL con el controlador JDBC de Amazon Web Services (AWS)

El controlador JDBC de Amazon Web Services (AWS) se ha diseñado como un contenedor JDBC avanzado. Este contenedor complementa y amplía la funcionalidad de un controlador JDBC existente para ayudar a las aplicaciones a aprovechar las características de las bases de datos en clúster, como Aurora PostgreSQL. El controlador se admite con el controlador pgJDBC de la comunidad.

Para instalar el controlador JDBC de AWS, añada el archivo .jar del controlador JDBC de AWS (ubicado en la aplicación `CLASSPATH`) y conserve las referencias al controlador de la comunidad pgJDBC. Actualice el prefijo de la URL de conexión de `jdbc:postgresql://` a `jdbc:aws-wrapper:postgresql://`.

Para obtener más información sobre el controlador JDBC de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del controlador JDBC de Amazon Web Services (AWS).

Conexión a Aurora PostgreSQL con el controlador de Python de Amazon Web Services (AWS)

El controlador de Python de Amazon Web Services (AWS) se ha diseñado como un contenedor Python avanzado. Este contenedor complementa y amplía la funcionalidad del controlador de Psycopg de código abierto. El controlador de Python de AWS se admite con las versiones 3.8 y posteriores de Python. Puede instalar el paquete de `aws-advanced-python-wrapper` mediante el comando `pip`, junto con los paquetes de código abierto de `psycopg`.

Para obtener más información sobre el controlador de Python de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del controlador de Python de Amazon Web Services (AWS).

Conexión a Aurora PostgreSQL con el contenedor de NodeJS avanzado de Amazon Web Services (AWS)

El contenedor de NodeJS avanzado de AWS complementa y amplía la funcionalidad de un controlador NodeJS existente. Ayuda a las aplicaciones a aprovechar las características de las bases de datos agrupadas en clústeres como Aurora PostgreSQL.

Para obtener más información sobre el contenedor de NodeJS avanzado de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del contenedor de NodeJS avanzado de Amazon Web Services (AWS).

Solución de errores de conexión de Aurora

Las causas frecuentes de errores de conexión a un nuevo clúster de base de datos de Aurora son las siguientes:

- El grupo de seguridad de la VPC no permite el acceso: su VPC debe permitir conexiones desde el dispositivo o desde una instancia Amazon EC2 mediante la configuración adecuada del grupo de seguridad en la VPC. Para resolverlo, modifique las reglas de entrada del grupo de seguridad de la VPC para permitir conexiones. Para ver un ejemplo, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).
- Puerto bloqueado por reglas de firewall – Compruebe el valor del puerto configurado para el clúster de Aurora base de datos. Si una regla de firewall bloquea ese puerto, puede volver a crear la instancia utilizando un puerto diferente.

- Configuración IAM incompleta o incorrecta – Si ha creado la Aurora instancia de base de datos para utilizar la autenticación IAM–basada, asegúrese de que está configurada correctamente. Para obtener más información, consulte Autenticación de bases de datos de IAM .

Para obtener más información acerca de cómo solucionar problemas de conexión de base de datos Aurora, consulte No puede conectarse a la instancia de base de datos de Amazon RDS.

Grupos de parámetros para Amazon Aurora

Parámetros de la base de datos especificar cómo está configurada la base de datos. Por ejemplo, los parámetros de la base de datos pueden especificar la cantidad de recursos, como la memoria, que se asignarán a una base de datos.

Administre la configuración de la base de datos mediante la asociación de las instancias de base de datos y los clústeres de bases de datos de Aurora con los grupos de parámetros. Aurora define los grupos de parámetros con la configuración predeterminada. También puede definir sus propios grupos de parámetros con una configuración personalizada.

Temas

- Descripción general de los grupos de parámetros
- Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora
- Grupos de parámetros de base de datos para instancias de Amazon Aurora
- Comparación de grupos de parámetros de la base de datos
- Especificación de parámetros de base de datos

Descripción general de los grupos de parámetros

Un grupo de parámetros del clúster de base de datos funciona como un contenedor para los valores de configuración del motor que se aplican a cada instancia de base de datos en un clúster de base de datos de Aurora. Por ejemplo, el modelo de almacenamiento compartido de Aurora requiere que cada instancia de base de datos en un clúster de Aurora utilice la misma configuración para parámetros, como `innodb_file_per_table`. Por lo tanto, los parámetros que afectan al diseño de almacenamiento físico forman parte del grupo de parámetros del clúster. Los grupos de parámetros del clúster de base de datos también incluyen valores predeterminados para todos los parámetros de la instancia.

Un grupo de parámetros de base de datos sirve de contenedor para los valores de configuración del motor que se aplican a una o varias instancias de bases de datos. Los grupos de parámetros de base de datos de se aplican a instancias de base de datos en Amazon RDS y Aurora. Estos ajustes de configuración se aplican a propiedades que pueden variar entre las instancias de base de datos dentro de un clúster de Aurora, como los tamaños de los búferes de memoria.

Temas

- Grupos de parámetros predeterminados y personalizados
- Parámetros de clústeres de base de datos estáticos y dinámicos
- Parámetros de instancias de base de datos estáticos y dinámicos
- Parámetros del conjunto de caracteres
- Parámetros y valores de parámetros admitidos

Grupos de parámetros predeterminados y personalizados

Si crea una instancia de base de datos sin especificar un grupo de parámetros de bases de datos, la instancia de base de datos utilizará un grupo de parámetros de base de datos predeterminado. Del mismo modo, si crea un clúster de base de datos de Aurora sin especificar un grupo de parámetros del clúster de base de datos, el clúster utiliza un grupo de parámetros de clúster de base de datos predeterminado. Cada grupo de parámetros predeterminado contiene los valores predeterminados del motor de base de datos, así como también los valores predeterminados del sistema Amazon RDS correspondientes al motor, la clase de computación y el almacenamiento asignado de la instancia.

La configuración de los parámetros de un grupo de parámetros predeterminado no se puede modificar. En su lugar, puede hacer lo siguiente:

1. Cree un nuevo grupo de parámetros.
2. Cambie la configuración de los parámetros que desee. No todos los parámetros del motor de base de datos pueden cambiarse en el grupo de parámetros.
3. Modifique su instancia o clúster de base de datos para asociar el nuevo grupo de parámetros .

   Para obtener más información sobre la modificación de un clúster de base de datos o instancia de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

> **Note**
>
> Si ha modificado la instancia de base de datos para usar un grupo de parámetros personalizado y la inicia, RDS la reinicia automáticamente como parte del proceso de inicio. Para las instancias Multi-AZ de RDS para SQL Server con las opciones AlwaysOn o Mirroring habilitadas, se espera una conmutación por error cuando la instancia se reinicie tras el proceso de startup.

RDS aplica los parámetros estáticos y dinámicos modificados en un grupo de parámetros recién asociado después de reiniciar la instancia de base de datos. Sin embargo, si modifica los parámetros dinámicos en el grupo de parámetros de base de datos después de asociarlos a la instancia de base de datos, dichos cambios se aplican inmediatamente sin reiniciar. Para obtener información sobre el cambio del grupo de parámetros de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Si actualiza los parámetros en un grupo de parámetros de base de datos, los cambios se aplican a todas las instancias de base de datos que se asocian a ese grupo de parámetros. Del mismo modo, si actualiza los parámetros dentro de un grupo de parámetros del clúster de base de datos de Aurora, los cambios se aplican a todos los clústeres de base de datos de Aurora asociados a ese grupo de parámetros del clúster de base de datos.

Si no desea crear un grupo de parámetros desde cero, puede copiar un grupo de parámetros existente con los comandos de la AWS CLI copy-db-parameter-group o copy-db-clúster-parameter-group. Puede que le resulte útil copiar un grupo de parámetros en algunos casos. Por ejemplo, podría querer incluir la mayoría de los parámetros y valores personalizados de un grupo de parámetros de la existente en un grupo de parámetros de una nueva.

Parámetros de clústeres de base de datos estáticos y dinámicos

Los parámetros de clúster de base de datos son estáticos o dinámicos. Se diferencian por lo siguiente:

- Cuando se cambia un parámetro estático y se guarda el grupo de parámetros de clúster de base de datos, el cambio de parámetro tendrá efecto después de reiniciar manualmente las instancias de base de datos en cada clúster de base de datos asociado. Al utilizar la Consola de administración de AWS para cambiar los valores de los parámetros del clúster de base de datos, siempre se utiliza `pending-reboot` para el `ApplyMethod`.
- Al cambiar un parámetro dinámico, el cambio de parámetros se aplica de forma predeterminada inmediatamente, sin necesidad de reiniciar. Cuando usa la consola, siempre usa `immediate` para `ApplyMethod`. Para aplazar el cambio de parámetros hasta después de reiniciar las instancias de base de datos de un clúster de base de datos asociado, utilice la AWS CLI o la API de RDS. Establezca `ApplyMethod` en `pending-reboot` para el cambio de parámetro.

Para obtener más información acerca del uso de la AWS CLI para cambiar el valor de un parámetro, consulte modify-db-clúster-parameter-group. Para obtener más información acerca del uso de la API de RDS para cambiar un valor de parámetro, consulte ModifyDBclústerParameterGroup.

Si cambia el grupo de parámetros del clúster de base de datos asociado a un clúster de base de datos, reinicie las instancias de base de datos en el clúster de base de datos. En el reinicio, se aplican los cambios a todas las instancias de base de datos del clúster de base de datos. Para determinar si la instancia de base de datos de un clúster de base de datos debe reiniciarse para aplicar los cambios, ejecute el comando AWS CLI que aparece a continuación.

```
aws rds describe-db-clusters --db-cluster-identifier db_cluster_identifier
```

Compruebe el `DBClusterParameterGroupStatus` valor de la instancia de base de datos principal en la salida. Si el valor es `pending-reboot`, reinicie las instancias de base de datos del clúster de base de datos.

Parámetros de instancias de base de datos estáticos y dinámicos

Los parámetros de instancia de base de datos son estáticos o dinámicos. Se diferencian en lo siguiente:

- Cuando se cambia un parámetro estático y se guarda el grupo de parámetros de base de datos, el cambio de parámetros se aplicará después de reiniciar manualmente las instancias de base de datos asociadas. Para los parámetros estáticos, La consola siempre utiliza `pending-reboot` para la `ApplyMethod`.
- Al cambiar un parámetro dinámico, el cambio de parámetros se aplica de forma predeterminada inmediatamente, sin necesidad de reiniciar. Al utilizar la Consola de administración de AWS para cambiar los valores de parámetros de clúster de base de datos, esta siempre utiliza `immediate` para el `ApplyMethod` para los parámetros dinámicos. Para aplazar el cambio de parámetros hasta después de reiniciar una instancia de base de datos asociada, utilice la AWS CLI o la API de RDS. Establezca `ApplyMethod` en `pending-reboot` para el cambio de parámetro.

Para obtener más información acerca del uso de la AWS CLI para cambiar el valor de un parámetro, consulte modificy-db-parameter-group. Para obtener más información acerca del uso de la API de RDS para cambiar un valor de parámetro, consulte ModifyDBParameterGroup.

Si una instancia de base de datos no está utilizando los últimos cambios de su grupo de parámetros de base de datos asociado, la consola muestra un estado de pending-reboot para el grupo de parámetros de base de datos. Este estado no genera un reinicio automático durante la siguiente ventana de mantenimiento. Para aplicar los cambios de parámetros más recientes en esa instancia de base de datos, reinicie manualmente la instancia de base de datos.

Parámetros del conjunto de caracteres

Antes de crear clúster, establezca en su grupo de parámetros cualquier parámetro relacionado con el conjunto de caracteres o la intercalación de su base de datos. Hágalo también antes de crear una base de datos en él. De este modo, garantiza que la base de datos predeterminada y las bases de datos nuevas utilicen el conjunto de caracteres y los valores de intercalación que especifique. Si cambia los parámetros de la intercalación o del conjunto de caracteres, los cambios de parámetros no se aplicarán a las bases de datos existentes.

Para algunos motores de base de datos, puede cambiar los valores de la intercalación o del conjunto de caracteres para una base de datos existente mediante el comando `ALTER DATABASE`; por ejemplo:

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

Para obtener más información acerca de cómo cambiar el conjunto de caracteres o los valores de intercalación de una base de datos, consulte la documentación del motor de la base de datos.

Parámetros y valores de parámetros admitidos

Para determinar los parámetros compatibles con su motor de base de datos, consulte los parámetros en el grupo de parámetros de base de datos y el grupo de parámetros de clúster de base de datos utilizado por el clúster de base de datos o la instancia de base de datos. Para obtener más información, consulte Visualización de los valores de parámetros de un grupo de parámetros de base de datos en Amazon Aurora y Visualización de los valores de parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

En muchos casos, puede especificar valores de parámetros de enteros y booleanos mediante expresiones, fórmulas y funciones. Las funciones pueden incluir una expresión logarítmica matemática. Sin embargo, no todos los parámetros admiten expresiones, fórmulas y funciones para valores de parámetros. Para obtener más información, consulte Especificación de parámetros de base de datos.

Para una base de datos global de Aurora, puede especificar distintos ajustes de configuración para los clústeres de Aurora individuales. Asegúrese de que la configuración sea lo suficientemente similar como para producir un comportamiento coherente si promueve un clúster secundario para convertirlo en el principal. Por ejemplo, utilice la misma configuración de zonas horarias y conjuntos de caracteres en todos los clústeres de una base de datos global de Aurora.

Si los parámetros de un grupo de parámetros se configuran de forma incorrecta, pueden producirse efectos adversos no deseados, como la degradación del rendimiento y la inestabilidad del sistema. Realice siempre cualquier modificación de los parámetros de base de datos con cuidado y haga una copia de seguridad de los datos antes de modificar un grupo de parámetros. Pruebe los cambios de configuración del grupo de parámetros en una instancia de base de datos o en un clúster de base de datos de prueba antes de aplicar dichos cambios a una instancia de base de datos de producción o a un clúster de base de datos.

Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora

Los clústeres de base de datos de Amazon Aurora utilizan los grupos de parámetros de clúster de base de datos. En las secciones siguientes se describe la configuración y administración de los grupos de parámetros de clúster de base de datos.

Temas

- Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos
- Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora
- Asociación de un grupo de parámetros de clúster de base de datos con un clúster de base de datos en Amazon Aurora
- Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora
- Restablecimiento de los parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora
- Copia de un grupo de parámetros de clúster de base de datos en Amazon Aurora
- Enumeración de grupos de parámetros de clúster de base de datos en Amazon Aurora
- Visualización de los valores de parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora
- Eliminación de un grupo de parámetros de clúster de base de datos en Amazon Aurora

Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos

Aurora utiliza un sistema de dos niveles de ajustes de configuración:

- Los parámetros de un grupo de parámetros de clúster de base de datos se aplican a todas las instancias de bases de datos de un clúster de base de datos. Sus datos se almacenan en el subsistema de almacenamiento compartido de Aurora. Debido a esto, todos los parámetros relacionados con el diseño físico de los datos de tabla deben ser los mismos para todas las instancias de base de datos en un clúster de Aurora. De igual forma, puesto que las instancias de base de datos de Aurora están conectadas mediante replicación, todos los parámetros para la configuración de replicación deben ser idénticos en un clúster de Aurora.
- Los parámetros de un grupo de parámetros de base de datos se aplican a una sola instancia de base de datos de un clúster de base de datos de Aurora. Estos parámetros están relacionados con aspectos como el uso de la memoria que puede variar según las instancias de base de datos en el mismo clúster de Aurora. Por ejemplo, un clúster suele contener instancias de base de datos con diferentes clases de instancia de AWS.

Cada clúster de Aurora se asocia a un grupo de parámetros del clúster de base de datos. Este grupo de parámetros asigna valores predeterminados para cada valor de configuración del motor de base de datos correspondiente. Los grupos de parámetros del clúster incluyen valores predeterminados para los parámetros del nivel de instancia y de clúster. Cada instancia de base de datos dentro de un clúster de aprovisionamiento o de Aurora Serverless v2 hereda la configuración del grupo de parámetros del clúster de base de datos.

Cada instancia de base de datos también está asociada a un grupo de parámetros de base de datos. Los valores del grupo de parámetros de base de datos pueden anular los valores predeterminados del grupo de parámetros de clúster. Por ejemplo, si una instancia de un clúster tiene problemas, puede asignar un grupo de parámetros de base de datos personalizado a esa instancia. El grupo de parámetros personalizado puede tener una configuración específica para los parámetros relacionados con la depuración o el ajuste del rendimiento.

Aurora asigna grupos de parámetros predeterminados cuando crea un clúster o una nueva instancia de base de datos según la versión y el motor de base de datos especificados. En su lugar, puede especificar un grupo de parámetros personalizado. Puede crear dichos grupos de parámetros usted mismo y puede editar los valores de parámetros. Puede especificar estos grupos de parámetros personalizados en el momento de la creación. También puede modificar un clúster o instancia de base de datos más adelante para utilizar un grupo de parámetros personalizado.

Para las instancias aprovisionadas y de Aurora Serverless v2, los valores de configuración que modifique en el grupo de parámetros del clúster de base de datos anulan los valores predeterminados del grupo de parámetros de base de datos. Si edita los valores correspondientes

en el grupo de parámetros de base de datos, dichos valores anulan la configuración del grupo de parámetros de clúster de base de datos.

Los ajustes de parámetros de base de datos que modifique tienen preferencia sobre los valores de grupo de parámetros de clúster de base de datos, incluso si devuelve los parámetros de configuración a sus valores predeterminados. Puede ver qué parámetros se sobrescriben mediante el comando de la AWS CLI describe-db-parameters o la operación de la API de RDS DescribeDBParameters. El campo `Source` contiene el valor `user` si modificó ese parámetro. Para restablecer uno o más parámetros de manera que el valor del grupo de parámetros del clúster de base de datos tenga preferencia, utilice el comando de la AWS CLI reset-db-parameter-group o la operación de la API de RDS ResetDBParameterGroup.

Los parámetros de instancia de base de datos y de clúster de base de datos disponible en Aurora varían en función de la compatibilidad del motor de base de datos.

Motor de base de datos	Parámetros
Aurora MySQL	Consulte Parámetros de configuración de Aurora MySQL. Para clústeres de Aurora Serverless, consulte la información adicional en Trabajo con los grupos de parámetros para Aurora Serverless v2 y Grupos de parámetros de Aurora Serverless v1.
Aurora PostgreSQL	Consulte Parámetros de Amazon Aurora PostgreSQL.. Para clústeres de Aurora Serverless, consulte la información adicional en Trabajo con los grupos de parámetros para Aurora Serverless v2 y Grupos de parámetros de Aurora Serverless v1.

> **Note**
>
> Los clústeres de Aurora Serverless v1 solo tienen grupos de parámetros de clúster de base de datos asociados, no grupos de parámetros de base de datos. Para clústeres de Aurora Serverless v2, debe realizar todos los cambios en los parámetros personalizados en el grupo de parámetros de clúster de base de datos.
> Aurora Serverless v2 utiliza los grupos de parámetros de clúster de base de datos y los grupos de parámetros de base de datos. Con Aurora Serverless v2, puede modificar casi todos los parámetros de configuración. Aurora Serverless v2 anula la configuración de

algunos parámetros de configuración relacionados con la capacidad para que la carga de trabajo no se interrumpa cuando se reduzcan las instancias de Aurora Serverless v2. Para obtener más información sobre los ajustes de configuración de Aurora Serverless y los ajustes que puede modificar, consulte Trabajo con los grupos de parámetros para Aurora Serverless v2 y Grupos de parámetros de Aurora Serverless v1.

Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora

Puede crear un nuevo grupo de parámetros de clúster de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Después de crear un grupo de parámetros de clústeres de base de datos, espere al menos 5 minutos antes de crear un clúster de base de datos que utilice ese grupo de parámetros de clúster de base de datos. Esto permite a Amazon RDS crear por completo el grupo de parámetros antes de que lo utilice el nuevo clúster de base de datos. Puede utilizar la página Parameter Groups (Grupos de parámetros) de la consola de Amazon RDS o el comando describe-db-clúster-parameters para comprobar que se ha creado el grupo de parámetros de clúster de base de datos.

Se aplican las siguientes limitaciones al nombre del grupo de parámetros de clústeres de base de datos:

- Debe tener de 1 a 255 letras, números o guiones.

  Los nombres de los grupos de parámetros predeterminados pueden incluir un punto, como `default.aurora-mysql5.7`. Sin embargo, los nombres de grupos de parámetros personalizados no pueden incluir un punto.
- El primer carácter debe ser una letra.
- El nombre no puede incluir dos guiones consecutivos ni finalizar con guion.

Consola

Para crear un grupo de parámetros de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. Elija Create parameter group.

4. En Nombre del grupo de parámetros, escriba el nombre del nuevo grupo de parámetros de clúster de bases de datos.
5. En Descripción, escriba una descripción del nuevo grupo de parámetros de clúster de bases de datos.
6. En Tipo de motor, elija el motor de base de datos.
7. En Familia del grupo de parámetros, seleccione una familia de grupo de parámetros de base de datos.
8. Seleccione Create (Crear).

AWS CLI

Para crear un grupo de parámetros de clúster de base de datos, use el comando AWS CLI de `create-db-cluster-parameter-group`.

En el siguiente ejemplo, se crea un grupo de parámetros de clúster de base de datos denominado mydbclústerparametergroup para la verisón 5.7 de Aurora MySQL con la descripción “My new clúster parameter group” (Mi grupo de parámetros de clúster nuevo).

Incluya los siguientes parámetros obligatorios:

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Para mostrar todas las familias de grupos de parámetros disponibles, use el siguiente comando:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> La salida contiene duplicados.

Example

Para Linux, macOS o Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new cluster parameter group"
```

Para Windows:

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new cluster parameter group"
```

El resultado de este comando debería ser similar al siguiente:

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
    }
}
```

API de RDS

Para crear un grupo de parámetros de clúster de base de datos, use la acción `CreateDBClusterParameterGroup` de la API de RDS.

Incluya los siguientes parámetros obligatorios:

- `DBClusterParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Asociación de un grupo de parámetros de clúster de base de datos con un clúster de base de datos en Amazon Aurora

Puede crear sus propios grupos de parámetros de clúster de base de datos con configuraciones personalizadas. Puede asociar un grupo de parámetros de clúster de base de datos con un clúster

de base de datos mediante la API de Consola de administración de AWS, AWS CLI o RDS. Puede hacerlo al crear o modificar un clúster de base de datos.

Para obtener más información acerca de cómo crear un grupo de parámetros de base de datos, consulte Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora. Para obtener más información acerca de la creación de un clúster de base de datos, consulte Creación de un clúster de base de datos de Amazon Aurora. Para obtener más información acerca de la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

> **Note**
>
> En el caso de Aurora PostgreSQL 15.2, 14.7, 13.10, 12.14 y todas las 11 versiones, al cambiar el grupo de parámetros de clúster de base de datos asociado a un clúster de base de datos, reinicie cada instancia de réplica para aplicar los cambios.
> Para determinar si la instancia de base de datos principal de un clúster de base de datos debe reiniciarse para aplicar los cambios, ejecute el siguiente AWS CLI comando:
> `aws rds describe-db-clusters --db-cluster-identifier` *`db_cluster_identifier`*
> Compruebe el `DBClusterParameterGroupStatus` valor de la instancia de base de datos principal en la salida. Si el valor es `pending-reboot`, reinicie la instancia de base de datos principal del clúster de base de datos.

Consola

Para asociar un grupo de parámetros de clúster de base de datos a un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione el clúster de base de datos que desee modificar.
3. Elija Modify (Modificar). Aparece la página Modify DB clúster (Modificar clúster de base de datos).
4. Cambie la configuración del grupo de parámetros de clúster de base de datos.
5. Elija Continue y consulte el resumen de las modificaciones.

El cambio se aplica inmediatamente independientemente de la configuración Programación de modificaciones .

6. En la página de confirmación, revise los cambios. Si son correctos, elija Modify clúster (Modificar clúster) para guardarlos.

   O bien, elija Back para editar los cambios o Cancel para cancelarlos.

AWS CLI

Para asociar un grupo de parámetros de clúster de base de datos con un clúster de base de datos, utilice el comando `modify-db-cluster` de AWS CLI con las siguientes opciones:

- `--db-cluster-name`
- `--db-cluster-parameter-group-name`

En el ejemplo siguiente asocia el grupo de parámetros de base de datos de `mydbclpg` con el clúster de base de datos de `mydbcluster`.

Example

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --db-cluster-parameter-group-name mydbclpg
```

En:Windows

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --db-cluster-parameter-group-name mydbclpg
```

API de RDS

Para asociar un grupo de parámetros de clúster de base de datos con un clúster de base de datos, utilice la operación `ModifyDBCluster` de la API de RDS con los siguientes parámetros:

- `DBClusterIdentifier`

- `DBClusterParameterGroupName`

Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora

Es posible modificar parámetros de un grupo de parámetros de clúster de base de datos creado por el cliente. No puede cambiar los valores de parámetros de un grupo de parámetros de clúster de base de datos predeterminado. Los cambios realizados en los parámetros de un grupo de parámetros de clúster de base de datos creado por el cliente se aplican a todas las instancias de clústeres de bases de datos asociados al grupo de parámetros de clúster de base de datos.

Consola

Para modificar un grupo de parámetros de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, seleccione el grupo de parámetros que desea modificar.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Edit (Editar).
5. Cambie los valores de los parámetros que desea modificar. Puede desplazarse por los parámetros utilizando las teclas de flecha de la parte superior derecha del cuadro de diálogo.

   No puede cambiar los valores de un grupo de parámetros predeterminado.
6. Elija Save changes.
7. Reinicie la instancia de base de datos principal (de escritura) en el clúster para aplicar los cambios.
8. A continuación, reinicie las instancias de base de datos de lectura para aplicarles los cambios.

   Si no reinicia las instancias de base de datos, una operación de conmutación por error podría tardar más de lo normal.

AWS CLI

Para modificar un grupo de parámetros de clúster de base de datos, utilice el comando `modify-db-cluster-parameter-group` de AWS CLI con los siguientes parámetros obligatorios:

- `--db-cluster-parameter-group-name`
- `--parameters`

En el siguiente ejemplo se modifican los valores de `server_audit_logging` y `server_audit_logs_upload` en el grupo de parámetros de clúster de base de datos denominado mydbclústerparametergroup.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

El comando produce un resultado similar al siguiente:

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

API de RDS

Para modificar un grupo de parámetros de clúster de base de datos, utilice el comando `ModifyDBClusterParameterGroup` de la API de RDS con los siguientes parámetros obligatorios:

- `DBClusterParameterGroupName`
- `Parameters`

Restablecimiento de los parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora

Puede restablecer los parámetros a sus valores predeterminados en un grupo de parámetros de clúster de base de datos creado por el cliente. Los cambios realizados en los parámetros de un grupo de parámetros de clúster de base de datos creado por el cliente se aplican a todas las instancias de clústeres de bases de datos asociados al grupo de parámetros de clúster de base de datos.

> **Note**
>
> En un grupo de parámetros de clúster de base de datos predeterminado, los parámetros siempre se establecen en sus valores predeterminados.

Consola

Para restablecer los parámetros de un grupo de parámetros de clúster de base de datos a sus valores predeterminados

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, elija el grupo de parámetros.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Edit (Editar).
5. Elija los parámetros que desea restablecer a sus valores predeterminados. Puede desplazarse por los parámetros utilizando las teclas de flecha de la parte superior derecha del cuadro de diálogo.

   No puede restablecer los valores de un grupo de parámetros predeterminado.
6. Elija Restablecer y, a continuación, confirme seleccionando Restablecer parámetros.
7. Reinicie la instancia de base de datos principal en el clúster de base de datos para aplicar los cambios a todas las instancias de base de datos del clúster de base de datos.

AWS CLI

Para restablecer los parámetros de un grupo de parámetros de clúster de base de datos a sus valores predeterminados, utilice el comando `reset-db-cluster-parameter-group` de AWS CLI con la siguiente opción requerida: `--db-cluster-parameter-group-name`.

Para restablecer todos los parámetros del grupo de parámetros de clúster de base de datos, especifique la opción `--reset-all-parameters`. Para restablecer parámetros específicos, especifique la opción `--parameters`.

En el ejemplo siguiente se restablecen todos los parámetros del grupo de parámetros DB denominado mydbparametergroup a sus valores predeterminados.

Example

Para Linux, macOS o Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Para Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

En el siguiente ejemplo se modifican los valores de `server_audit_logging` y `server_audit_logs_upload` en el grupo de parámetros de clúster de base de datos denominado mydbclústerparametergroup.

Example

Para Linux, macOS o Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Para Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

El comando produce un resultado similar al siguiente:

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

API de RDS

Para restablecer los parámetros de un grupo de parámetros de clúster de base de datos a sus valores predeterminados, utilice el comando API de RDS `ResetDBClusterParameterGroup` con el siguiente parámetro requerido: `DBClusterParameterGroupName`.

Para restablecer todos los parámetros del grupo de parámetros de clúster de base de datos, defina el parámetro `ResetAllParameters` en `true`. Para restablecer parámetros específicos, especifique el parámetro `Parameters`.

Copia de un grupo de parámetros de clúster de base de datos en Amazon Aurora

Puede copiar los grupos de parámetros de clúster de base de datos personalizados que cree. Copiar un grupo de parámetros es una solución conveniente cuando ya se ha creado un grupo de parámetros de clúster de base de datos y se desea incluir la mayoría de los parámetros y valores personalizados de ese grupo en un nuevo grupo de parámetros de clúster de base de datos. Puede copiar un grupo de parámetros de clúster de base de datos mediante el comando copy-db-clúster-parameter-group de la AWS CLI o la operación CopyDBclústerParameterGroup de la API de RDS.

Después de copiar un grupo de parámetros de clústeres de base de datos, espere al menos 5 minutos antes de crear un clúster de base de datos que utilice ese grupo de parámetros de clúster de base de datos. Esto permite a Amazon RDS copiar por completo el grupo de parámetros antes de que lo utilice el nuevo clúster de base de datos. Puede utilizar la página Parameter Groups (Grupos de parámetros) de la consola de Amazon RDS o el comando describe-db-clúster-parameters para comprobar que se ha creado el grupo de parámetros de clúster de base de datos.

> **Note**
>
> No es posible copiar un grupo de parámetros predeterminado. Sin embargo, puede crear un grupo de parámetros que se base en uno predeterminado.
> No puede copiar un grupo de parámetros de clúster de base de datos en una Cuenta de AWS o Región de AWS diferente.

Consola

Para copiar un grupo de parámetros de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, seleccione el grupo de parámetros personalizado que desea copiar.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Copy (Copiar).
5. En New DB parameter group identifier (Nuevo identificador de grupo de parámetros de base de datos), escriba el nombre del nuevo grupo de parámetros.
6. En Description (Descripción), escriba una descripción para el nuevo grupo de parámetros.
7. Elija Copy.

AWS CLI

Para copiar un grupo de parámetros de clúster de base de datos, utilice el comando `copy-db-cluster-parameter-group` de AWS CLI con los siguientes parámetros obligatorios:

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

En el siguiente ejemplo se crea un nuevo grupo de parámetros de clúster de base de datos denominado `mygroup2` que es una copia del grupo de parámetros de clúster de base de datos `mygroup1`.

Example

Para Linux, macOS o Unix:

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Para Windows:

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

API de RDS

Para copiar un grupo de parámetros de clúster de base de datos, utilice la operación `CopyDBClusterParameterGroup` de la API de RDS con los siguientes parámetros obligatorios:

- `SourceDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupDescription`

Enumeración de grupos de parámetros de clúster de base de datos en Amazon Aurora

Es posible obtener un listado de los grupos de parámetros de clúster de base de datos que se han creado para una cuenta de AWS.

> **Note**
>
> Los grupos de parámetros predeterminados se crean automáticamente a partir de una plantilla de parámetros predeterminados cuando se crea un clúster de base de datos para un motor y una versión de base de datos específicos. Estos grupos de parámetros predeterminados contienen los valores preferidos para los parámetros y no se pueden modificar. Los valores de los parámetros se pueden modificar cuando se crea un grupo de parámetros personalizado.

Consola

Para obtener una lista de todos los grupos de parámetros de clúster de base de datos de una cuenta de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).

   Los grupos de parámetros de clúster de base de datos aparecen en la lista con <guilabel>Grupo de parámetros de clúster de base de datos para Tipo.

AWS CLI

Para obtener la lista de todos los grupos de parámetros de clúster de base de datos para una cuenta de AWS, utilice el comando `describe-db-cluster-parameter-groups` de AWS CLI.

Example

En el siguiente ejemplo, se obtiene la lista de todos los grupos de parámetros de clúster de base de datos disponibles en una cuenta de AWS.

```
aws rds describe-db-cluster-parameter-groups
```

En el siguiente ejemplo se describe el grupo de parámetros mydbclústerparametergroup.

Para Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Para Windows:

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

El comando devuelve una respuesta similar a la siguiente:

```
{
    "DBClusterParameterGroups": [
```

```
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
    ]
}
```

API de RDS

Para obtener la lista de todos los grupos de parámetros de clúster de base de datos de una cuenta de AWS, utilice la acción `DescribeDBClusterParameterGroups` de la API de RDS.

Visualización de los valores de parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora

Es posible obtener una lista de todos los parámetros de un grupo de parámetros de clúster de base de datos y sus valores.

Consola

Para ver los valores de los parámetros de un grupo de parámetros de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).

   Los grupos de parámetros de clúster de base de datos aparecen en la lista con Grupo de parámetros de clúster de base de datos para Tipo.
3. Seleccione el nombre del grupo de parámetros de clúster de base de datos para ver su lista de parámetros.

AWS CLI

Para ver los valores de los parámetros de un grupo de parámetros de clúster de base de datos, utilice el comando `describe-db-cluster-parameters` de AWS CLI con el siguiente parámetro obligatorio.

- `--db-cluster-parameter-group-name`

Example

En el siguiente ejemplo se obtiene la lista de los parámetros y los valores de los parámetros de un grupo de parámetros de clúster de base de datos denominado mydbparametergroup, en formato JSON.

El comando devuelve una respuesta similar a la siguiente:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
        {
            "ParameterName": "aurora_binlog_read_buffer_size",
            "ParameterValue": "5242880",
            "Description": "Read buffer size used by master dump thread when the switch
 aurora_binlog_use_large_read_buffer is ON.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "8192-536870912",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
```

```
...
```

API de RDS

Para ver los valores de los parámetros de un grupo de parámetros de clúster de base de datos, utilice el comando `DescribeDBClusterParameters` de la API de RDS con el siguiente parámetro obligatorio.

- `DBClusterParameterGroupName`

En algunos casos, no se muestran los valores permitidos para un parámetro. Estos son siempre parámetros en los que el origen es el predeterminado del motor de base de datos.

Para ver los valores de estos parámetros, puede ejecutar las siguientes instrucciones SQL:

- MySQL:

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL:

```
-- Show the value of a particular parameter
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

Eliminación de un grupo de parámetros de clúster de base de datos en Amazon Aurora

Puede eliminar un grupo de parámetros de clúster de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS. Un grupo de parámetros de clúster de base de datos solo se puede eliminar si no está asociado a un clúster de base de datos.

Consola

Para eliminar grupos de parámetros

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).

   Los grupos de parámetros aparecen en una lista.
3. Elija el nombre del grupo de parámetros de clúster de base de datos que se va a eliminar.
4. Elija Acciones y, a continuación, elija Eliminar.
5. Revise los nombres de los grupos de parámetros y seleccione Eliminar.

AWS CLI

Para eliminar un grupo de parámetros de clúster de base de datos, utilice el comando `delete-db-cluster-parameter-group` de la AWS CLI con los siguientes parámetros obligatorios:

- `--db-parameter-group-name`

Example

En el siguiente ejemplo, se elimina un grupo de parámetros de clúster de base de datos con el nombre mydbparametergroup.

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

API de RDS

Para eliminar un grupo de parámetros de clúster de base de datos, utilice el comando `DeleteDBClusterParameterGroup` de la API de RDS con los siguientes parámetros obligatorios.

- `DBParameterGroupName`

Grupos de parámetros de base de datos para instancias de Amazon Aurora

Las instancias de base de datos utilizan grupos de parámetros de base de datos. En las secciones siguientes se describe cómo configurar y administrar los grupos de parámetros de instancia de base de datos.

Temas

- Creación de un grupo de parámetros de base de datos en Amazon Aurora
- Asociación de un grupo de parámetros de base de datos con una instancia de base de datos en Amazon Aurora
- Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora
- Restablecimiento de los parámetros de un grupo de parámetros de base de datos a sus valores predeterminados en Amazon Aurora
- Copia de un grupo de parámetros de base de datos en Amazon Aurora
- Enumeración de grupos de parámetros de base de datos en Amazon Aurora
- Visualización de los valores de parámetros de un grupo de parámetros de base de datos en Amazon Aurora
- Eliminación de un grupo de parámetros de base de datos en Amazon Aurora

Creación de un grupo de parámetros de base de datos en Amazon Aurora

Puede crear un nuevo grupo de parámetros de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Se aplican las siguientes limitaciones al nombre del grupo de parámetros de base de datos:

- Debe tener de 1 a 255 letras, números o guiones.

  Los nombres de los grupos de parámetros predeterminados pueden incluir un punto, como `default.mysql8.0`. Sin embargo, los nombres de grupos de parámetros personalizados no pueden incluir un punto.
- El primer carácter debe ser una letra.
- El nombre no puede incluir dos guiones consecutivos ni finalizar con guion.

Consola

Para crear un grupo de parámetros de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. Elija Create parameter group.
4. Para Nombre del grupo de parámetros, escriba el nombre del nuevo grupo de parámetros de base de datos.
5. En Descripción, escriba una descripción del nuevo grupo de parámetros de base de datos.
6. En Tipo de motor, elija el motor de base de datos.
7. En Familia del grupo de parámetros, seleccione una familia de grupo de parámetros de base de datos.
8. En Tipo, elija Grupo de parámetros de base de datos.
9. Seleccione Create (Crear).

AWS CLI

Para crear un grupo de parámetros de base de datos, utilice el comando `create-db-parameter-group` de la AWS CLI. En el siguiente ejemplo se crea un grupo de parámetros de base de datos denominado mydbparametergroup para MySQL versión 8.0 con la descripción “My new parameter group”.

Incluya los siguientes parámetros obligatorios:

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Para mostrar todas las familias de grupos de parámetros disponibles, use el siguiente comando:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> La salida contiene duplicados.

Example

Para Linux, macOS o Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new parameter group"
```

Para Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new parameter group"
```

El resultado de este comando debería ser similar al siguiente:

```
DBPARAMETERGROUP  mydbparametergroup  aurora-mysql5.7  My new parameter group
```

API de RDS

Para crear un grupo de parámetros de base de datos, utilice la operación `CreateDBParameterGroup` de la API de RDS.

Incluya los siguientes parámetros obligatorios:

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Asociación de un grupo de parámetros de base de datos con una instancia de base de datos en Amazon Aurora

Puede crear sus propios grupos de parámetros de base de datos con configuraciones personalizadas. Puede asociar un grupo de parámetros de base de datos con una instancia de base de datos mediante Consola de administración de AWS, la AWS CLI, o la API de RDS. Puede hacerlo al crear o modificar una instancia de base de datos.

Para obtener información sobre la creación de un grupo de parámetros de base de datos, consulte Creación de un grupo de parámetros de base de datos en Amazon Aurora. Para obtener información acerca de la modificación de una instancia de base de datos, consulte Modificación de una instancia de base de datos en un clúster de base de datos.

> **Note**
>
> Al asociar un nuevo grupo de parámetros de base de datos con una instancia de base de datos, los parámetros estáticos y dinámicos modificados se aplican solo después de reiniciar la instancia de base de datos. Sin embargo, si modifica los parámetros dinámicos en el grupo de parámetros de base de datos después de asociarlos a la instancia de base de datos, dichos cambios se aplican inmediatamente sin reiniciar.

Consola

Para asociar un grupo de parámetros de base de datos con una instancia de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos que desee modificar.
3. Elija Modify. Aparece la página Modify DB instance (Modificar instancia de base de datos).
4. Cambie la configuración del grupo de parámetros de base de datos.
5. Elija Continue y consulte el resumen de las modificaciones.
6. (Opcional) Seleccione Apply immediately (Aplicar inmediatamente) para aplicar los cambios inmediatamente. Si se selecciona esta opción, puede producirse una interrupción en algunos casos.

7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify DB instance (Modificar instancia de base de datos) para guardar los cambios.

   O bien, elija Back (Atrás) para editar los cambios o Cancel (Cancelar) para cancelarlos.

AWS CLI

Para asociar un grupo de parámetros de base de datos con una instancia de base de datos, utilice el comando `modify-db-instance` de AWS CLI con las siguientes opciones:

- `--db-instance-identifier`
- `--db-parameter-group-name`

En el ejemplo siguiente se asocia el `mydbpg` grupo de parámetros de base de datos con la `database-1` instancia de base de datos. Los cambios se aplican inmediatamente mediante `--apply-immediately`. Utilícelo `--no-apply-immediately` para aplicar los cambios durante la siguiente ventana de mantenimiento.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

API de RDS

Para asociar un grupo de parámetros de base de datos con una instancia de base de datos, utilice la operación `ModifyDBInstance` de la API de RDS con los siguientes parámetros:

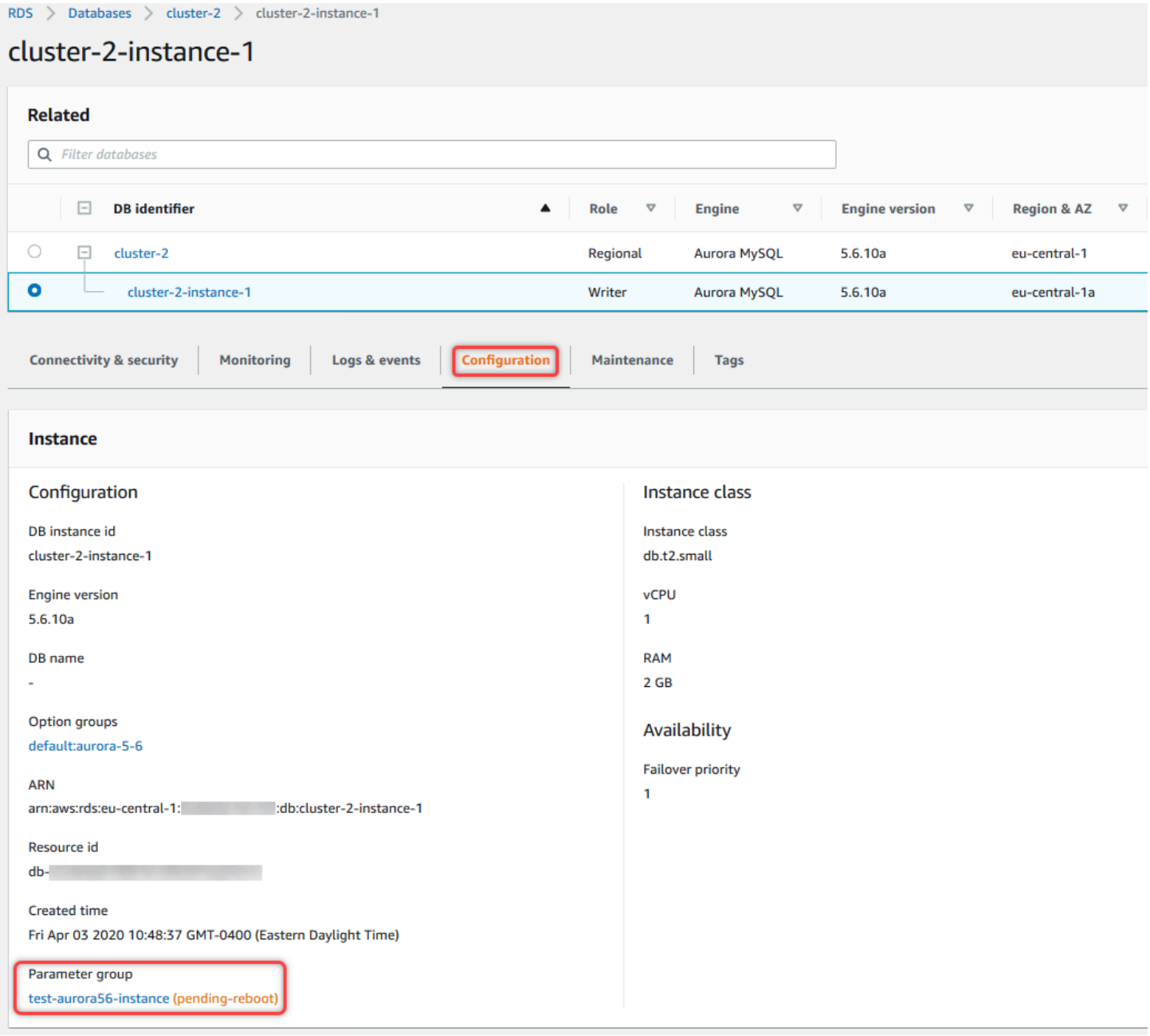
- `DBInstanceName`
- `DBParameterGroupName`

Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora

Es posible modificar los valores de los parámetros de un grupo de parámetros de base de datos creado por el cliente; no es posible modificar los valores de los parámetros de un grupo de parámetros de base de datos predeterminado. Los cambios realizados en los parámetros de un grupo de parámetros de base de datos creado por el cliente se aplican a todas las instancias de bases de datos asociadas al grupo de parámetros de base de datos.

Hay dos tipos de parámetros: parámetros dinámicos y parámetros estáticos. Los cambios en los parámetros dinámicos se aplican a la instancia de base de datos inmediatamente sin necesidad de reiniciar. Los cambios en los parámetros estáticos se aplican solo después de reiniciar la instancia de base de datos.

La consola de RDS muestra el estado del grupo de parámetros de base de datos asociado a una instancia de base de datos en la pestaña Configuration (Configuración). Por ejemplo, si la instancia de base de datos no está utilizando los cambios más recientes del grupo de parámetros de base de datos que tiene asociado, la consola de RDS muestra el grupo de parámetros de base de datos con el estado pending-reboot. Para aplicar los cambios de parámetros más recientes en esa instancia de base de datos, reinicie manualmente la instancia de base de datos.



Consola

Modificación de parámetros en un grupo de parámetros de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, elija el nombre del grupo de parámetros que desea modificar.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Edit (Editar).

5. Cambie los valores de los parámetros que desee modificar. Puede desplazarse por los parámetros utilizando las teclas de flecha de la parte superior derecha del cuadro de diálogo.

   No puede cambiar los valores de un grupo de parámetros predeterminado.

6. Elija Save changes.

AWS CLI

Para modificar un grupo de parámetros de base de datos, utilice el AWS CLI `modify-db-parameter-group` comando con las siguientes opciones requeridas:

- `--db-parameter-group-name`
- `--parameters`

En el siguiente ejemplo se modifican los valores de `max_connections` y `max_allowed_packet` en el grupo de parámetros de base de datos denominado mydbparametergroup.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

El comando produce un resultado similar al siguiente:

```
DBPARAMETERGROUP  mydbparametergroup
```

API de RDS

Para modificar un grupo de parámetros de base de datos, utilice la operación `ModifyDBParameterGroup` de la API de RDS con los siguientes parámetros requeridos:

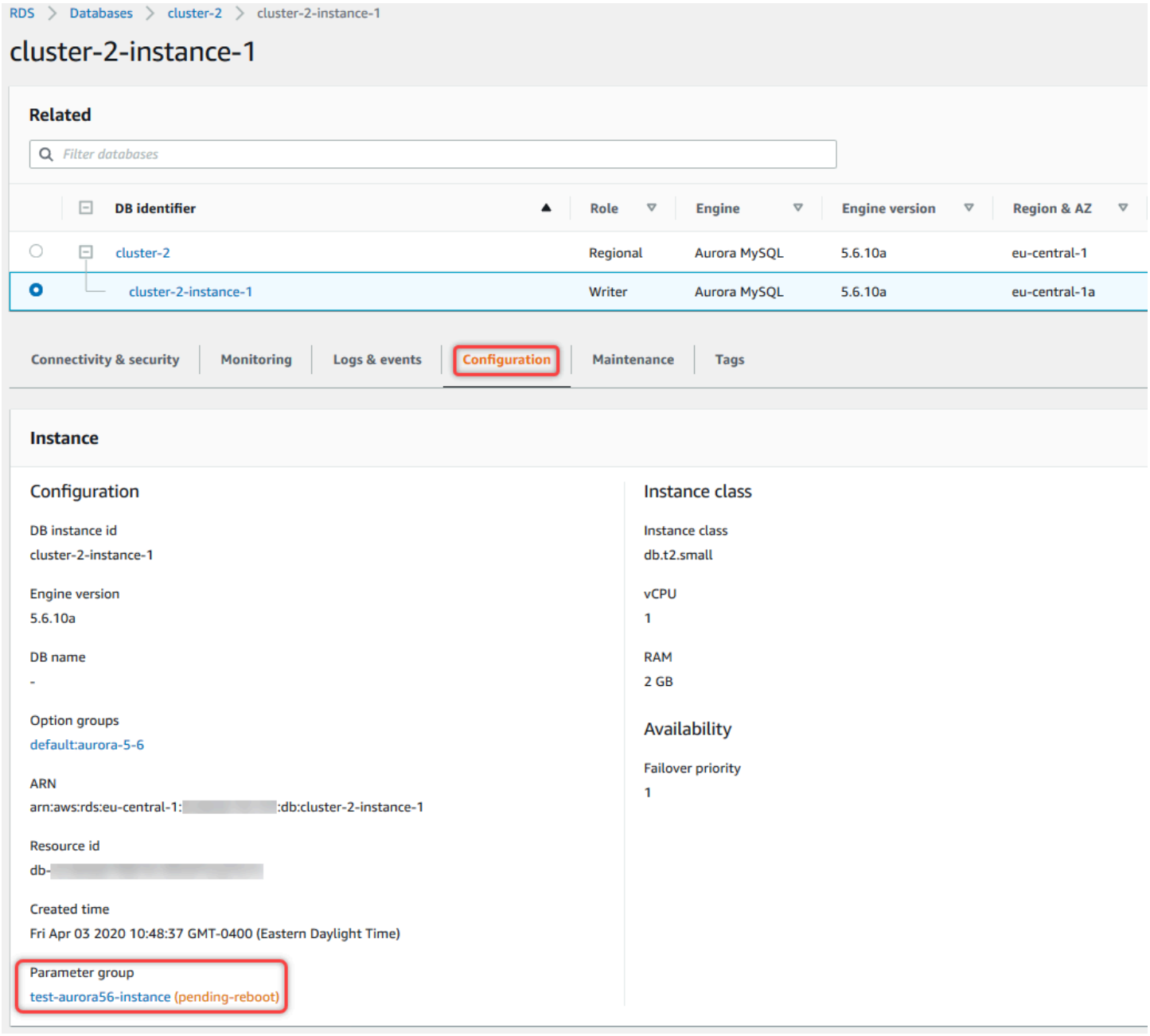
- `DBParameterGroupName`
- `Parameters`

Restablecimiento de los parámetros de un grupo de parámetros de base de datos a sus valores predeterminados en Amazon Aurora

Puede restablecer los valores de los parámetros de un grupo de parámetros de base de datos creado por el cliente a sus valores predeterminados. Los cambios realizados en los parámetros de un grupo de parámetros de base de datos creado por el cliente se aplican a todas las instancias de bases de datos asociadas al grupo de parámetros de base de datos.

Cuando utiliza la consola, puede restablecer parámetros específicos a sus valores predeterminados. Sin embargo, no puede restablecer fácilmente todos los parámetros del grupo de parámetros de base de datos a la vez. Cuando utiliza la AWS CLI o la API de RDS, puede restablecer parámetros específicos a sus valores predeterminados. También puede restablecer fácilmente todos los parámetros del grupo de parámetros de base de datos a la vez.

Los cambios en algunos parámetros se aplican a la instancia de base de datos inmediatamente sin necesidad de reiniciar. Los cambios en otros parámetros se aplican únicamente después de reiniciar la instancia de base de datos. La consola de RDS muestra el estado del grupo de parámetros de base de datos asociado a una instancia de base de datos en la pestaña Configuration (Configuración). Por ejemplo, podría darse por ejemplo que la instancia de base de datos no está utilizando los cambios más recientes del grupo de parámetros de base de datos asociado. De ser así, la consola de RDS muestra el grupo de parámetros de base de datos con el estado pending-reboot. Para aplicar los cambios de parámetros más recientes en esa instancia de base de datos, reinicie manualmente la instancia de base de datos.



> **Note**
>
> En un grupo de parámetros de base de datos predeterminado, los parámetros siempre se establecen en sus valores predeterminados.

Consola

Para restablecer los parámetros de un grupo de parámetros de base de datos a sus valores predeterminados

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, elija el grupo de parámetros.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Edit (Editar).
5. Elija los parámetros que desea restablecer a sus valores predeterminados. Puede desplazarse por los parámetros utilizando las teclas de flecha de la parte superior derecha del cuadro de diálogo.

   No puede restablecer los valores de un grupo de parámetros predeterminado.
6. Elija Restablecer y, a continuación, confirme seleccionando Restablecer parámetros.

AWS CLI

Para restablecer algunos o todos los parámetros de un grupo de parámetros de base de datos, utilice el comando AWS CLI `reset-db-parameter-group` con la siguiente opción requerida: `--db-parameter-group-name`.

Para restablecer todos los parámetros del grupo de parámetros de base de datos, especifique la opción `--reset-all-parameters`. Para restablecer parámetros específicos, especifique la opción `--parameters`.

En el ejemplo siguiente se restablecen todos los parámetros del grupo de parámetros DB denominado mydbparametergroup a sus valores predeterminados.

Example

Para Linux, macOS o Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Para Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

En el ejemplo siguiente se restablecen las opciones `max_connections` y `max_allowed_packet` a sus valores predeterminados en el grupo de parámetros de base de datos denominado mydbparametergroup.

Example

Para Linux, macOS o Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Para Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

El comando produce un resultado similar al siguiente:

```
DBParameterGroupName  mydbparametergroup
```

API de RDS

Para restablecer los parámetros de un grupo de parámetros de base de datos a sus valores predeterminados, utilice el comando `ResetDBParameterGroup` API de RDS con el siguiente parámetro requerido: `DBParameterGroupName`.

Para restablecer todos los parámetros del grupo de parámetros de base de datos, defina el parámetro `ResetAllParameters` en `true`. Para restablecer parámetros específicos, especifique el parámetro `Parameters`.

Copia de un grupo de parámetros de base de datos en Amazon Aurora

Puede copiar los grupos de parámetros de base de datos personalizados que cree. Copiar un grupo de parámetros puede ser una solución práctica. Por ejemplo, podría darse cuando haya creado un grupo de parámetros de base de datos y desee incluir la mayoría de los parámetros y valores personalizados en un nuevo grupo de parámetros de base de datos. Puede copiar un grupo de parámetros de base de datos utilizando la Consola de administración de AWS. También puede utilizar el comando AWS CLI copy-db-parameter-group o la operación CopyDBParameterGroup de la API de RDS.

Después de copiar un grupo de parámetros de base de datos, espere al menos 5 minutos antes de crear la primera instancia de base de datos que utilice ese grupo de parámetros de base de datos como grupo de parámetros predeterminado. Esto permite a Amazon RDS finalizar por completo la acción de copia antes de que se utilice el grupo de parámetros. Esto es especialmente importante para los parámetros que son críticos al crear la base de datos predeterminada de una instancia de base de datos. Un ejemplo es el conjunto de caracteres para la base de datos predeterminada definida por el parámetro `character_set_database`. Utilice la opción Parameter Groups (Grupos de parámetros) de la consola de Amazon RDS o el comando describe-db-parameters para comprobar que se ha creado el grupo de parámetros de base de datos.

> **Note**
>
> No es posible copiar un grupo de parámetros predeterminado. Sin embargo, puede crear un grupo de parámetros que se base en uno predeterminado.
> No puede copiar un grupo de parámetros de base de datos en una Cuenta de AWS o Región de AWS diferente.

Consola

Para copiar un grupo de parámetros de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, seleccione el grupo de parámetros personalizado que desea copiar.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Copy (Copiar).

5. En New DB parameter group identifier (Nuevo identificador de grupo de parámetros de base de datos), escriba el nombre del nuevo grupo de parámetros.
6. En Description (Descripción), escriba una descripción para el nuevo grupo de parámetros.
7. Elija Copy.

AWS CLI

Para copiar un grupo de parámetros de base de datos, utilice el comando AWS CLI de `copy-db-parameter-group` con las siguientes opciones requeridas:

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

En el siguiente ejemplo se crea un nuevo grupo de parámetros de base de datos denominado `mygroup2` que es una copia del grupo de parámetros de base de datos `mygroup1`.

Example

Para Linux, macOS o Unix:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Para Windows:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

API de RDS

Para copiar un grupo de parámetros de base de datos, utilice la operación `CopyDBParameterGroup` de la API de RDS con los siguientes parámetros obligatorios:

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Enumeración de grupos de parámetros de base de datos en Amazon Aurora

Es posible obtener un listado de los grupos de parámetros de base de datos que se han creado para una cuenta de AWS.

> **Note**
>
> Los grupos de parámetros predeterminados se crean automáticamente a partir de una plantilla de parámetros predeterminados cuando se crea una instancia de base de datos para un motor y una versión de base de datos específicos. Estos grupos de parámetros predeterminados contienen los valores preferidos para los parámetros y no se pueden modificar. Los valores de los parámetros se pueden modificar cuando se crea un grupo de parámetros personalizado.

Consola

Para obtener una lista de todos los grupos de parámetros de base de datos de una cuenta de AWS.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).

   Los grupos de parámetros de base de datos aparecen en una lista.

AWS CLI

Para obtener la lista de todos los grupos de parámetros de base de datos para una cuenta de AWS, utilice el comando AWS CLI `describe-db-parameter-groups`.

Example

En el siguiente ejemplo se obtiene la lista de todos los grupos de parámetros de base de datos disponibles en una cuenta de AWS.

```
aws rds describe-db-parameter-groups
```

El comando devuelve una respuesta similar a la siguiente:

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

En el siguiente ejemplo se describe el grupo de parámetros mydbparamgroup1.

Para Linux, macOS o Unix:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Para Windows:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

El comando devuelve una respuesta similar a la siguiente:

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

API de RDS

Para obtener la lista de todos los grupos de parámetros de base de datos de una cuenta de AWS, utilice la operación `DescribeDBParameterGroups` de la API de RDS.

Visualización de los valores de parámetros de un grupo de parámetros de base de datos en Amazon Aurora

Es posible obtener una lista de todos los parámetros de un grupo de parámetros de base de datos y sus valores.

Consola

Para ver los valores de los parámetros de un grupo de parámetros de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).

   Los grupos de parámetros de base de datos aparecen en una lista.
3. Seleccione el nombre del grupo de parámetros para ver su lista de parámetros.

AWS CLI

Para ver los valores de los parámetros de un grupo de parámetros de base de datos, utilice el comando `describe-db-parameters` de la AWS CLI con el siguiente parámetro obligatorio.

- `--db-parameter-group-name`

Example

En el siguiente ejemplo se obtiene la lista de los parámetros y los valores de los parámetros de un grupo de parámetros de base de datos denominado mydbparametergroup.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

El comando devuelve una respuesta similar a la siguiente:

```
DBPARAMETER  Parameter Name           Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                     engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                  engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                     engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size        32768            system           integer
 dynamic     true
DBPARAMETER  socket                   /tmp/mysql.sock  system           string
 static      false
```

API de RDS

Para ver los valores de los parámetros de un grupo de parámetros de base de datos, utilice el comando `DescribeDBParameters` de la API de RDS con el siguiente parámetro obligatorio.

- `DBParameterGroupName`

Eliminación de un grupo de parámetros de base de datos en Amazon Aurora

Puede eliminar un grupo de parámetros de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS. Un grupo de parámetros solo se puede eliminar si no está asociado a una instancia de base de datos.

Consola

Eliminación de un grupo de parámetros de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).

   Los grupos de parámetros de base de datos aparecen en una lista.
3. Elija el nombre del grupo de parámetros que se va a eliminar.
4. Elija Acciones y, a continuación, elija Eliminar.
5. Revise los nombres de los grupos de parámetros y seleccione Eliminar.

AWS CLI

Para eliminar un grupo de parámetros de base de datos, utilice el comando `delete-db-parameter-group` de la AWS CLI con los siguientes parámetros obligatorios:

- `--db-parameter-group-name`

Example

En el siguiente ejemplo, se elimina un grupo de parámetros de base de datos con el nombre mydbparametergroup.

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

API de RDS

Para eliminar un grupo de parámetros de base de datos, utilice la API `DeleteDBParameterGroup` de RDS con los siguientes parámetros obligatorios.

- `DBParameterGroupName`

Comparación de grupos de parámetros de la base de datos

Puede usar la Consola de administración de AWS para ver las diferencias entre dos grupos de parámetros de base de datos.

Los grupos de parámetros especificados deben ser grupos de parámetros de base de datos o ambos deben ser grupos de parámetros de clústeres de base de datos. Esto es cierto incluso si el motor de base de datos y la versión son iguales. Por ejemplo, no puede comparar un grupo de parámetros de base de datos de `aurora-mysql8.0` (Aurora MySQL versión 3) con un grupo de parámetros de clústeres de base de datos de `aurora-mysql8.0`.

Puede comparar los grupos de parámetros de base de datos de Aurora MySQL y RDS para MySQL, incluso para versiones diferentes, pero no puede comparar los grupos de parámetros de base de datos de Aurora PostgreSQL y RDS para PostgreSQL.

Para comparar dos grupos de parámetros de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, seleccione los dos grupos de parámetros que desea comparar.

   > **Note**
   >
   > Para comparar un grupo de parámetros predeterminado con un grupo de parámetros personalizado, primero elija el grupo de parámetros predeterminado en la pestaña Predeterminado y, a continuación, elija el grupo de parámetros personalizado en la pestaña Personalizado.

4. En Acciones, elija Comparar.

Especificación de parámetros de base de datos

Los tipos de parámetros de base de datos incluyen lo siguiente:

- Entero
- Booleano
- Cadena

- Largo
- Doble
- Timestamp
- Objeto de otros tipos de datos definidos
- Matriz de valores de tipo entero, booleano, en cadena, largos, dobles, temporales o de objeto

También puede especificar parámetros de enteros y booleanos mediante expresiones, fórmulas y funciones.

Contenido

- Fórmulas de parámetros de base de datos
  - Variables de las fórmulas de parámetros de base de datos
  - Operadores de las fórmulas de parámetros de base de datos
- Funciones de parámetros de base de datos
- Expresiones de registro de parámetros de base de datos
- Ejemplos de valores de los parámetros de base de datos

Fórmulas de parámetros de base de datos

Una fórmula de parámetros de base de datos es una expresión que da como resultado un valor entero o un valor booleano. Se encierra la expresión entre llaves: {}. Puede especificar fórmulas para el valor de un parámetro de base de datos o como argumento de una función de parámetro de base de datos.

Sintaxis

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

Variables de las fórmulas de parámetros de base de datos

Cada variable de la fórmula devuelve un entero o un valor booleano. Los nombres de las variables distinguen entre mayúsculas y minúsculas.

AllocatedStorage

Devuelve un entero que representa el tamaño, en bytes, del volumen de datos.

DBInstanceClassMemory

Devuelve un entero del número de bytes de memoria disponibles para el proceso de base de datos. Este número se calcula internamente. Para ello, comienza con la cantidad total de memoria de la clase de instancia de base de datos. De esto, el cálculo resta la memoria reservada del sistema operativo y los procesos de RDS que administran la instancia. Por lo tanto, el número siempre es un poco inferior al de las cifras de memoria que se muestran en las tablas de clases de instancia en Clases de instancia de base de datos de Amazon Aurora. El valor exacto depende de una combinación de factores. Estos incluyen la clase de instancia, motor de base de datos y de si aplica a una instancia de RDS o a una instancia que forme parte de un clúster de Aurora.

DBInstanceVCPU

Devuelve un entero que representa el número de unidades de procesamiento centrales virtuales (vCPU) utilizadas por Amazon RDS para administrar la instancia.

EndPointPort

Devuelve un entero que representa el puerto utilizado al conectarse a la instancia de base de datos.

TrueIfReplica

Devuelve 1 si la instancia de base de datos es una réplica de lectura y `0` si no lo es. Es el valor predeterminado del parámetro `read_only` en Aurora MySQL.

Operadores de las fórmulas de parámetros de base de datos

Las fórmulas de parámetros de base de datos admiten dos operadores: división y multiplicación.

Operador de división: /

Divide el dividendo entre el divisor, y devuelve un cociente entero. Los decimales del cociente se truncan, no se redondean.

Sintaxis

```
dividend / divisor
```

Los argumentos del dividendo y el divisor deben ser expresiones enteras.

Operador de multiplicación: *

Multiplica las expresiones, devolviendo el producto de las expresiones. Los decimales de las expresiones se truncan, no se redondean.

Sintaxis

```
expression * expression
```

Las dos expresiones deben dar como resultado valores enteros.

Funciones de parámetros de base de datos

Los argumentos de las funciones de parámetro de base de datos se especifican como enteros o fórmulas. Cada función debe tener un argumento como mínimo. Especifique varios argumentos como una lista separada por comas. La lista no puede tener ningún miembro vacío; por ejemplo, argument1,,argument3. Los nombres de las funciones no distinguen entre mayúsculas y minúsculas.

IF

Devuelve un argumento.

Sintaxis

```
IF(argument1, argument2, argument3)
```

Devuelve el segundo argumento si el primer argumento da como resultado true. En caso contrario, devuelve el tercer argumento.

GREATEST

Devuelve el valor más grande de una lista de números enteros o fórmulas de parámetros.

Sintaxis

```
GREATEST(argument1, argument2,...argumentn)
```

Devuelve un número entero.

LEAST

Devuelve el valor más pequeño de una lista de números enteros o fórmulas de parámetros.

Sintaxis

```
LEAST(argument1, argument2,...argumentn)
```

Devuelve un número entero.

SUM

Suma los valores de los números enteros o fórmulas de parámetros especificados.

Sintaxis

```
SUM(argument1, argument2,...argumentn)
```

Devuelve un número entero.

Expresiones de registro de parámetros de base de datos

Puede establecer un valor de parámetro de base de datos entero en una expresión de registro. Se encierra la expresión entre llaves: {}. Por ejemplo:

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

La función `log` representa la base de registro 2. En este ejemplo también se utiliza la variable de fórmula `DBInstanceClassMemory`. Consulte Variables de las fórmulas de parámetros de base de datos.

Ejemplos de valores de los parámetros de base de datos

Estos ejemplos muestran el uso de fórmulas, funciones y expresiones para los valores de los parámetros de base de datos.

> **Warning**
>
> Establecer parámetros incorrectamente en un grupo de parámetros de base de datos puede tener efectos adversos no deseados. Estos pueden incluir el rendimiento degradado y la

inestabilidad del sistema. Tenga cuidado siempre que modifique los parámetros de base de datos y haga una copia de seguridad de los datos antes de modificar el grupo de parámetros de base de datos. Pruebe los cambios de los grupos de parámetros en instancias de bases de datos de prueba, creadas mediante restauraciones a un momento dado, antes de aplicar dichos cambios de grupo de parámetros a las instancias de bases de datos de producción.

Example uso de la función de parámetro de base de datos LEAST

Puede especificar la función `LEAST` en un valor de parámetro de Aurora MySQL `table_definition_cache`. Úsalo para establecer el número de definiciones de tabla que se pueden almacenar en la caché de definiciones con un mínimo de `DBInstanceClassMemory`/393040 o 20 000.

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Migración de datos a un clúster de base de datos de Amazon Aurora

Tiene varias opciones para migrar datos desde una base de datos existente a un clúster de base de datos de Amazon Aurora, en función de la compatibilidad con el motor de base de datos. Las opciones de migración dependen también de la base de datos desde la que se realiza la migración y del tamaño de los datos que se van a migrar.

Migración de datos a un clúster de base de datos de Amazon Aurora MySQL

Puede migrar datos desde una de las siguientes fuentes a un clúster de base de datos de Amazon Aurora MySQL.

- Una instancia de base de datos de RDS for MySQL
- Una base de datos MySQL externa a Amazon RDS
- Una bases de datos que no sea compatible con MySQL

Para obtener más información, consulte Migración de datos a un clúster de base de datos de Amazon Aurora MySQL.

Migración de datos a un clúster de base de datos de Amazon Aurora PostgreSQL

Puede migrar datos desde una de las siguientes fuentes a un clúster de base de datos de Amazon Aurora PostgreSQL.

- Una instancia de base de datos PostgreSQL en Amazon RDS
- Una base de datos que no sea compatible con PostgreSQL

Para obtener más información, consulte Migración de datos a Amazon Aurora con compatibilidad con PostgreSQL.

Creación de una caché de Amazon ElastiCache mediante el uso de ajustes del clúster de base de datos de Aurora

ElastiCache es un servicio de caché en memoria totalmente administrado que proporciona latencias de lectura y escritura de microsegundos que permiten que los casos de uso sean flexibles y en tiempo real. ElastiCache puede ayudarle a acelerar el rendimiento de las aplicaciones y bases de datos. Puede usar ElastiCache como almacén de datos principal para casos de uso que no requieran durabilidad de los datos, como tablas de clasificación de juegos, transmisiones y análisis de datos. ElastiCache ayuda a eliminar la complejidad propia de la implementación y la administración de un entorno de computación distribuido. Para obtener más información, consulte Common ElastiCache Use Cases and How ElastiCache Can Help para Memcached y Common ElastiCache Use Cases and How ElastiCache Can Help para Redis OSS. Puede utilizar la consola de Amazon RDS para crear cachés de ElastiCache.

Puede utilizar Amazon ElastiCache en dos formatos. Puede empezar con una memoria caché sin servidor o diseñar su propio clúster de caché. Si decide diseñar su propio clúster de caché, ElastiCache es compatible con los motores de Memcached y Redis OSS. Si no está seguro de qué motor desea utilizar, consulte Comparing Memcached and Redis OSS. Para obtener más información acerca de Amazon ElastiCache, consulte la Guía del usuario de Amazon ElastiCache.

Temas

- Información general sobre la creación de cachés de ElastiCache con ajustes del clúster de base de datos de Aurora
- Creación de una caché de ElastiCache con ajustes de un clúster de base de datos de Aurora

Información general sobre la creación de cachés de ElastiCache con ajustes del clúster de base de datos de Aurora

Puede crear una caché de ElastiCache desde Amazon RDS con los mismos ajustes de configuración que una instancia de base de datos de RDS recién creada o existente.

Algunos casos de uso para asociar una caché de ElastiCache al clúster de base de datos:

- Puede ahorrar costos y mejorar su rendimiento si utiliza ElastiCache con RDS en lugar de solo RDS.

- Puede utilizar la caché de ElastiCache como almacén de datos principal para las aplicaciones que no requieran durabilidad de los datos. Las aplicaciones existentes que utilizan Redis OSS o Memcached pueden utilizar ElastiCache sin prácticamente ninguna modificación.

Al crear una caché de ElastiCache desde RDS, la caché de ElastiCache hereda los siguientes ajustes del clúster de base de datos de Aurora asociado:

- Ajustes de conectividad de ElastiCache
- Ajustes de seguridad de ElastiCache

También puede configurar los parámetros de configuración de la caché según sus necesidades.

Configuración de ElastiCache en sus aplicaciones

Debe configurar sus aplicaciones para que utilicen cachés de ElastiCache. También puede optimizar y mejorar el rendimiento de las cachés configurando las aplicaciones para que utilicen estrategias de almacenamiento en caché en función de sus requisitos.

- Para acceder a su caché de ElastiCache y comenzar a trabajar, consulte Getting started with ElastiCache (Redis OSS) y Getting started with ElastiCache (Memcached).
- Para obtener más información sobre las estrategias de almacenamiento en caché, consulte Caching strategies and best practices, para Memcached, y Caching strategies and best practices para Redis OSS.
- Para obtener más información sobre la alta disponibilidad en los clústeres de ElastiCache (Redis OSS), consulte High availability using replication groups.
- Puede incurrir en costos relacionados con el almacenamiento de copias de seguridad, la transferencia de datos dentro o entre regiones, o el uso de AWS Outposts. Para obtener más información sobre los precios, consulte Precios de Amazon ElastiCache.

Creación de una caché de ElastiCache con ajustes de un clúster de base de datos de Aurora

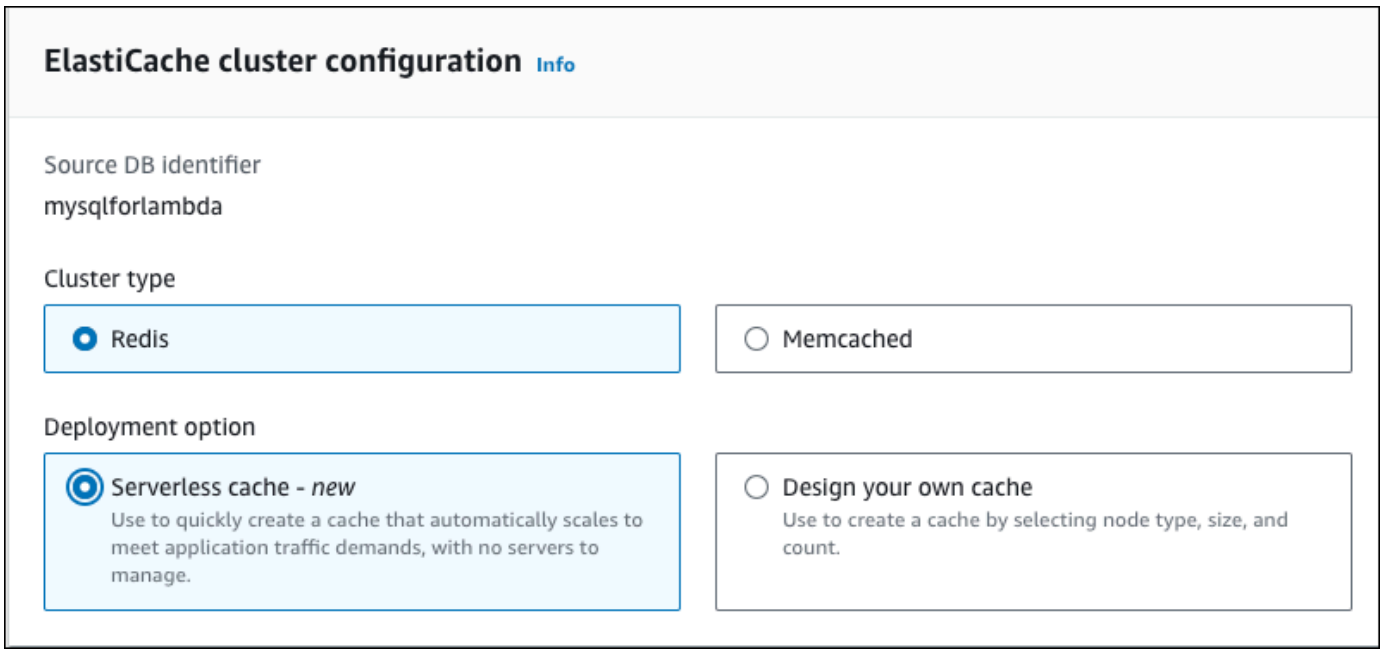
Puede crear una caché de ElastiCache para sus clústeres de base de datos de Aurora con una configuración heredada del clúster de base de datos.

Creación de una caché de ElastiCache con ajustes de un clúster de base de datos

1. Para crear un clúster de base de datos, siga las instrucciones en Creación de un clúster de base de datos de Amazon Aurora.
2. Tras crear un clúster de base de datos, la consola muestra la ventana Complementos sugeridos. Seleccione Crear un clúster de ElastiCache desde RDS con los ajustes de la base de datos.

   Para una base de datos existente, en la página Bases de datos, seleccione el clúster de base de datos que corresponda. En el menú desplegable Acciones, elija Crear clúster de ElastiCache para crear una caché de ElastiCache en RDS que tenga la misma configuración que el clúster de base de datos de Aurora existente.

   En la sección de configuración de ElastiCache, el Identificador de base de datos de origen muestra de qué instancia de base de datos hereda la configuración la caché de ElastiCache.
3. Elija si desea crear un clúster de Redis OSS o Memcached. Para obtener más información, consulte Comparing Memcached and Redis OSS.



4. Después de esto, elija si desea crear una Caché sin servidor o si prefiere Diseñar su propia caché. Para obtener más información, consulte Choosing between deployment options.

   Si elige Caché sin servidor:

   a. En Configuración de caché, introduzca los valores de Nombre y Descripción.

b. En Ver la configuración predeterminada, deje la configuración predeterminada para establecer la conexión entre la caché y el clúster de base de datos.

c. También puede editar la configuración predeterminada seleccionando Personalizar configuración predeterminada. Seleccione Configuración de conectividad de ElastiCache, Configuración de seguridad de ElastiCache y Límites de uso máximo.

5. Si elige Diseñar su propia caché:

a. Si elige Clúster de Redis OSS, elija si desea mantener el modo de clúster Habilitado o Deshabilitado. Para obtener más información, consulte Replication: Redis OSS (clúster Mode Disabled) vs. Redis OSS (clúster Mode Enabled).

b. Introduzca valores para Nombre, Descripción y Versión del motor.

En Versión del motor, el valor predeterminado recomendado es la versión del motor más reciente. También puede elegir la Versión del motor para la caché de ElastiCache que mejor se adapte a sus requisitos.

c. Elija el tipo de nodo en la opción Tipo de nodo. Para obtener más información, consulte Administración de nodos.

Si elige crear un clúster de Redis OSS con el Modo de clúster configurado en Habilitado, introduzca el número de particiones (particiones/grupos de nodos) en la opción Número de particiones.

Introduzca el número de réplicas de cada partición en Número de réplicas.

> **Note**
>
> El tipo de nodo seleccionado, la cantidad de particiones y la cantidad de réplicas afectan al rendimiento de la caché y a los costos de los recursos. Asegúrese de que estos ajustes se correspondan a las necesidades de su base de datos. Para obtener información sobre los precios, consulte Precios de Amazon ElastiCache.

d. Seleccione Configuración de conectividad de ElastiCache y Configuración de seguridad de ElastiCache. Puede conservar la configuración predeterminada o personalizarla según sus necesidades.

6. Compruebe la configuración predeterminada y heredada de su caché de ElastiCache. Algunos ajustes no se pueden cambiar después de la creación.

> **Note**
>
> RDS podría ajustar el periodo de copia de seguridad de la caché de ElastiCache para cumplir con el requisito del periodo mínimo de 60 minutos. El periodo de copia de seguridad de la base de datos de origen sigue siendo el mismo.

7. Cuando esté listo, elija Crear caché de ElastiCache.

La consola abre un banner de confirmación para la creación de la caché de ElastiCache. Siga el enlace del banner a la consola de ElastiCache para ver los detalles de la caché. La consola de ElastiCache muestra la caché de ElastiCache recién creada.

Migración automática de bases de datos de EC2 a Amazon Aurora mediante AWS Database Migration Service

Puede utilizar la consola de Aurora para migrar una base de datos de EC2 a Aurora. Aurora utiliza AWS Database Migration Service (AWS DMS) para migrar bases de datos de EC2 de origen. AWS DMS permite migrar bases de datos relacionales a su nube de AWS. Para obtener más información sobre AWS Database Migration Service, consulte ¿Qué es AWS Database Migration Service? en la Guía del usuario de AWS Database Migration Service.

Para comenzar la migración, debe crear un clúster de bases de datos de Aurora equivalente donde migrar los datos. Tras crear la base de datos de destino, puede importar la base de datos de EC2 a ella. Para las bases de datos de origen de menos de 1 TiB, esta acción de migración reduce el tiempo y los recursos necesarios para migrar los datos a Aurora .

Descripción general

La consola de Aurora le permite migrar bases de datos de EC2 a bases de datos de Aurora equivalentes. Debe crear una base de datos de Aurora para permitir la migración desde la consola.

Puede migrar las bases de datos de EC2 para los siguientes motores de bases de datos:

- MySQL
- PostgreSQL

El proceso de migración consta de los pasos siguientes:

- Cree una base de datos equivalente en Aurora. Para que las bases de datos sean equivalentes, deben tener el mismo motor de base de datos y versiones de motor compatibles. También deben estar en la misma VPC. Para obtener instrucciones sobre cómo crear una base de datos, consulte Creación de un clúster de base de datos de Amazon Aurora .
- Elija el tipo de replicación para la base de datos:
  - Migración a carga completa: Aurora copia la base de datos de origen completa en la base de datos de destino y crea nuevas tablas en la de destino cuando es necesario.

> **Note**
>
> Esta opción provoca una interrupción en la base de datos de Aurora.

- Migración a carga completa y con captura de datos de cambios (CDC): similar a la migración a carga completa, con esta opción, Aurora copia la base de datos de origen completa a la base de datos de destino. Sin embargo, después de la migración a carga completa, Aurora aplica todos los cambios capturados en el origen a la base de datos de destino. La captura de datos de cambios recopila los cambios en los registros de la base de datos mediante la API nativa del motor de la base de datos.

  > **Note**
  >
  > Esta opción provoca una interrupción en la base de datos de Aurora.

- Captura de datos de cambios (CDC): utilice esta opción para mantener la base de datos de destino disponible durante la migración. Aurora migra los cambios continuos de la base de datos de origen a la base de datos de destino.

- Aurora crea los recursos de red necesarios para facilitar la migración. Una vez que Aurora crea los recursos necesarios, le notifica acerca de los recursos creados y le permite iniciar la transferencia de datos.

  El tiempo necesario para completar la migración depende del tipo de replicación y del tamaño de la base de datos de origen.

Requisitos previos

MySQL

Antes de comenzar a trabajar con una base de datos de MySQL como base de datos de origen, asegúrese de cumplir los siguientes requisitos previos. Estos requisitos previos se aplican a orígenes administrados por AWS.

Debe tener una cuenta para AWS DMS que tiene el rol de administrador de replicación. El rol necesita los siguientes privilegios:

- REPLICATION CLIENT: este privilegio es necesario solo para tareas de CDC. Es decir, las tareas de solo carga completa no necesitan este privilegio.
- REPLICATION SLAVE: este privilegio es necesario solo para tareas de CDC. Es decir, las tareas de solo carga completa no necesitan este privilegio.

El usuario de AWS DMS también debe disponer de privilegios SELECT para las tablas de origen designadas para la replicación.

Conceda los siguientes privilegios si utiliza las evaluaciones previas a la migración específicas de MySQL.

```
grant select on mysql.user to <dms_user>;
grant select on mysql.db to <dms_user>;
grant select on mysql.tables_priv to <dms_user>;
grant select on mysql.role_edges to <dms_user>  #only for MySQL version 8.0.11 and
 higher
```

PostgreSQL

Antes de migrar datos desde una base de datos de origen de PostgreSQL administrada por AWS, haga lo siguiente:

- Le recomendamos que utilice una cuenta de usuario de AWS con los permisos mínimos necesarios para la instancia de base de datos de PostgreSQL como cuenta de usuario para el punto de conexión de origen de PostgreSQL para AWS DMS. No se recomienda el uso de la cuenta principal. La cuenta debe tener el rol `rds_superuser` y el rol `rds_replication`. El rol de `rds_replication` concede permisos para administrar ranuras lógicas y para transmitir datos mediante ranuras lógicas.

> **Note**
>
> Algunas transacciones de AWS DMS están inactivas durante un tiempo antes de que el motor de DMS las utilice de nuevo. Al usar el parámetro `idle_in_transaction_session_timeout` en PostgreSQL versiones 9.6 y superiores, puede provocar transacciones inactivas en el tiempo de espera y que se devuelva un error.

Limitaciones

Se aplican las siguientes limitaciones al proceso de migración automática:

- El estado de la base de datos de destino debe ser Disponible para iniciar la migración de la base de datos de origen.

- Al migrar desde una base de datos de MySQL de origen, su cuenta de Aurora debe tener el rol de administrador de replicación. También debe tener los privilegios adecuados aplicados para ese rol.
- La instancia de EC2 y la base de datos de destino deben estar en la misma VPC.
- No puede migrar la base de datos de EC2 a las siguientes bases de datos de destino cuando utiliza la acción Migrar datos desde la base de datos de EC2:
  - Aurora global database
  - Aurora Limitless database
  - Aurora Serverless v1
  - Bases de datos con una versión de MySQL inferior a la 5.7
  - Bases de datos con una versión de PostgreSQL inferior a la 10.4

Creación de recursos de IAM para migraciones homogéneas

Aurora utiliza AWS DMS para migrar sus datos. Para acceder a las bases de datos y para migrar los datos, AWS DMS crea un entorno sin servidor para migraciones de datos homogéneas. En este entorno, AWS DMS requiere acceso a la interconexión de VPC, las tablas de enrutamiento, los grupos de seguridad y otros recursos de AWS. Además, AWS DMS almacena los registros, las métricas y el progreso de cada migración de datos en Amazon CloudWatch. Para crear un proyecto de migración de datos, AWS DMS necesita acceder a estos servicios.

Además, AWS DMS requiere acceso a los secretos que representan un conjunto de credenciales de usuario para autenticar la conexión de base de datos, tanto la de origen como la de destino.

> **Note**
>
> Con la acción Migrar datos de una instancia de EC2, puede utilizar la consola de Aurora para generar estos recursos de IAM. Omita este paso si utiliza los recursos de IAM generados por la consola.

Para este procedimiento, necesita los siguientes recursos de IAM:

Temas

- Creación de una política de IAM para migraciones de datos homogéneas
- Creación de un rol de IAM para migraciones de datos homogéneas
- Creación de un rol y una política de acceso a un secreto

- Creación de un rol de IAM para que AWS DMS administre Amazon VPC

Creación de una política de IAM para migraciones de datos homogéneas

En este paso, se crea una política de IAM que proporciona a AWS DMS acceso a los recursos de Amazon EC2 y CloudWatch. Después, cree un rol de IAM y asocie esta política.

Creación de una política de IAM para una migración de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Políticas.
3. Elija Crear política.
4. En la página Crear política, elija la pestaña JSON.
5. Pegue el siguiente objeto JSON en el editor.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. Elija Siguiente: Etiquetas y Siguiente: Revisar.
7. Ingrese **`HomogeneousDataMigrationsPolicy`** para Nombre* y elija Crear política.

Creación de un rol de IAM para migraciones de datos homogéneas

En este paso, se crea un rol de IAM que proporciona acceso a AWS Secrets Manager, Amazon EC2 y CloudWatch.

Creación de un rol de IAM para migraciones de datos

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles en el panel de navegación.
3. Elija Crear rol.
4. En la página Seleccionar entidad de confianza, para Tipo de entidad de confianza, elija Servicio de AWS. Para Casos de uso para otros servicios de AWS, elija DMS.
5. Seleccione la casilla de verificación DMS y elija Siguiente.
6. En la página Agregar permisos, elija HomogeneousDataMigrationsPolicy que haya creado anteriormente. Elija Siguiente.
7. En la página Asignar nombre, revisar y crear, ingrese **HomogeneousDataMigrationsRole** para Nombre del rol y elija Crear rol.
8. En la página Roles, escriba **HomogeneousDataMigrationsRole** para Nombre del rol. Elija HomogeneousDataMigrationsRole.
9. En la página HomogeneousDataMigrationsRole, elija la pestaña Relaciones de confianza. Elija Editar la política de confianza.
10. En la página Editar política de confianza, pegue el siguiente JSON en el editor y sustituya el texto existente.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

En el ejemplo anterior, sustituya *your_region* por el nombre de la Región de AWS.

La política anterior basada en recursos proporciona a las entidades principales de servicios de AWS DMS permisos para realizar tareas de acuerdo con la política HomogeneousDataMigrationsPolicy administrada por el cliente.

11. Elija Actualizar política.

Creación de un rol y una política de acceso a un secreto

Siga los procedimientos que se indican a continuación para crear su rol y su política de acceso a un secreto que permitan a DMS acceder a las credenciales de usuario de sus bases de datos de origen y destino.

Creación de un rol y una política de acceso a un secreto que permitan a Amazon RDS acceder a AWS Secrets Manager para acceder al secreto pertinente

1. Inicie sesión en la Consola de administración de AWS y abra la consola de AWS Identity and Access Management (IAM) en https://console.aws.amazon.com/iam/.
2. Elija Políticas, después elija Crear política.
3. Elija JSON e ingrese la siguiente política para permitir el acceso al secreto y el descifrado del secreto.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName-ABCDEF"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

Aquí, *secret_arn* es el ARN del secreto, que puede obtener del `SecretsManagerSecretId`, según corresponda, y *kms_key_arn* es el ARN de la clave de AWS KMS que utiliza para cifrar el secreto, como en el siguiente ejemplo.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                      ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> Si utiliza la clave de cifrado predeterminada creada por AWS Secrets Manager, no tiene que especificar los permisos de AWS KMS para *`kms_key_arn`*.
> Si desea que la política proporcione acceso a ambos secretos, simplemente especifique un objeto de recurso JSON adicional para el otro *`secret_arn`*.

4. Revise y cree la política con un nombre descriptivo y una descripción opcional.
5. Elija Roles, después elija Crear rol.
6. Elija Servicio de AWS como tipo de entidad de confianza.
7. Elija DMS de la lista de servicios como servicio de confianza y, a continuación, elija Siguiente: Permisos.
8. Busque y asocie la política que creó en el paso 4 y, a continuación, agregue las etiquetas que desee y revise el rol. En este punto, edite las relaciones de confianza del rol para usar la entidad principal de servicio regional de Amazon RDS como entidad de confianza. Esta entidad principal tiene el formato siguiente.

```
dms.region-name.amazonaws.com
```

Aquí, *`region-name`* es el nombre de la región, por ejemplo `us-east-1`. Por lo tanto, le sigue una entidad principal de servicio regional de Amazon RDS para esta región.

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

Creación de un rol de IAM para que AWS DMS administre Amazon VPC

Debe crear un rol de IAM para que AWS DMS administre la configuración de VPC para sus recursos. Este rol debe estar disponible para que la migración se realice correctamente.

Creación del **`dms-vpc-role`** para la migración de la base de datos

<result>

Esto crea el rol para que DMS administre la configuración de la VPC para la migración.
</result>

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola, elija Roles y, a continuación, seleccione Crear rol.
3. Seleccione la opción Servicio de AWS para la opción Seleccionar entidad de confianza.

   Para Caso de uso, seleccione DMS.
4. Para el paso Agregar permisos, seleccione `AmazonDMSVPCManagementRole` y elija Siguiente.
5. En la página Asignar nombre, revisar y crear, especifique el Nombre del rol como `dms-vpc-role` y elija Crear rol.

Configuración de la migración de datos para bases de datos de EC2

Para empezar a migrar los datos desde la base de datos de EC2 de origen, debe crear una base de datos de Aurora equivalente. Para obtener instrucciones sobre cómo crear una base de datos, consulte Creación de un clúster de base de datos de Amazon Aurora .

Después de crear la base de datos de destino, siga los pasos que se indican a continuación para configurar la migración de datos:

Configuración del proyecto de migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. Elija el menú desplegable Acciones y seleccione la opción Migrar datos desde la base de datos de EC2. Para ver una lista de las bases de datos admitidas, consulte Limitaciones.
3. En la sección Seleccionar la base de datos de EC2 de origen:

   1. Compruebe el Tipo de motor y asegúrese de que sea el mismo que el de la base de datos de origen.

      Compruebe también si las versiones del motor son compatibles.
   2. Para Instancia EC2, elija la instancia de EC2 en la que reside la base de datos de origen.
   3. En Puerto, introduzca el puerto en el que la base de datos de origen permite el tráfico.
   4. En Secreto, seleccione Crear y usar un secreto nuevo si aún no tiene uno. Especifique el Nombre de usuario y la Contraseña para su base de datos de origen. Elija también la clave de KMS con la que quiera cifrar su secreto.

      Si tiene un secreto existente, seleccione Usar secreto existente y elija un secreto de la lista.

5. Para Rol de IAM para el secreto, si ya tiene un rol de IAM existente, seleccione Usar un rol de IAM existente y elija un rol de IAM del menú desplegable que pueda acceder al ID secreto del paso anterior.

   Si no tiene un rol de IAM existente, elija Crear y usar un nuevo rol de IAM. Escriba un nombre para el rol en el campo Nombre del rol de IAM. Puede ver los permisos asociados a este rol en el siguiente enlace.

4. En la sección Ver la base de datos de RDS de destino:

   1. Confirme la configuración de la base de datos de destino en la parte superior de la sección.
   2. En Secreto, seleccione Crear y usar un secreto nuevo si no tiene ya uno que contenga las credenciales de su base de datos de destino.

      Si tiene un secreto existente, seleccione el secreto de la lista desplegable.
   3. En Rol de IAM para el secreto, seleccione un rol de IAM que pueda acceder al secreto en el paso anterior. También puede crear un nuevo rol de IAM si no tiene uno.

      Si el menú desplegable no rellena los roles de IAM, especifique el ARN del rol de IAM en el formato `arn:aws:iam:`*`account_id`*`:role/`*`roleName`*.

5. En la sección Configurar la migración de datos:

   1. Seleccione el tipo de migración de datos entre Carga completa, Carga completa y captura de datos de cambios (CDC) o Captura de datos de cambios (CDC). Para obtener más información sobre estas opciones, consulte Descripción general.

      No puede modificar el tipo de migración una vez iniciada la migración.
   2. Para Rol de IAM para la migración de datos, si ya tiene un rol de IAM existente, seleccione Usar un rol de IAM existente y elija un rol de IAM del menú desplegable que otorgue a DMS los permisos para crear los recursos necesarios para la migración. Si no tiene un rol de IAM existente, elija Crear y usar un nuevo rol de IAM.

6. Confirme que la pestaña Ver la configuración de migración muestra la configuración necesaria para que la migración de datos se configure correctamente.
7. Seleccione Migrar para completar la configuración de la migración.

Tras completar estos pasos, podrá ver los recursos que se están configurando para la migración de datos si selecciona Ver detalles en el panel de progreso de la consola. Una vez configurados los recursos necesarios, la migración se inicia automáticamente. Si crea

Para migrar varias bases de datos a la base de datos de destino, vuelva a iniciar este proceso con los detalles de la nueva base de datos de EC2.

Administración de migraciones de datos

Tras utilizar la acción Migrar datos desde una base de datos de EC2 desde la consola de RDS, Aurora inicia la migración automáticamente.

Si utilizó la consola de AWS DMS para crear los recursos de migración, puede iniciar el proceso de migración.

Inicio de la migración de datos

Siga estos pasos para iniciar la migración de datos:

Inicio de una migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. En la página de detalles de la base de datos, elija la pestaña Migraciones de datos.
3. En la pestaña Migraciones de datos, la sección Migraciones de datos asociadas muestra las migraciones de datos disponibles.

   Las migraciones configuradas mediante la consola Aurora se inician automáticamente una vez configurados los recursos necesarios.

   Las migraciones configuradas mediante la consola de DMS están configuradas como Listas.

   Para iniciar estas migraciones, seleccione el menú desplegable Acciones y, a continuación, seleccione Iniciar.
4. Esto inicia la migración de datos de su base de datos de EC2.

Detención de la migración de datos

En el caso de las migraciones de datos cuyo tipo de replicación es a plena carga, si se detiene la migración, el proceso se para y no se puede reanudar. Una vez detenida la migración, debe reiniciarse.

En el caso de las migraciones con el tipo de replicación configurado como captura de datos de cambio (CDC) o carga completa y CDC, puede detener el proceso de replicación continua y reanudarlo más adelante.

Detención de una migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. En la página de detalles de la base de datos, elija la pestaña Migraciones de datos.
3. En la pestaña Migraciones de datos, la sección Migraciones de datos asociadas muestra las migraciones de datos en curso.

   Para detener una migración, seleccione una migración de datos y elija Detener en el menú desplegable Acciones.
4. Esto detiene la migración de datos de su base de datos de EC2.

Reanudación de la migración de datos

Para las migraciones de datos cuyo tipo de replicación sea carga completa y captura de datos de cambio (CDC) o captura de datos de cambio (CDC), puede reanudar el proceso de CDC desde el último punto de parada.

Reanudación de una migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. En la página de detalles de la base de datos, elija la pestaña Migraciones de datos.
3. En la pestaña Migraciones de datos, la sección Migraciones de datos asociadas muestra las migraciones de datos detenidas.

   Para reanudar una migración, seleccione una migración de datos y elija Reanudar el procesamiento en el menú desplegable Acciones.
4. Esto reanuda la migración de datos de su base de datos de EC2.

Eliminación de la migración de datos

Para eliminar una migración de datos asociada, utilice las siguientes instrucciones

Eliminación de una migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. En la página de detalles de la base de datos, elija la pestaña Migraciones de datos.

3. Para eliminar una migración, seleccione una migración de datos y elija Eliminar en el menú desplegable Acciones.
4. Esto elimina la migración de datos.

La eliminación de una migración de datos que estuviera en curso no afecta a los datos que ya se hayan cargado en la base de datos de destino.

Reinicio de la migración de datos

Para reiniciar una migración de datos asociada desde un punto de inicio de CDC, siga estas instrucciones

Reinicio de una migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. En la página de detalles de la base de datos, elija la pestaña Migraciones de datos.
3. Para reiniciar una migración, seleccione una migración de datos y elija Reiniciar en el menú desplegable Acciones.
4. Esto reinicia la migración de datos desde un punto de inicio de CDC.

El reinicio de una migración de datos que estuviera en curso no afecta a los datos que ya se hayan cargado en la base de datos de destino.

Monitorización de migraciones de datos

Tras iniciar las migraciones de datos, puede monitorizar su estado y su progreso. Las migraciones de grandes conjuntos de datos tardan horas en completarse. Para mantener la fiabilidad, la disponibilidad y el alto rendimiento de la migración de datos, monitoree el progreso con regularidad.

Comprobación del estado y el progreso de la migración de datos

1. Seleccione la base de datos de destino en la página Bases de datos de la consola de RDS.
2. En la página de detalles de la base de datos, elija la pestaña Migraciones de datos.
3. La sección Migraciones de datos asociadas muestra las migraciones de datos. Comprobación de la columna Estado.
4. Para las migraciones de datos en curso, la columna Progreso de migración muestra el porcentaje de datos migrados.

5. Para monitorizar el proceso en CloudWatch, utilice el enlace de la columna CloudWatch.

Estados de migración

Para cada migración de datos que ejecute, la consola de Aurora muestra el Estado. La siguiente lista incluye los estados:

- `Ready`: la migración de datos está lista para comenzar.
- `Starting`: Aurora crea el entorno sin servidor para la migración de datos.
- `Load running`: Aurora realiza la migración de carga completa.
- `Load complete, replication ongoing`: Aurora ha completado la carga completa y ahora replica los cambios en curso. Este estado solo se aplica a las migraciones de carga completa y de tipo CDC.
- `Replication ongoing`: Aurora está replicando los cambios en curso. Este estado solo se aplica a las migraciones de tipo CDC.
- `Stopping`: Aurora está deteniendo las migraciones de datos. Este estado se aplica cuando decide detener la migración de datos desde el menú Acciones.
- `Stopped`: Aurora ha detenido la migración de datos.
- `Failed`: la migración de datos ha producido un error. Para obtener más información, consulte los archivos de registro.
- `Restarting`: la migración de datos ha reiniciado una replicación de datos en curso desde un punto de partida de CDC.

Tutorial: Creación de un clúster de bases de datos de MySQL con un grupo de parámetros personalizados

En este tutorial, creará un clúster de bases de datos de MySQL con un grupo de parámetros personalizados. Para obtener más información sobre los grupos de parámetros , consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

> **Important**
>
> La creación de una cuenta de AWS no supone ningún costo. No obstante, al completar este tutorial, puede incurrir en costos por los recursos de AWS que utilice. Puede eliminar estos recursos después de completar el tutorial si ya no son necesarios.

Para crear un clúster de bases de datos con configuraciones y ajustes personalizados, puede utilizar grupos de parámetros personalizados. Los grupos de parámetros personalizados son especialmente útiles si trabaja con varias bases de datos y desea configurar los ajustes de manera uniforme.

Al completar estos pasos, aprenderá lo siguiente:

- Cómo utilizar Amazon Aurora para crear un clúster de bases de datos de MySQL con un grupo de parámetros personalizados.
- Cómo utilizar determinado parámetros para clústeres de bases de datos de MySQL.

Para completar este tutorial, realice las siguientes tareas:

1. Cree un grupo de parámetros de clúster de bases de datos con el parámetro `default_password_lifetime` de MySQL.
2. Cree un clúster de bases de datos de MySQL con el grupo de parámetros de clúster de bases de datos personalizados que ha creado.

Temas

- Requisitos previos
- Creación de un grupo de parámetros de clúster de bases de datos de Amazon Aurora
- Modificación de el valor de parámetro en el grupo de parámetros personalizados

- Creación de un clúster de bases de datos de MySQL con un grupo de parámetros de clúster de bases de datos

Requisitos previos

Este tutorial requiere que tenga una Cuenta de AWS y un usuario con acceso administrativo. Si aún no los ha configurado, complete los pasos de las secciones siguientes:

- Cómo crear una Cuenta de AWS
- Creación de un usuario con acceso administrativo

Creación de un grupo de parámetros de clúster de bases de datos de Amazon Aurora

En este tutorial, aprenderá a crear un grupo de parámetros personalizados con default_password_lifetime para clúster de bases de datos de MySQL en la consola. El parámetro `default_password_lifetime` controla el número de días que faltan para que la contraseña del cliente caduque automáticamente. Para obtener más información sobre otros parámetros disponibles para clústeres de bases de datos de MySQL, consulte Parámetros de configuración de Aurora MySQL.

Para crear un grupo de parámetros

1. Abra la consola de Amazon RDS y seleccione Grupos de parámetros.
2. En Grupos de parámetros personalizados, elija Crear grupo de parámetros.
3. Establezca los detalles del grupo de parámetros.
    1. Escriba un nombre para el grupo de parámetros.
    2. Introduzca la descripción del grupo de parámetros.
    3. En Tipo de motor, elija Aurora MySQL.
    4. En Familia de grupos de parámetros, elija aurora-mysql8.0.
    5. En Tipo, elija Grupo de parámetros de clúster de bases de datos.
4. Seleccione Crear.

El nuevo grupo de parámetros de clúster de base de datos aparece en la página Grupos de parámetros de la consola de Amazon RDS. Los siguientes pasos ilustran cómo modificar los valores de los parámetros para personalizar el grupo de parámetros.

Modificación de el valor de parámetro en el grupo de parámetros personalizados

Siga estos pasos para modificar el valor de parámetro en el grupo de parámetros que creó en Creación de un grupo de parámetros de clúster de bases de datos de Amazon Aurora.

Modificación de los valores de parámetro en el grupo de parámetros

1. Abra la consola de Amazon RDS y seleccione Grupos de parámetros.
2. En Grupos de parámetros personalizados, elija el nombre del grupo de parámetros del clúster de bases de datos que ha creado.
3. Elija Editar.
4. En el cuadro de búsqueda Filtrar los parámetros, busque el parámetro personalizado `default_password_lifetime`.
5. Seleccione la casilla de verificación situada junto al parámetro e introduzca un valor correspondiente al número de días que se va a establecer para este parámetro de duración de la contraseña.
6. Seleccione Save Changes (Guardar cambios).

El grupo de parámetros personalizado ahora está disponible para asociarlo con Amazon Aurora para el clúster de bases de datos de MySQL 8.0.

Creación de un clúster de bases de datos de MySQL con un grupo de parámetros de clúster de bases de datos

Por último, cree un clúster de bases de datos de MySQL con el grupo de parámetros personalizados que ha creado en los pasos anteriores. Los siguientes pasos muestran cómo crear el clúster de bases de datos de MySQL con el grupo de parámetros personalizados.

Creación de un clúster de bases de datos de MySQL con un grupo de parámetros personalizados y un nuevo grupo de opciones

1. Abra la consola de Amazon RDS y seleccione Bases de datos.

2. Elija Creación de base de datos.
3. En Elegir un método de creación de base de datos, elija Creación estándar.
4. En Opciones del motor, elija Aurora (compatible con MySQL).
5. Seleccione Configuración adicional.
   - En Nombre de base de datos inicial, elija un nombre para el clúster de bases de datos.
   - En la lista desplegable del grupo de parámetros del clúster de bases de datos, seleccione el nombre del grupo de parámetros del clúster de bases de datos que ha creado anteriormente.
6. Para este tutorial, puede dejar la configuración predeterminada para cualquier otro ajuste de la base de datos o modificarla según se requiera.
7. Elija Creación de base de datos.

RDS crea un clúster de bases de datos de MySQL con un grupo de parámetros personalizado. Para obtener más información sobre esta base de datos, consulte la página Bases de datos de la consola de Amazon RDS.

En este tutorial, ha configurado un clúster de bases de datos de MySQL con ajustes personalizados mediante un grupo de parámetros personalizados. Este clúster de bases de datos de MySQL de reciente creación administra la duración de la contraseña del usuario mediante el parámetro `default_password_lifetime`. Para optimizar la base de datos, puede aplicar una configuración adicional al grupo de parámetros personalizados y agregar opciones.

Cuando haya terminado de crear la instancia de base de datos personalizada, debe eliminar los recursos para evitar incurrir en costos no deseados. Para eliminar un clúster de bases de datos, siga las instrucciones que se indican en Eliminación de clústeres e instancias de base de datos de Aurora.

Administración de un clúster de base de datos de Amazon Aurora

En esta sección, se muestra cómo administrar y mantener el clúster de base de datos de Aurora. Aurora funciona en clústeres de servidores de base de datos que están conectados en una topología de reproducción. Por lo tanto, la administración de Aurora suele requerir la implementación de cambios en varios servidores; además, es necesario asegurarse de que todas las réplicas de Aurora mantengan el ritmo del servidor de origen. Dado que Aurora escala transparentemente el almacenamiento subyacente a medida que crecen sus datos, la administración de Aurora requiere relativamente poco esfuerzo administrativo del almacenamiento en disco. Del mismo modo, dado que Aurora realiza automáticamente copias de seguridad continuas, un clúster de Aurora no requiere una planificación extensa ni tiempo de inactividad a la hora de realizarlas.

Temas

- Detención e inicio de un clúster de bases de datos de Amazon Aurora
- Conexión automática de una instancia de EC2 y un clúster de base de datos de Aurora
- Conexión automática de una función de Lambda y un clúster de base de datos de Aurora
- Modificación de un clúster de base de datos de Amazon Aurora
- Adición de réplicas de Aurora a un clúster de base de datos
- Administración del rendimiento y el escalado para clústeres de base de datos Aurora
- Clonación de un volumen de clúster de base de datos de Amazon Aurora
- Integración de Aurora con otros servicios de AWS
- Mantenimiento de un clúster de base de datos de Amazon Aurora
- Reinicio de un clúster de base de datos de Amazon Aurora o de una instancia de base de datos de Amazon Aurora
- Conmutación por error de un clúster de base de datos de Amazon Aurora
- Eliminación de clústeres e instancias de base de datos de Aurora
- Etiquetado de los recursos de Amazon Aurora y Amazon RDS
- Nombres de recursos de Amazon (ARN) en Amazon RDS
- Actualizaciones de Amazon Aurora

Detención e inicio de un clúster de bases de datos de Amazon Aurora

Detener e iniciar los clústeres de bases de datos de Aurora le ayuda a administrar los costos para los entornos de desarrollo y pruebas. Puede detener temporalmente todas las instancias de base de datos su clúster, en lugar de configurar y eliminar todas las instancias de base de datos cada vez que use el clúster.

Temas

- Información general de detención e inicio de un clúster de bases de datos Aurora
- Limitaciones para la detención e inicio de clústeres de base de datos de Aurora
- Detención de un clúster de bases de datos de Aurora
- Posibles operaciones mientras un clúster de bases de datos de Aurora está detenido
- Inicio de un clúster de bases de datos de Aurora

Información general de detención e inicio de un clúster de bases de datos Aurora

Durante los periodos en los que no necesite un clúster de base de datos de Aurora, puede detener todas las instancias de ese clúster a la vez. Puede volver a iniciar el clúster en cualquier momento que necesite usarlo. El inicio y la detención simplifican los procesos de configuración y eliminación de clústeres usados para desarrollo, pruebas o actividades similares que no requieren disponibilidad continua. Puede realizar todos los procedimientos de la Consola de administración de AWS implicados con una sola acción, independientemente de cuántas instancias haya en el clúster.

Mientras su instancia de base de datos esté detenida, solo se le cobrará el almacenamiento del clúster, las instantáneas manuales y el almacenamiento de la copia de seguridad automática dentro de su periodo de retención especificado. No se le cobrarán las horas de ninguna instancia de base de datos.

> **Important**
>
> Puede detener un clúster de bases de datos durante un máximo de siete días. Si no inicia manualmente el clúster de bases de datos después de siete días, el clúster de bases de

datos se inicia automáticamente para que no quede atrás en cuanto a las actualizaciones de mantenimiento necesarias.

Para minimizar los cargos de un clúster de Aurora de carga ligera, puede detener el clúster en lugar de eliminar todas sus réplicas de Aurora. En el caso de clústeres con más de una o dos instancias, eliminar y volver a crear las instancias de base de datos con frecuencia solo es práctico si se usa la AWS CLI o la API de Amazon RDS. Una secuencia de operaciones de ese tipo también puede ser difícil de realizar en el orden correcto, por ejemplo, eliminar todas las réplicas de Aurora antes de eliminar la instancia principal para evitar activar el mecanismo de conmutación por error.

No use la opción de iniciar y detener si necesita mantener su clúster de bases de datos en ejecución pero tiene más capacidad de la que necesita. Si su clúster es demasiado costoso o no está muy ocupado, elimine una o varias de las instancias de base de datos o cambie todas las instancias de base de datos a clase de instancia pequeña. No puede detener una instancia de base de datos Aurora individual.

El tiempo necesario para detener el clúster de base de datos varía en función de factores como las clases de instancias de base de datos, el estado de la red, el tipo de motor de base de datos y el estado de la base de datos. El proceso puede durar varios minutos. El servicio Amazon RDS realiza las siguientes acciones:

- Apaga los procesos del motor de base de datos.
- Apaga los procesos de la plataforma RDS.
- Finaliza las instancias de Amazon EC2 subyacentes.

El tiempo necesario para reiniciar el clúster de base de datos varía en función de factores como el tamaño de la base de datos, las clases de instancias de base de datos, el estado de la red, el tipo de motor de base de datos y el estado de la base de datos en el momento en el que se cerró el clúster. El proceso de inicio puede llegar a tardar horas, pero suele tardar varios minutos. Le recomendamos que tenga en cuenta la variabilidad del tiempo de inicio al crear el plan de disponibilidad.

Para iniciar el clúster de base de datos, el servicio realiza acciones como las siguientes:

- Aprovisiona las instancias de Amazon EC2 subyacentes.
- Inicia los procesos de la plataforma RDS.
- Inicia los procesos del motor de base de datos.

- Recupera las instancias de base de datos (la recuperación ocurre incluso después de un apagado normal).

Limitaciones para la detención e inicio de clústeres de base de datos de Aurora

Algunos clústeres de Aurora no se pueden detener e iniciar:

- Solo puede detener e iniciar un clúster que forma parte de una base de datos global de Aurora si es el único clúster de la base de datos global.
- No puede detener e iniciar un clúster que tenga una réplica de lectura entre regiones.
- No puede detener e iniciar un clúster que forme parte de una implementación azul/verde.
- No se puede detener e iniciar un Aurora Serverless v1 clúster. Con Aurora Serverless v2 no se puede detener ni iniciar el clúster.

Detención de un clúster de bases de datos de Aurora

Para usar un clúster de bases de datos Aurora o realizar tareas de administración, siempre se empieza con un clúster de bases de datos Aurora en ejecución, a continuación, se detiene el clúster y, después, se inicia de nuevo. Mientras su clúster esté detenido, se le cobrará el almacenamiento del clúster, las instantáneas manuales y el almacenamiento de la copia de seguridad automática dentro de su periodo de retención especificado, pero no se le cobrarán las horas de instancia de base de datos.

La operación de detención detiene primero las instancias de réplica de Aurora y, a continuación, la instancia principal, para evitar activar el mecanismo de conmutación por error.

Consola

Para detener un clúster de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija un clúster. Puede realizar la operación de detención desde esta página o navegar a la página de detalles del clúster de bases de datos que desea detener.

3. En Actions (Acciones), elija Stop temporarily (Detener temporalmente).
4. En la ventana Stop DB clúster temporarily (Detener temporalmente el clúster de base de datos), seleccione la confirmación de que el clúster de base de datos se reiniciará automáticamente a los 7 días.
5. Elija Stop temporarily (Detener temporalmente) para detener el clúster de base de datos o elija Cancel (Cancelar) para cancelar la operación.

AWS CLI

Para detener una instancia de base de datos con la AWS CLI, llame al comando stop-db-clúster con los siguientes parámetros:

- `--db-cluster-identifier`: el nombre del clúster de Aurora.

Example

```
aws rds stop-db-cluster --db-cluster-identifier mydbcluster
```

API de RDS

Para detener una instancia de base de datos con la API de Amazon RDS, llame a la operación StopDBclúster con el siguiente parámetro:

- `DBClusterIdentifier`: el nombre del clúster de Aurora.

Posibles operaciones mientras un clúster de bases de datos de Aurora está detenido

Mientras un clúster de Aurora esté detenido, se podrá restaurar a cualquier momento previo que esté especificado en su periodo de retención de copias de seguridad. Para obtener información acerca de cómo hacer una restauración en un momento dado, consulte Restauración de datos.

No se puede modificar la configuración de un clúster de bases de datos Aurora ni de ninguna de sus instancias de base de datos mientras el clúster esté detenido. Tampoco puede añadir ni quitar instancias de base de datos del clúster, ni eliminar el clúster si todavía tiene alguna instancia de base de datos asociada. Debe iniciar el clúster antes de realizar cualquier tarea administrativa de ese tipo.

Al detener un clúster de bases de datos se eliminan las acciones pendientes, excepto para el grupo de parámetros de clúster de bases de datos o para los grupos de parámetros de base de datos de las instancias del clúster de bases de datos.

Aurora aplica cualquier mantenimiento programado a su clúster detenido después de que se vuelva a iniciar. Recuerde que después de siete días Aurora inicia automáticamente cualquier clúster detenido para que no se quede demasiado rezagado en su estado de mantenimiento.

Además, Aurora no realiza copias de seguridad automatizadas porque los datos subyacentes no pueden cambiar mientras el clúster está detenido. Aurora no extiende el periodo de retención de copia de seguridad del clúster de bases de datos mientras está detenido.

Inicio de un clúster de bases de datos de Aurora

Para iniciar un clúster de bases de datos Aurora, siempre debe comenzar con un clúster de Aurora que ya está en estado detenido. Cuando inicia el clúster, todas sus instancias de base de datos se vuelven disponibles otra vez. El clúster mantiene sus ajustes de configuración como puntos de enlace, grupos de parámetros y grupos de seguridad de VPC.

El inicio del clúster de base de datos suele tardar varios minutos.

Consola

Para iniciar un clúster de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija un clúster. Puede realizar la operación de inicio desde esta página o navegar a la página de detalles del clúster de bases de datos que desea iniciar.
3. En Actions (Acciones), elija Start (Iniciar).

AWS CLI

Para iniciar un clúster de bases de datos con la AWS CLI, llame al comando start-db-instance con los siguientes parámetros:

- `--db-cluster-identifier`: el nombre del clúster de Aurora. Este nombre es un identificador de clúster específico que se elige cuando se crea el clúster o el identificador de la instancia de base de datos que eligió con `-cluster` añadido al final.

Example

```
aws rds start-db-cluster --db-cluster-identifier mydbcluster
```

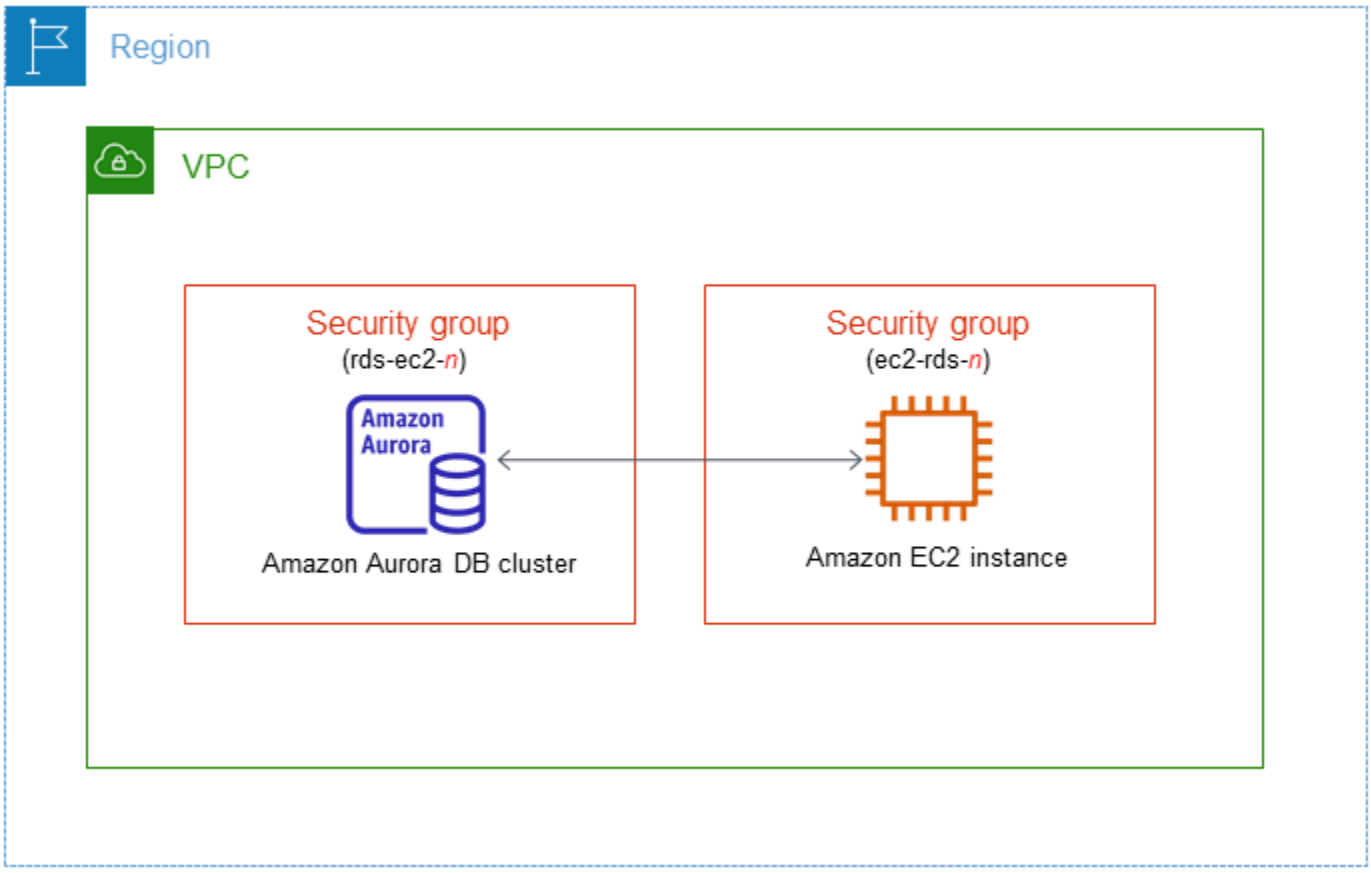
API de RDS

Para iniciar un clúster de bases de datos de Aurora con la API de Amazon RDS, llame a la operación StartDBclúster con el siguiente parámetro:

- `DBCluster`: el nombre del clúster de Aurora. Este nombre es un identificador de clúster específico que se elige cuando se crea el clúster o el identificador de la instancia de base de datos que eligió con `-cluster` añadido al final.

Conexión automática de una instancia de EC2 y un clúster de base de datos de Aurora

Puede utilizar la consola de Amazon RDS para simplificar la configuración de una conexión entre una instancia de Amazon Elastic Compute Cloud (Amazon EC2) y un clúster de base de datos de Aurora. A menudo, el clúster de base de datos se encuentra en una subred privada y la instancia de EC2 en una subred pública dentro de una VPC. Puede usar un cliente SQL en su instancia de EC2 para conectarse al clúster de base de datos. La instancia de EC2 también puede ejecutar servidores web o aplicaciones que accedan al clúster de base de datos privado.



Si desea conectarse a una instancia de EC2 que no esté en la misma VPC que el clúster de base de datos de Aurora, consulte los escenarios en Escenarios de acceso a un clúster de base de datos en una VPC.

Temas

- Descripción general de la conectividad automática con una instancia de EC2
- Conexión automática de una instancia de EC2 y un clúster de base de datos de Aurora

- Visualización de los recursos de computación conectados
- Conexión a una instancia de base de datos que ejecuta un motor de base de datos específico

Descripción general de la conectividad automática con una instancia de EC2

Cuando se configura una conexión entre una instancia EC2 y un clúster de base de datos de Aurora, Amazon RDS configura automáticamente el grupo de seguridad de la VPC para su instancia EC2 y su clúster de base de datos.

Estos son los requisitos para conectar una instancia de EC2 a un clúster de base de datos de Aurora:

- La instancia de EC2 debe existir en la misma VPC que el clúster de base de datos.

  Si no existen instancias de EC2 en la misma VPC, la consola proporciona un enlace para crear una.
- Actualmente, el clúster de base de datos no puede ser un clúster de base de datos de Aurora Serverless ni parte de una base de datos de Aurora.
- El usuario que establece la conectividad debe tener permisos para realizar las siguientes operaciones de Amazon EC2:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Si la instancia de la base de datos y la instancia de EC2 se encuentran en diferentes zonas de disponibilidad, su cuenta podría incurrir en costos cruzados de la zona de disponibilidad.

Cuando se establece una conexión con una instancia de EC2, Amazon RDS realiza una acción basada en la configuración actual de los grupos de seguridad asociados al clúster de base de datos y la instancia de EC2, como se describe en la siguiente tabla.

Configuración del grupo de seguridad de RDS actual	Configuración del grupo de seguridad de EC2 actual	Acción de RDS
Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n* (donde *n* es un número). No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen.	Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `ec2-rds-`*n* (donde *n* es un número). No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad solo tiene una regla de salida con el grupo de seguridad del clúster de base de datos como origen.	RDS no realiza ninguna acción. Ya se configuró automátic amente una conexión entre la instancia de EC2 y el clúster de base de datos. Como ya existe una conexión entre la instancia de EC2 y la base de datos de RDS, los grupos de seguridad no se modifican.
Se aplica alguna de las siguientes condiciones: • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n*. • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n*. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con la instancia de EC2. Amazon RDS no puede usar un grupo de seguridad que no	Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado a la instancia de EC2 con un nombre que coincida con el patrón `ec2-rds-`*n*. • Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `ec2-rds-`*n*. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos. Amazon RDS no puede usar un grupo de seguridad	RDS action: create new security groups

Configuración del grupo de seguridad de RDS actual	Configuración del grupo de seguridad de EC2 actual	Acción de RDS
tenga una regla de entrada con el grupo de seguridad de la VPC de la instancia de EC2 como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado. Los ejemplos de modificaciones incluyen agregar una regla o cambiar el puerto de una regla existente.	que no tenga una regla de salida con el grupo de seguridad de la VPC del clúster de base de datos como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	
Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n*. No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen.	Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `ec2-rds-`*n*. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos. Amazon RDS no puede usar un grupo de seguridad que no tenga una regla de salida con el grupo de seguridad de la VPC del clúster de base de datos como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	RDS action: create new security groups

Configuración del grupo de seguridad de RDS actual	Configuración del grupo de seguridad de EC2 actual	Acción de RDS
Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n*. No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2 como origen.	Existe un grupo de seguridad de EC2 válido para la conexión, pero no está asociado a la instancia de EC2. Este grupo de seguridad tiene un nombre que coincide con el patrón `ec2-rds-`*n*. No se ha modificado. Solo tiene una regla de salida con el grupo de seguridad del clúster de base de datos como origen.	RDS action: associate EC2 security group

Configuración del grupo de seguridad de RDS actual	Configuración del grupo de seguridad de EC2 actual	Acción de RDS
Se aplica alguna de las siguientes condiciones: • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n*. • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-ec2-`*n*. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con la instancia de EC2. Amazon RDS no puede usar un grupo de seguridad que no tenga una regla de entrada con el grupo de seguridad de la VPC de la instancia de EC2 como origen. Amazon RDS tampoco puede usar un grupo de seguridad modificado.	Hay uno o más grupos de seguridad asociados a la instancia de EC2 con un nombre que coincide con el patrón `ec2-rds-`*n*. No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad solo tiene una regla de salida con el grupo de seguridad del clúster de base de datos como origen.	RDS action: create new security groups

Acción de RDS de: crear nuevos grupos de seguridad

Amazon RDS realiza las siguientes acciones:

- Crea un nuevo grupo de seguridad que coincide con el patrón `rds-ec2-`*n*. Este grupo de seguridad tiene una regla de entrada con el grupo de seguridad de VPC de la instancia de EC2

como origen. Este grupo de seguridad que está asociado al clúster de base de datos y permite que la instancia de EC2 acceda al clúster de base de datos.

- Crea un nuevo grupo de seguridad que coincide con el patrón `ec2-rds-`*n*. Este grupo de seguridad tiene una regla de salida con el grupo de seguridad de la VPC del clúster de base de datos como destino. Este grupo de seguridad está asociado a la instancia de EC2 y permite que la instancia de EC2 envíe tráfico al clúster de base de datos.

Acción de RDS de: asociar un grupo de seguridad EC2

Amazon RDS asocia el grupo de seguridad de EC2 válido y existente con la instancia de EC2. Este grupo de seguridad permite que la instancia de EC2 envíe tráfico al clúster de base de datos.

Conexión automática de una instancia de EC2 y un clúster de base de datos de Aurora

Antes de configurar una conexión entre una instancia de EC2 y un clúster de base de datos de Aurora, asegúrese de cumplir con los requisitos descritos en Descripción general de la conectividad automática con una instancia de EC2.

Si realiza cambios en los grupos de seguridad después de configurar la conectividad, los cambios pueden afectar a la conexión entre la instancia de EC2 y el clúster de base de datos de Aurora.

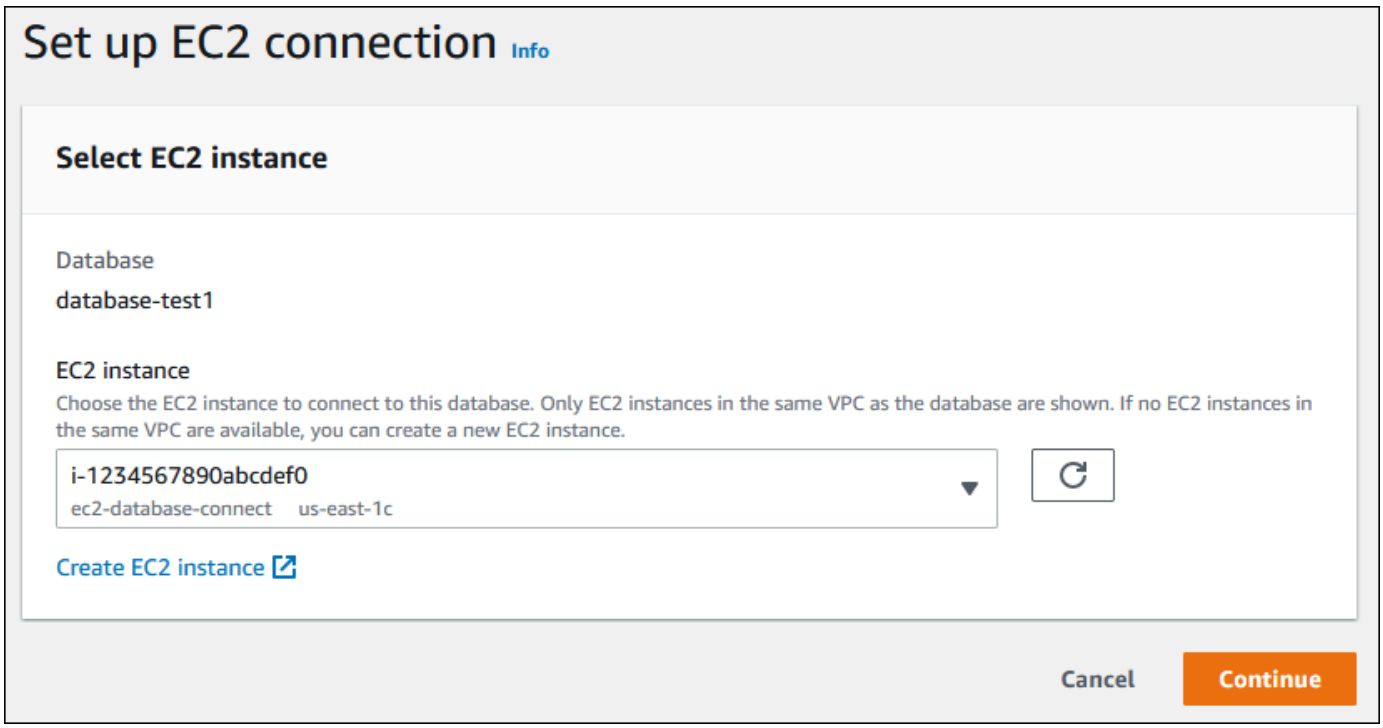
> **Note**
>
> Solo puede configurar automáticamente una conexión entre una instancia de EC2 y un clúster de base de datos de Aurora automáticamente utilizando la Consola de administración de AWS. No puede configurar una conexión automáticamente con la AWS CLI o la API de RDS.

Para conectar automáticamente de una instancia de EC2 y un clúster de base de datos de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, luego, el clúster de bases de datos.
3. En Acciones, elija Configurar conexión de EC2.

Aparece la página Set up EC2 connection Configurar conexión de EC2).

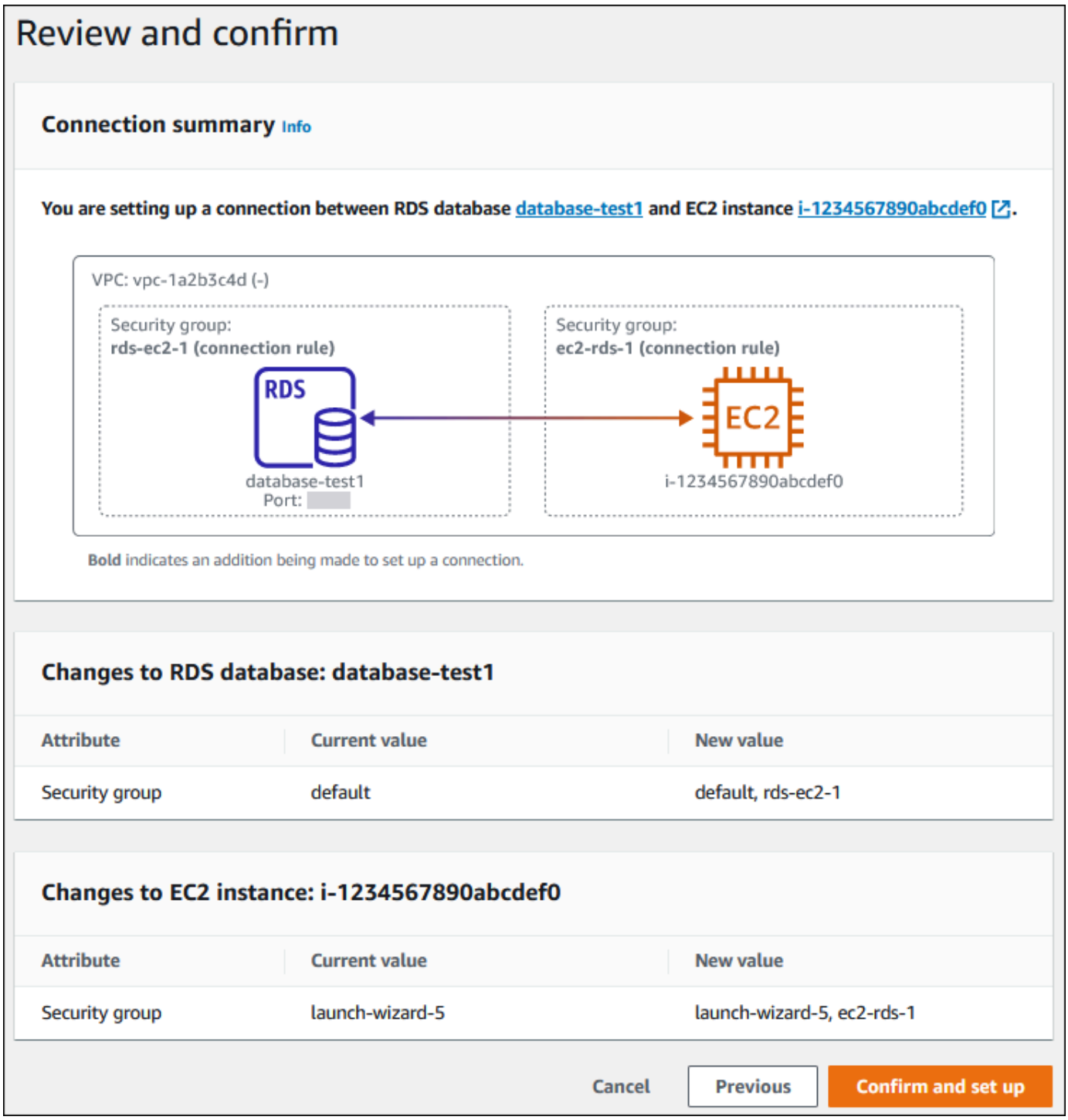
4. En la página Set up EC2 connection (Configurar conexión de EC2), elija la instancia de EC2.



Si no existen instancias de EC2 en la misma VPC, elija Create EC2 instance (Crear instancia de EC2) para crear una. En este caso, asegúrese de que la nueva instancia de EC2 esté en la misma VPC que clúster de base de datos.

5. Elija Continuar.

Aparece la página Review and confirm (Revisar y confirmar).



6. En la página Review and confirm (Revisar y confirmar), revise los cambios que realizará RDS para configurar la conectividad con la instancia de EC2.

   Si los cambios son correctos, seleccione Confirmar y configurar.

   Si los cambios no son correctos, seleccione Previous (Anterior) o Cancel (Cancelar).

Visualización de los recursos de computación conectados

Puede utilizar la Consola de administración de AWS para ver los recursos de computación que están conectados a un clúster de base de datos de Aurora. Los recursos que se muestran incluyen conexiones de recursos informáticos que se configuraron automáticamente. Puede definir la conectividad con los recursos informáticos de manera automática de las siguientes maneras:

- Puede seleccionar el recurso informático al crear la base de datos.

  Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.
- Puede configurar la conectividad entre una base de datos existente y un recurso informático.

  Para obtener más información, consulte Conexión automática de una instancia de EC2 y un clúster de base de datos de Aurora.

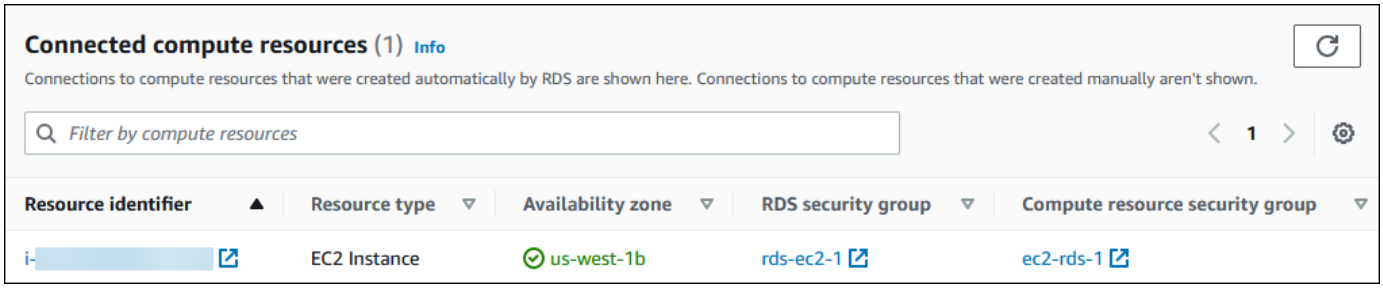
Los recursos informáticos de la lista no incluyen los que se conectaron a la base de datos manualmente. Por ejemplo, puede permitir que un recurso informático acceda a una base de datos manualmente añadiendo una regla al grupo de seguridad de la VPC asociado a la base de datos.

Para que un recurso informático coincida, se deben cumplir las siguientes condiciones:

- El nombre del grupo de seguridad asociado al recurso informático coincide con el patrón `ec2-rds-`*n* (donde *n* es un número).
- El grupo de seguridad asociado al recurso de computación tiene una regla de salida con el rango de puertos establecido en el puerto utilizado por el clúster de base de datos.
- El grupo de seguridad asociado al recurso informático tiene una regla de salida en la que el origen está establecido en un grupo de seguridad asociado al clúster de base de datos.
- El nombre del grupo de seguridad asociados al clúster de base de datos coincide con el patrón `rds-ec2-`*n* (donde *n* es un número).
- El grupo de seguridad asociado al clúster de base de datos tiene una regla de entrada con el rango de puertos establecido en el puerto utilizado por el clúster de base de datos.
- El grupo de seguridad asociado al clúster de base de datos tiene una regla de entrada con el origen establecido en un grupo de seguridad asociado al recurso informático.

Para ver los recursos de computación conectados a un clúster de base de datos de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, luego, el nombre del clúster de bases de datos.
3. En la pestaña Connectivity & security (Conectividad y seguridad), consulte los recursos informáticos en Connected compute resources (Recursos informáticos conectados).



Conexión a una instancia de base de datos que ejecuta un motor de base de datos específico

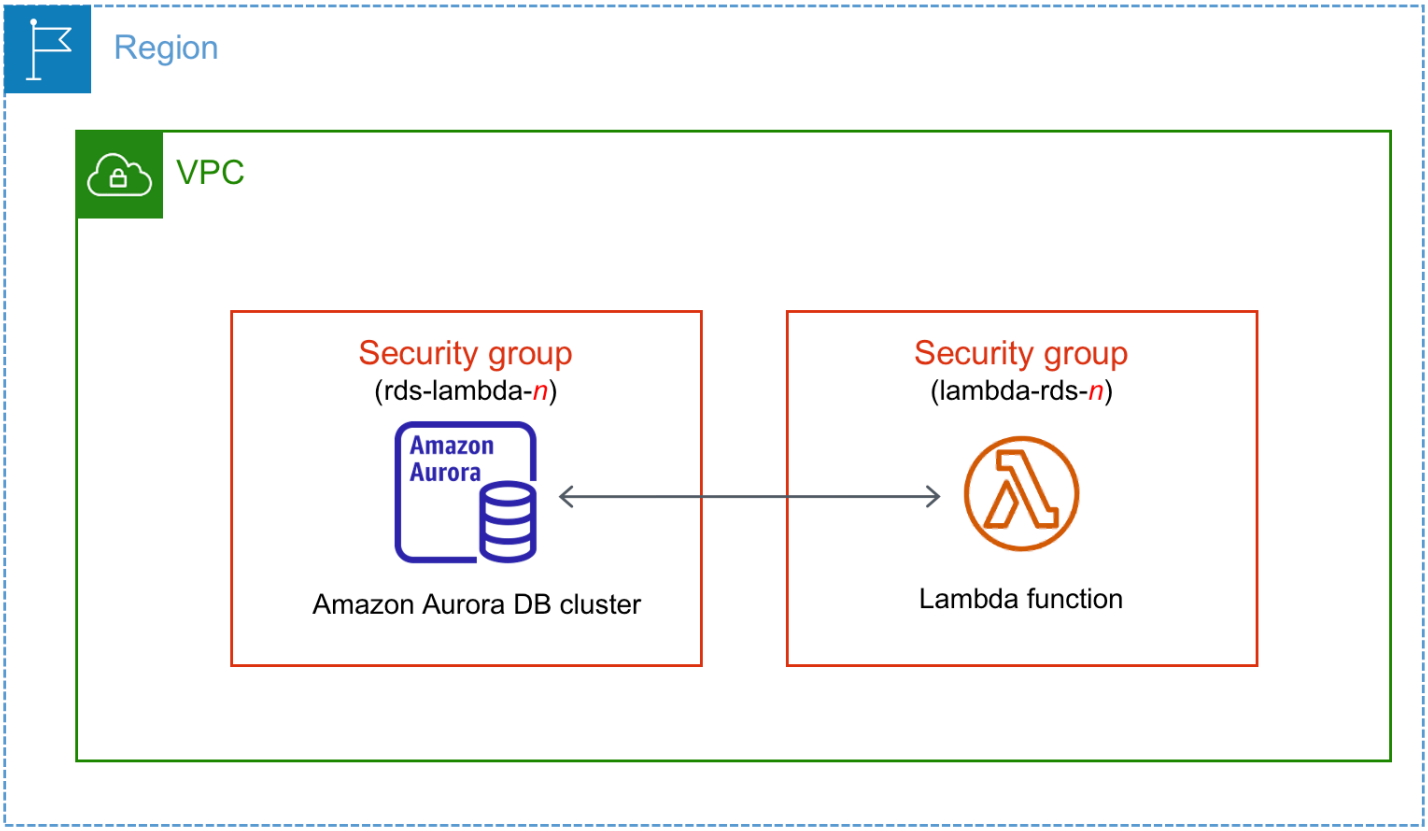
Para obtener información sobre cómo conectarse a una instancia de base de datos que ejecuta un motor de base de datos específico, siga las instrucciones para su motor de base de datos:

- Conexión a un clúster de base de datos Amazon Aurora MySQL
- Conexión a un clúster de base de datos Amazon Aurora PostgreSQL

Conexión automática de una función de Lambda y un clúster de base de datos de Aurora

Puede utilizar la consola de Amazon RDS para simplificar la configuración de una conexión entre una función de Lambda y un clúster de base de datos de Aurora. A menudo, el clúster de base de datos se encuentra en una subred privada dentro de una VPC. Las aplicaciones pueden utilizar la función de Lambda para acceder a su clúster de base de datos privado.

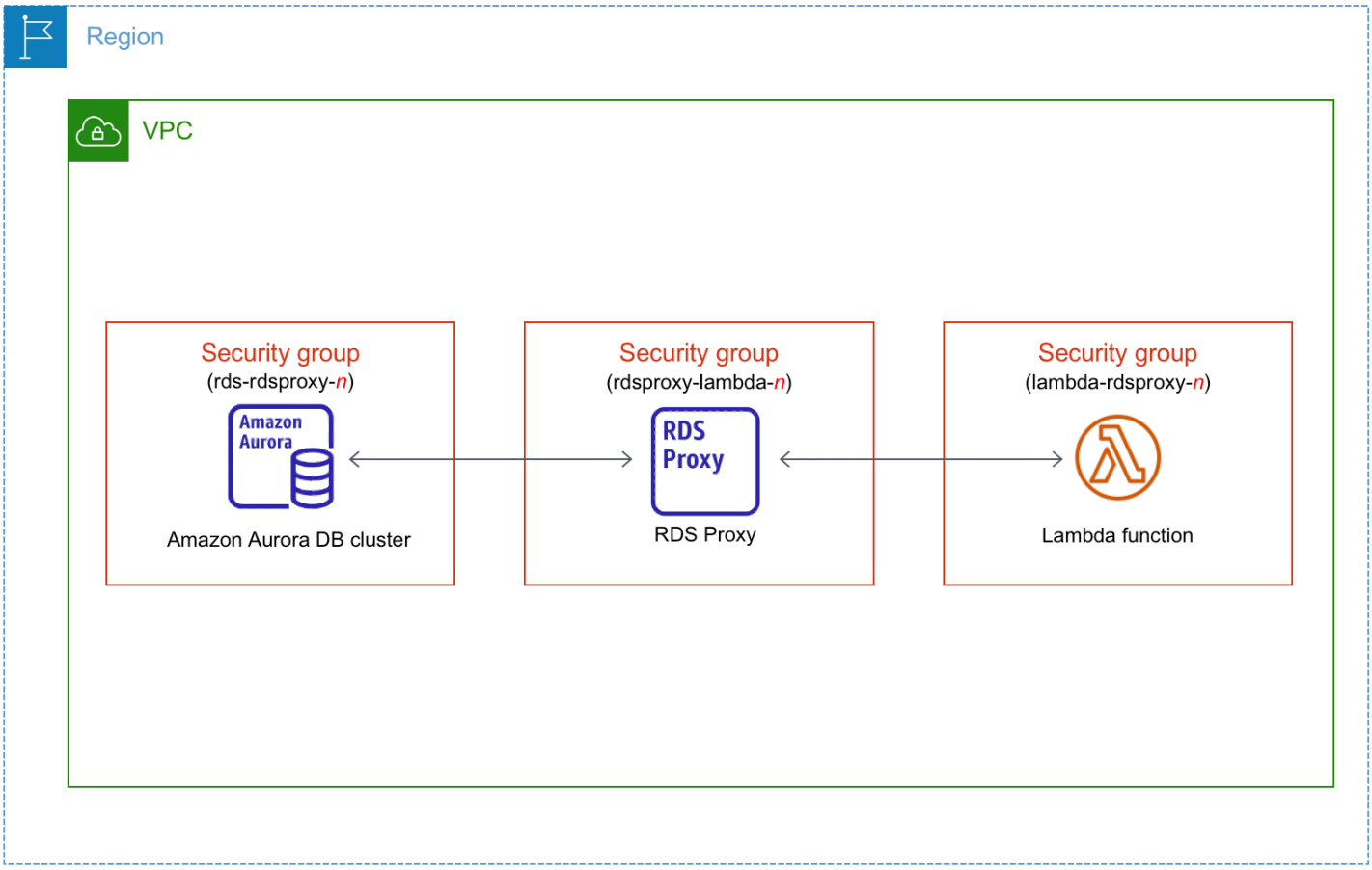
La siguiente imagen muestra una conexión directa entre su clúster de base de datos y su función de Lambda.



Puede configurar la conexión entre la función de Lambda y su clúster de base de datos a través de RDS Proxy para mejorar el rendimiento y la resiliencia de la base de datos. A menudo, las funciones de Lambda hacen frecuentes conexiones cortas a la base de datos que aprovechan el grupo de conexiones que ofrece RDS Proxy. Puede aprovechar cualquier autenticación de AWS Identity and Access Management (IAM) que ya tenga para las funciones de Lambda, en lugar de administrar las credenciales de la base de datos en el código de la aplicación de Lambda. Para obtener más información, consulte Amazon RDS Proxy para Aurora.

Cuando utiliza la consola para conectarse con un proxy existente, Amazon RDS actualiza el grupo de seguridad del proxy para permitir las conexiones desde su clúster de base de datos y la función de Lambda.

También puede crear un nuevo proxy desde la misma página de la consola. Al crear un proxy en la consola, para acceder al clúster de base de datos, debe introducir las credenciales de la base de datos o seleccionar un secreto de AWS Secrets Manager.



Tip

Para conectar rápidamente una función de Lambda a un clúster de base de datos de Aurora, también puede utilizar el asistente guiado integrado en la consola. Para abrir el asistente, haga lo siguiente:

1. Abra la página de Funciones en la consola de Lambda.
2. Seleccione la función a la que desea conectar una base de datos.
3. En la pestaña Configuración, seleccione Bases de datos de RDS.
4. Seleccione Conectar a la base de datos de RDS.

Después de haber conectado la función a una base de datos, podrá crear un proxy. Para ello, elija Agregar proxy.

Temas

- Información general de la conectividad automática con una función de Lambda
- Conexión automática de una función de Lambda y un clúster de base de datos de Aurora
- Visualización de los recursos de computación conectados

Información general de la conectividad automática con una función de Lambda

Estos son los requisitos para conectar una función de Lambda a un clúster de base de datos de Aurora:

- La función de Lambda debe encontrarse en la misma VPC que el clúster de base de datos.
- Actualmente, el clúster de base de datos no puede ser un clúster de base de datos de Aurora Serverless ni parte de una base de datos de Aurora.
- El usuario que configure la conectividad debe tener permisos para realizar las siguientes operaciones de Amazon RDS, Amazon EC2, Lambda, Secrets Manager e IAM:
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBClusters`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBCluster`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`

    - `ec2:DescribeSecurityGroups`
    - `ec2:RevokeSecurityGroupEgress`
    - `ec2:RevokeSecurityGroupIngress`
- Lambda
    - `lambda:CreateFunctions`
    - `lambda:ListFunctions`
    - `lambda:UpdateFunctionConfiguration`
- Secrets Manager
    - `secretsmanager:CreateSecret`
    - `secretsmanager:DescribeSecret`
- IAM
    - `iam:AttachPolicy`
    - `iam:CreateRole`
    - `iam:CreatePolicy`
- AWS KMS
    - `kms:describeKey`

> **Note**
>
> Si el clúster de base de datos y la función de Lambda se encuentran en diferentes zonas de disponibilidad, su cuenta podría incurrir en costes cruzados de la zona de disponibilidad.

Cuando se configura una conexión entre una función de Lambda y un clúster de base de datos de Aurora, Amazon RDS configura el grupo de seguridad de la VPC para su función y su clúster de base de datos. Si usa RDS Proxy, Amazon RDS también configura el grupo de seguridad de la VPC para el proxy. Amazon RDS realiza una acción de acuerdo con la configuración actual de los grupos de seguridad asociados al clúster de base de datos, la función de Lambda y el proxy, tal como se describe en la siguiente tabla.

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincide con el patrón `rds-lambda-` *n* o si un proxy ya está conectado a su clúster de base de datos, RDS comprueba si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE` . No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen.	Hay uno o más grupos de seguridad asociados a la función de Lambda con un nombre que coincide con el patrón `lambda-rds-` *n* o `lambda-rdsproxy-` *n* (donde *n* es un número). No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad solo tiene una regla de salida bien con el grupo de seguridad de la VPC del clúster de base de datos o el proxy como destino.	Hay uno o más grupos de seguridad asociados al proxy con un nombre que coincide con el patrón `rdsproxy-lambda-` *n* (donde *n* es un número). No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene reglas de entrada y salida con los grupos de seguridad de la VPC de la función de Lambda y el clúster de base de datos.	Amazon RDS no realiza ninguna acción. Ya se configuró automáticamente una conexión entre la función de Lambda, el proxy (opcional) y el clúster de base de datos. Como ya existe una conexión entre la función, el proxy y la base de datos, los grupos de seguridad no se modifican.
Se aplica alguna de las siguientes condiciones:	Se aplica alguna de las siguientes condiciones:	Se aplica alguna de las siguientes condiciones:	RDS action: create new security groups

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
• No hay ningún grupo de seguridad asociado al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda-` *n* o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE`. • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda-` *n* o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE`. Sin embargo, ninguno de estos grupos de seguridad se puede usar para	• No hay ningún grupo de seguridad asociado a la función de Lambda con un nombre que coincida con el patrón `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. • Hay uno o más grupos de seguridad asociados a la función de Lambda con un nombre que coincide con el patrón `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos. Amazon RDS no puede utilizar un grupo de seguridad que no tenga una	• No hay ningún grupo de seguridad asociado al proxy con un nombre que coincida con el patrón `rdsproxy-lambda-` *n*. • Hay uno o más grupos de seguridad asociados al proxy con un nombre que coincide con `rdsproxy-lambda-` *n*. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos o la función de Lambda. Amazon RDS no puede utilizar un grupo de seguridad que no tenga reglas de entrada y salida con el grupo de seguridad de la VPC	

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
la conexión con la función de Lambda. Amazon RDS no puede usar un grupo de seguridad que no tengan una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificad o. Los ejemplos de modificaciones incluyen agregar una regla o cambiar el puerto de una regla existente.	regla de salida con el grupo de seguridad de la VPC del clúster de base de datos o el proxy como destino. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	del clúster de base de datos y la función de Lambda. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda- n` o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE` . No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen.	Hay uno o más grupos de seguridad asociados a la función de Lambda con un nombre que coincide con el patrón `lambda-rds- n` o `lambda-rdsproxy- n`. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos. Amazon RDS no puede utilizar un grupo de seguridad que no tenga una regla de salida con el grupo de seguridad de la VPC del clúster de base de datos o el proxy como destino. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	Hay uno o más grupos de seguridad asociados al proxy con un nombre que coincide con el patrón `rdsproxy-lambda- n`. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos o la función de Lambda. Amazon RDS no puede utilizar un grupo de seguridad que no tenga reglas de entrada y salida con el grupo de seguridad de la VPC del clúster de base de datos y la función de Lambda. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	RDS action: create new security groups

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda-`*n* o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE`. No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene solo una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen.	Existe un grupo de seguridad de Lambda válido para la conexión, pero no está asociado a la función de Lambda. Este grupo de seguridad tiene un nombre que coincide con el patrón `lambda-rds-`*n* o `lambda-rdsproxy-`*n*. No se ha modificado. Solo tiene una regla de salida con el grupo de seguridad de la VPC del clúster de base de datos o el proxy como destino.	Existe un grupo de seguridad del proxy válido para la conexión, pero no está asociado al proxy. Este grupo de seguridad tiene un nombre que coincide con el patrón `rdsproxy-lambda-`*n*. No se ha modificado. Tiene reglas de entrada y salida con el grupo de seguridad de la VPC del clúster de base de datos y la función de Lambda.	RDS action: associate Lambda security group

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda- n` o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE` . • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda- n` o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE` . Sin embargo,	Hay uno o más grupos de seguridad asociados a la función de Lambda con un nombre que coincide con el patrón `lambda-rds- n` o `lambda-rdsproxy- n`. No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad solo tiene una regla de salida con el grupo de seguridad de la VPC de la instancia de base de datos o el proxy como destino.	Hay uno o más grupos de seguridad asociados al proxy con un nombre que coincide con el patrón `rdsproxy-lambda- n`. No se ha modificado ningún grupo de seguridad que coincida con el patrón. Este grupo de seguridad tiene reglas de entrada y salida con el grupo de seguridad de la VPC del clúster de base de datos y la función de Lambda.	RDS action: create new security groups

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con la función de Lambda o el proxy. Amazon RDS no puede usar un grupo de seguridad que no tengan una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.			

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda- n` o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE` . • Hay uno o más grupos de seguridad asociados al clúster de base de datos con un nombre que coincida con el patrón `rds-lambda- n` o si el valor de `TargetHealth` de un proxy asociado es `AVAILABLE` . Sin embargo,	Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado a la función de Lambda con un nombre que coincida con el patrón `lambda-rds- n` o `lambda-rdsproxy- n`. • Hay uno o más grupos de seguridad asociados a la función de Lambda con un nombre que coincide con el patrón `lambda-rds- n` o `lambda-rdsproxy- n`. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos.	Se aplica alguna de las siguientes condiciones: • No hay ningún grupo de seguridad asociado al proxy con un nombre que coincida con el patrón `rdsproxy-lambda- n`. • Hay uno o más grupos de seguridad asociados al proxy con un nombre que coincide con `rdsproxy-lambda- n`. Sin embargo, Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con el clúster de base de datos o la función de Lambda. Amazon RDS no puede utilizar un grupo de seguridad	RDS action: create new security groups

Configuración del grupo de seguridad de RDS actual	Configuración actual del grupo de seguridad de Lambda	Configuración actual del grupo de seguridad del proxy	Acción de RDS
Amazon RDS no puede usar ninguno de estos grupos de seguridad para la conexión con la función de Lambda o el proxy. Amazon RDS no puede usar un grupo de seguridad que no tengan una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	Amazon RDS no puede utilizar un grupo de seguridad que no tenga una regla de salida con el grupo de seguridad de la VPC del clúster de base de datos o el proxy como origen. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	que no tenga reglas de entrada y salida con el grupo de seguridad de la VPC del clúster de base de datos y la función de Lambda. Amazon RDS tampoco puede usar un grupo de seguridad que se haya modificado.	

Acción de RDS de: crear nuevos grupos de seguridad

Amazon RDS realiza las siguientes acciones:

- Crea un nuevo grupo de seguridad que coincide con el patrón `rds-lambda-`*n* o `rds-rdsproxy-`*n* (si elige utilizar RDS Proxy). Este grupo de seguridad tiene una regla de entrada con el grupo de seguridad de la VPC de la función de Lambda o el proxy como origen. Este grupo de seguridad está asociado al clúster de base de datos y permite que la función o el proxy accedan al clúster de base de datos.

- Crea un nuevo grupo de seguridad que coincide con el patrón `lambda-rds-`*n* o `lambda-rdsproxy-`*n*. Este grupo de seguridad tiene una regla de salida bien con el grupo de seguridad de la VPC del clúster de base de datos o el proxy como destino. Este grupo de seguridad está asociado a la función de Lambda y permite que la función envíe tráfico al clúster de base de datos o que envíe tráfico a través de un proxy.
- Crea un nuevo grupo de seguridad que coincide con el patrón `rdsproxy-lambda-`*n*. Este grupo de seguridad tiene reglas de entrada y salida con el grupo de seguridad de la VPC del clúster de base de datos y la función de Lambda.

Acción de RDS : asociar un grupo de seguridad de Lambda

Amazon RDS asocia el grupo de seguridad de Lambda válido y existente a la función de Lambda. Este grupo de seguridad permite que la función envíe tráfico al clúster de base de datos o que envíe tráfico a través de un proxy.

Conexión automática de una función de Lambda y un clúster de base de datos de Aurora

Puede utilizar la consola de Amazon RDS para conectar automáticamente una función de Lambda a su clúster de base de datos. Esto simplifica el proceso de establecer una conexión entre estos recursos.

También puede usar RDS Proxy para incluir un proxy en la conexión. Las funciones de Lambda hacen frecuentes conexiones cortas a la base de datos que aprovechan el grupo de conexiones que ofrece RDS Proxy. Puede aprovechar cualquier autenticación de IAM que ya tenga para sus funciones de Lambda, en lugar de administrar las credenciales de la base de datos en el código de la aplicación de Lambda.

Puede conectar un clúster de base de datos existente a funciones de Lambda nuevas y existentes mediante la página de configuración de conexiones de Lambda. El proceso de configuración configura automáticamente los grupos de seguridad necesarios en su nombre.

Antes de configurar una conexión entre una función de Lambda y un clúster de base de datos, asegúrese de que:

- Su función de Lambda y el clúster de base de datos estén en la misma VPC.

- Tiene los permisos adecuados para su cuenta de usuario. Para obtener más información acerca de los requisitos, consulte Información general de la conectividad automática con una función de Lambda.

Si realiza cambios en los grupos de seguridad después de configurar la conectividad, los cambios podrían afectar a la conexión entre la función de Lambda y el clúster de base de datos.

> **Note**
>
> Puede configurar automáticamente una conexión entre el clúster de base de datos y una función de Lambda solo en la Consola de administración de AWS. Para conectar una función de Lambda, todas las instancias del clúster de base de datos deben estar en estado Disponible.

Para conectar automáticamente una función de Lambda y un clúster de base de datos

<result>

Tras confirmar la configuración, Amazon RDS inicia el proceso de conexión de su función de Lambda, RDS Proxy (si ha utilizado un proxy) y clúster de base de datos. La consola muestra el cuadro de diálogo Detalles de la conexión, que muestra los cambios del grupo de seguridad que permiten las conexiones entre los recursos.
</result>

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Bases de datos y, a continuación, seleccione el clúster de base de datos que desea conectar a una función de Lambda.
3. En Acciones, elija Configurar la conexión de Lambda.
4. En la página Configurar la conexión de Lambda, en Seleccionar la función de Lambda, realice una de las siguientes acciones:
   - Si ya tiene una función de Lambda en la misma VPC que su clúster de base de datos, elija Elegir una función existente y, a continuación, seleccione la función.
   - Si no tiene una función de Lambda en la misma VPC, elija Crear función nueva y, a continuación, introduzca un Nombre de la función. El tiempo de ejecución predeterminado

está establecido en Nodejs.18. Puede modificar la configuración de la nueva función de Lambda en la consola de Lambda después de completar la configuración de la conexión.

5. (Opcional) En RDS Proxy, seleccione Conexión mediante RDS Proxy y, a continuación, realice una de las siguientes acciones:
   - Si ya tiene un proxy que quiere usar, elija Elegir un proxy existente y, a continuación, elija el proxy.
   - Si no dispone de un proxy y desea que Amazon RDS lo cree automáticamente, elija Crear un proxy nuevo. A continuación, para Credenciales de la base de datos, realice una de las siguientes acciones:

     a. Elija Nombre de usuario y contraseña de la base de datos y, a continuación, introduzca el Nombre de usuario y la Contraseña para su clúster de base de datos.

     b. Elija Secreto de Secrets Manager. A continuación, para Seleccionar secreto, elija un secreto de AWS Secrets Manager. Si no tiene ningún secreto de Secrets Manager, elija Crear un nuevo secreto de Secrets Manager para crear un nuevo secreto. Después de crear el secreto, en Seleccionar secreto, elija el nuevo secreto.

     Después de crear el nuevo proxy, elija Elegir un proxy existente y, a continuación, elija el proxy. Tenga en cuenta que el proxy puede tardar algún tiempo en estar disponible para la conexión.
6. (Opcional) Amplíe Resumen de conexión y verifique las actualizaciones destacadas de sus recursos.
7. Elija Set up (Configurar).

Visualización de los recursos de computación conectados

Puede utilizar la Consola de administración de AWS para ver las funciones de Lambda que están conectadas a su clúster de base de datos. Los recursos que se muestran incluyen las conexiones de los recursos de computación que Amazon RDS configuró automáticamente.

Los recursos de computación de la lista no incluyen los que se conectan manualmente al clúster de base de datos. Por ejemplo, para permitir que un recurso de computación acceda manualmente a su clúster de base de datos puede añadir una regla al grupo de seguridad de la VPC asociado a la base de datos.

Para que la consola muestre una función de Lambda, se deben cumplir las siguientes condiciones:

- El nombre del grupo de seguridad asociado al recurso de computación coincide con el patrón `lambda-rds-`*n* o `lambda-rdsproxy-`*n* (donde *n* es un número).
- El grupo de seguridad asociado al recurso de computación tiene una regla de salida con el rango de puertos establecido en el puerto utilizado por el clúster de base de datos o un proxy asociado. El destino de la regla de salida debe establecerse en un grupo de seguridad asociado al clúster de base de datos o un proxy asociado.
- Si la configuración incluye un proxy, el nombre del grupo de seguridad adjunto al proxy asociado a la base de datos coincide con el patrón `rdsproxy-lambda-`*n* (donde *n* es un número).
- El grupo de seguridad asociado a la función tiene una regla de salida con el rango de puertos establecido en el puerto utilizado por el clúster de base de datos o un proxy asociado. El destino debe establecerse en un grupo de seguridad asociado al clúster de base de datos o un proxy asociado.

Para ver los recursos de computación conectados automáticamente a un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Bases de datos y, a continuación, elija el clúster de base de datos.
3. En la pestaña Conectividad y seguridad, consulte los recursos de computación en Recursos de computación conectados.

Modificación de un clúster de base de datos de Amazon Aurora

Puede cambiar la configuración de un clúster de base de datos para completar tareas como el cambio del periodo de retención de copia de seguridad o el puerto de la base de datos. También puede modificar instancias de base de datos en un clúster de base de datos para completar tareas como el cambio de la clase de dicha instancia de base de datos o la activación de información sobre rendimiento para ella. En este tema se detalla el proceso de modificación de un clúster de base de datos Aurora y sus instancias de base de datos y se describe la configuración para cada uno de ellos.

Recomendamos que pruebe cualquier cambio en un clúster o una instancia de prueba de una base de datos antes de modificar un clúster o una instancia de base de datos de producción para que pueda comprender completamente el impacto de cada cambio. Esto es especialmente importante al actualizar la versión de la base de datos.

Temas

- Modificación del clúster de base de datos con la consola, CLI y API
- Modificación de una instancia de base de datos en un clúster de base de datos
- Cambio de la contraseña del usuario maestro de la base de datos
- Configuración para Amazon Aurora
- Configuración que no se aplica a los clústeres de base de datos de Amazon Aurora
- Configuración que no se aplica a las instancias de base de datos de Amazon Aurora

Modificación del clúster de base de datos con la consola, CLI y API

Puede modificar un clúster de base de datos utilizando la Consola de administración de AWS, la AWS CLI o la API de RDS.

> **Note**
>
> La mayoría de las modificaciones se pueden aplicar de inmediato o en el siguiente periodo de mantenimiento programado. Algunas modificaciones, como activar la protección contra la eliminación se aplican inmediatamente, independientemente de cuándo decida aplicarlas. El cambio de la contraseña maestra en la Consola de administración de AWS siempre se aplica inmediatamente. Sin embargo, al utilizar la AWS CLI o la API de RDS, puede elegir si

desea aplicar este cambio inmediatamente o durante el siguiente período de mantenimiento programado.
Si utiliza puntos de conexión SSL y cambia el identificador del clúster de base de datos, detenga y reinicie el clúster de base de datos para actualizar los puntos de conexión SSL. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon Aurora.

Consola

Para modificar un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione el clúster de base de datos que desee modificar.
3. Elija Modify (Modificar). Aparece la página Modify DB clúster (Modificar clúster de base de datos).
4. Cambie los parámetros que desee. Para obtener más información acerca de cada configuración, consulte Configuración para Amazon Aurora.

   > **Note**
   >
   > En la Consola de administración de AWS, algunos cambios en el nivel de instancia solo se aplican a la instancia actual de la base de datos, mientras que otros se aplican a la totalidad del clúster de base de datos. Para obtener información sobre si una configuración se aplica a la instancia de base de datos o al clúster de la base de datos, consulte el ámbito de la configuración en Configuración para Amazon Aurora. Para cambiar una configuración que modifique todo el clúster de base de datos en el nivel de la instancia en la Consola de administración de AWS, siga las instrucciones de Modificación de una instancia de base de datos en un clúster de base de datos.

5. Cuando haya realizado todos los cambios que desee, elija Continue y compruebe el resumen de las modificaciones.
6. Para aplicar los cambios inmediatamente, seleccione Apply immediately.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify clúster (Modificar clúster) para guardarlos.

O bien, elija Back para editar los cambios o Cancel para cancelarlos.

AWS CLI

Para modificar un clúster de base de datos mediante la AWS CLI, llame al comando modify-db-clúster. Especifique el identificador de clúster de bases de datos y los valores de la configuración que desea modificar. Para obtener más información acerca de cada configuración, consulte Configuración para Amazon Aurora.

> **Note**
>
> Algunos ajustes se aplican únicamente a las instancias de base de datos. Para cambiar dichos ajustes, siga las instrucciones de Modificación de una instancia de base de datos en un clúster de base de datos.

Example

El siguiente comando modifica `mydbcluster` configurando el periodo de retención de copia de seguridad en 1 semana (7 días).

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 7
```

En:Windows

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 7
```

API de RDS

Para modificar un clúster de base de datos mediante la API de Amazon RDS, llame a la operación ModifyDBclúster. Especifique el identificador de clúster de bases de datos y los valores de la configuración que desea modificar. Para obtener información acerca de cada parámetro, consulte Configuración para Amazon Aurora.

> **Note**
>
> Algunos ajustes se aplican únicamente a las instancias de base de datos. Para cambiar dichos ajustes, siga las instrucciones de Modificación de una instancia de base de datos en un clúster de base de datos.

Modificación de una instancia de base de datos en un clúster de base de datos

Puede modificar una instancia de base de datos en un clúster de base de datos utilizando la Consola de administración de AWS, la AWS CLI o la API de RDS.

Al modificar una instancia de base de datos, puede aplicar los cambios inmediatamente. Para aplicar los cambios inmediatamente, seleccione la opción Apply Immediately (Aplicar inmediatamente) en la Consola de administración de AWS, utilice el parámetro `--apply-immediately` al llamar a la AWS CLI o establezca el parámetro `ApplyImmediately` en `true` cuando utilice la API de Amazon RDS.

Si no elige aplicar cambios inmediatamente, los cambios se aplazarán hasta la siguiente ventana de mantenimiento. Durante la siguiente ventana de mantenimiento, se aplica cualquiera de estos cambios diferidos. Si decide aplicar cambios inmediatamente, se aplicarán los cambios nuevos y los cambios previamente diferidos.

Para ver las modificaciones pendientes para la siguiente ventana de mantenimiento, utilice el comando describe-db-clústers de la AWS CLI y marque el campo `PendingModifiedValues`.

> **Important**
>
> Si alguna de las modificaciones diferidas requiere un tiempo de inactividad, al elegir Apply immediately (Aplicar inmediatamente) puede causar un tiempo de inactividad inesperado para la instancia de base de datos. No hay tiempo de inactividad para el resto de instancias de base de datos en el clúster de base de datos.
> Las modificaciones que se aplazan no aparecen en la salida del comando `describe-pending-maintenance-actions` CLI. Las acciones de mantenimiento solo incluyen las actualizaciones del sistema programadas para la siguiente ventana de mantenimiento.

Consola

Para modificar una instancia de base de datos en un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos que desea modificar.
3. Para Actions (Acciones), elija Modify (Modificar). Aparece la página Modify DB instance (Modificar instancia de base de datos).
4. Cambie los parámetros que desee. Para obtener más información acerca de cada ajuste, consulte Configuración para Amazon Aurora.

   > **Note**
   >
   > Algunos ajustes se aplican a todo el clúster de la base de datos y deben cambiarse a nivel del clúster. Para cambiar dichos ajustes, siga las instrucciones de Modificación del clúster de base de datos con la consola, CLI y API.
   > En la Consola de administración de AWS, algunos cambios en el nivel de instancia solo se aplican a la instancia actual de la base de datos, mientras que otros se aplican a la totalidad del clúster de base de datos. Para obtener información sobre si una configuración se aplica a la instancia de base de datos o al clúster de la base de datos, consulte el ámbito de la configuración en Configuración para Amazon Aurora.

5. Cuando haya realizado todos los cambios que desee, elija Continue y compruebe el resumen de las modificaciones.
6. Para aplicar los cambios inmediatamente, seleccione Apply immediately.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify DB instance (Modificar instancia de base de datos) para guardar los cambios.

   O bien, elija Back para editar los cambios o Cancel para cancelarlos.

AWS CLI

Para modificar una instancia de base de datos en un clúster de base de datos mediante la AWS CLI, llame al comando modify-db-instance. Especifique el identificador de instancias de bases de datos

y los valores de la configuración que desea modificar. Para obtener información acerca de cada parámetro, consulte Configuración para Amazon Aurora.

> **Note**
>
> Algunos ajustes se aplican al clúster de base de datos completo. Para cambiar dichos ajustes, siga las instrucciones de Modificación del clúster de base de datos con la consola, CLI y API.

Example

El siguiente código modifica `mydbinstance` al establecer la clase de instancia de base de dato en `db.r4.xlarge`. Los cambios se aplican durante el siguiente periodo de mantenimiento si se utiliza el parámetro `--no-apply-immediately`. Utilice `--apply-immediately` para aplicar los cambios inmediatamente.

Para Linux, macOS o:Unix

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r4.xlarge \
    --no-apply-immediately
```

En:Windows

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.r4.xlarge ^
    --no-apply-immediately
```

API de RDS

Para modificar una instancia de base de datos mediante la API de Amazon RDS, llame a la operación ModifyDBInstance. Especifique el identificador de instancias de bases de datos y los valores de la configuración que desea modificar. Para obtener información acerca de cada parámetro, consulte Configuración para Amazon Aurora.

> **Note**
>
> Algunos ajustes se aplican al clúster de base de datos completo. Para cambiar dichos ajustes, siga las instrucciones de Modificación del clúster de base de datos con la consola, CLI y API.

Cambio de la contraseña del usuario maestro de la base de datos

Puede usar la Consola de administración de AWS o la AWS CLI para cambiar la contraseña del usuario maestro.

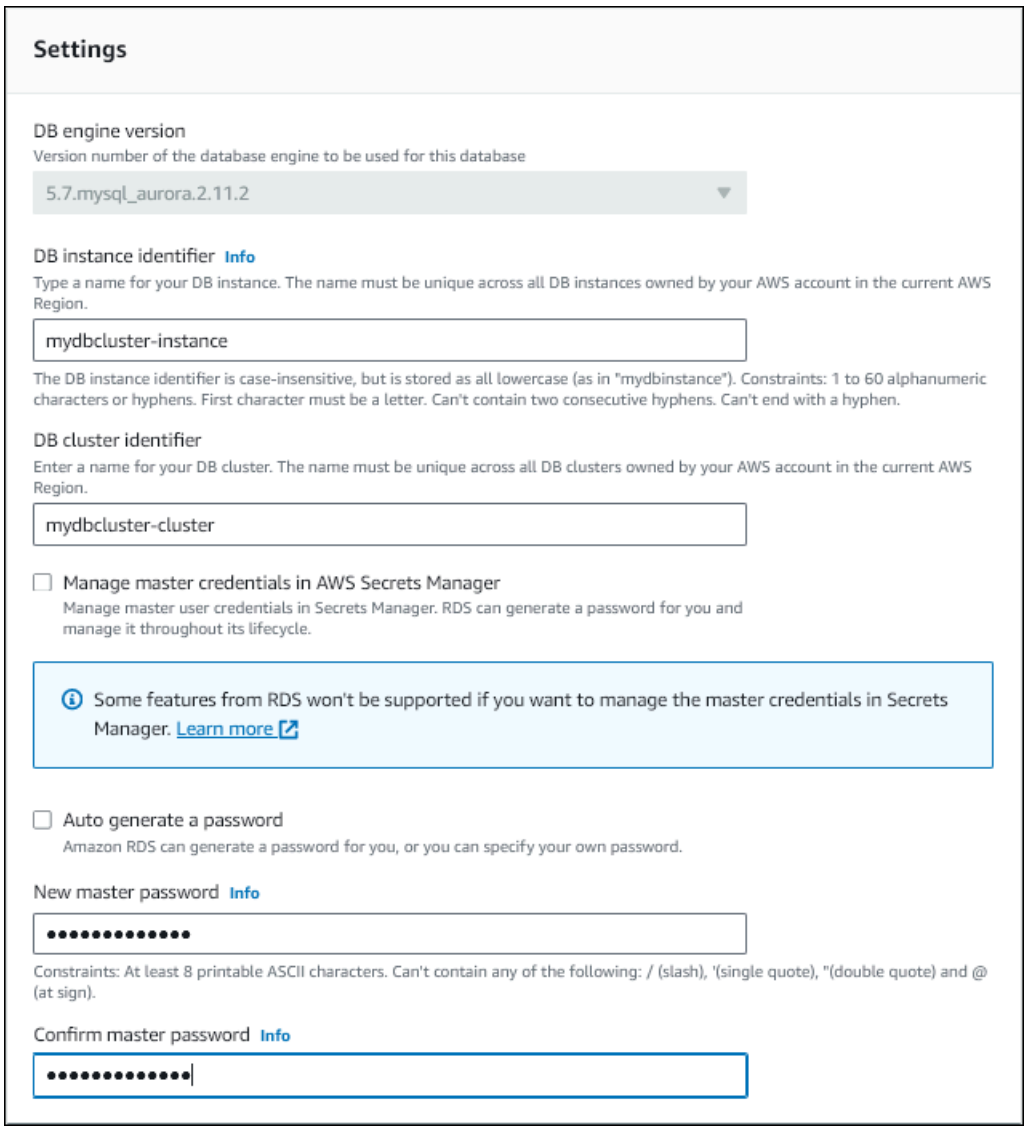
Consola

Para modificar la instancia de base de datos escritora para cambiar la contraseña del usuario maestro, utilice la Consola de administración de AWS.

Para cambiar la contraseña del usuario maestro

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos que desea modificar.
3. Para Actions (Acciones), elija Modify (Modificar).

   Aparece la página Modificar instancia de base de datos.
4. Introduzca una Nueva contraseña maestra.
5. En Confirmar contraseña maestra, introduzca la misma contraseña nueva.



6. Elija Continue (Continuar) y consulte el resumen de las modificaciones.

> **Note**
>
> Los cambios de contraseña siempre se aplican inmediatamente.

7. En la página de confirmación, elija Modify DB instance (Modificar instancia de base de datos).

CLI

Para cambiar la contraseña del usuario maestro con la AWS CLI, llame al comando modify-db-clúster. Especifique el identificador del clúster de base de datos y la nueva contraseña, tal y como se muestra en los siguientes ejemplos.

No es necesario que especifique `--apply-immediately|--no-apply-immediately`, ya que los cambios de contraseña siempre se aplican de forma inmediata.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --master-user-password mynewpassword
```

En:Windows

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --master-user-password mynewpassword
```

Configuración para Amazon Aurora

La siguiente tabla contiene detalles sobre la configuración que se puede modificar, los métodos para modificar la configuración y el ámbito de la configuración. El ámbito determina si la configuración se aplica al clúster de base de datos completo o si puede establecerse solamente para instancias de base de datos específicas.

> **Note**
>
> Hay opciones de configuración adicionales disponibles si se modifica un clúster de base de datos de Aurora Serverless v1 o Aurora Serverless v2. Para obtener más información sobre estas opciones, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1 y Administración de clústeres de bases de datos de Aurora Serverless v2.
> Además, algunas opciones de configuración no están disponibles para Aurora Serverless v1 y Aurora Serverless v2 debido a sus limitaciones. Para obtener más información, consulte Limitaciones de Aurora Serverless v1 y Requisitos y limitaciones para Aurora Serverless v2.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Auto minor version upgrade (Actualización automática de versiones secundarias) Si desea permitir que la instancia de base de datos reciba automáticamente las actualizaciones preferidas de la versión secundaria del motor de base de datos cuando estén disponibles. Las actualizaciones se instalan únicamente durante la ventana de mantenimiento programada. Para obtener más información acerca de las actualizaciones de motor de Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.	**Note** Esta configuración está activada de forma predeterminada. Para cada nuevo clúster, elija el valor adecuado para esta configuración en función de su importancia, duración prevista y la cantidad de pruebas de verificación que realice después de cada actualización. Cuando cambie esta configuración, realice esta modificación para cada instancia de base de datos del clúster de Aurora.	El clúster de base de datos completo	No se produce una interrupción durante este cambio. Las interrupciones se producen durante futuras ventanas de mantenimiento cuando Aurora aplica actualizaciones automáticas.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Para obtener más información acerca de la configuración de Auto minor version upgrade (Actualización automática de la versión secundaria) para Aurora MySQL, consulte Activación de actualizaciones automáticas entre versiones secundarias de Aurora MySQL.	Si alguna instancia de base de datos del clúster tiene esta configuración desactivada, el clúster no se actualiza automáticamente. Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute modify-db-instance y establezca la opción `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` . Con la API de RDS, llame a ModifyDBInstance y configure el parámetro `AutoMinorVersionUpgrade` .		

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Backup retention period (Periodo de retención de copia de seguridad) El número de días que se conservan las copias de seguridad automáticas. El valor mínimo es 1. Para obtener más información, consulte Copias de seguridad.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute modify-db-clúster y configure la opción `--backup-retention-period`. Con la API de RDS, llame a ModifyDBclúster y configure el parámetro `BackupRetentionPeriod` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Ventana Copia de seguridad (hora de inicio) El intervalo de tiempo durante el que se realizan las copias de seguridad automáticas de las bases de datos. Backup Window se expresa mediante una hora de inicio en tiempo universal coordinado (UTC) y una duración en horas. Las copias de seguridad Aurora son continuas e incrementales, pero la ventana de copia de seguridad se utiliza para crear una copia de seguridad diaria del sistema que se conserva dentro del período de retención de la copia de seguridad. Puede copiarlo para conservarlo fuera del período de retención.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute modify-db-clúster y configure la opción `--preferred-backup-window`. Con la API de RDS, llame a ModifyDBclúster y configure el parámetro `PreferredBackupWindow` .	El clúster de base de datos completo.	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
La ventana de mantenimiento y la ventana de copia de seguridad de la instancia de base de datos no se pueden solapar. Para obtener más información, consulte Intervalo de copia de seguridad.			
Configuración de la capacidad Son las propiedades de escalado de un clúster de base de datos de Aurora Serverless v1. Solo es posible modificar las propiedades de escalado para clústeres de base de datos en el modo del motor de base de datos `serverless` . Para obtener más información sobre Aurora Serverless v1, consulte Uso de Amazon Aurora Serverless v1.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute modify-db-clúster y configure la opción `--scaling-configuration` . Con la API de RDS, llame a ModifyDBc lúster y configure el parámetro `ScalingCo nfiguration` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio. El cambio se produce inmediatamente. Este ajuste omite la configuración de aplicación inmediata.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Certificate authority (Autoridad de certificado) Entidad de certificación (CA) del certificado de servidor que utiliza la instancia de base de datos.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute modify-db-instance y establezca la opción `--ca-certificate-identifier` . Con la API de RDS, llame a ModifyDBInstance y configure el parámetro `CACertificateIdentifier` .	Solo la instancia de base de datos especificada	Solo se produce una interrupción si el motor de base de datos no admite la rotación sin reinicio. Puede utilizar el comando de la AWS CLI describe-db-engine-versions para determinar si el motor de base de datos admite la rotación sin reinicio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Configuración de almacenamiento en clústeres El tipo de almacenamiento para el clúster de base de datos: Aurora I/O-Optimized o Aurora Standard. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute modify-db-clúster y configure la opción `--storage-type`. Con la API de RDS, llame a ModifyDBclúster y configure el parámetro `StorageType` .	El clúster de base de datos completo	Si se cambia el tipo de almacenamiento de un clúster de base de datos de Aurora PostgreSQL con clases de instancia de lecturas optimizadas, se produce una interrupción. Esto no ocurre cuando se cambian los tipos de almacenamiento de los clústeres con otros tipos de clases de instancias. Para obtener más información sobre los tipos de clase de instancia de base de datos, consulte Tipos de clase de instancia de base de datos.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Copy Tags To Snapshots (Copiar etiquetas en instantáneas) Selecciónelo para especificar qué etiquetas definidas para este clúster de base de datos se copian en las instantáneas de base de datos creadas desde este clúster de base de datos. Para obtener más información, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute https://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-cluster.html y establezca la opción `--copy-tags-to-snapshot` o `--no-copy-tags-to-snapshot`. Mediante la API de RDS, realice una llamada a https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_ModifyDBCluster.html y establezca el parámetro `CopyTagsToSnapshot`.	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Data API (API de datos) Puede acceder a Aurora Serverless v1 con aplicaciones web basadas en servicios, incluidas AWS Lambda y AWS AppSync. Esta configuración solo se aplica a un clúster de base de datos de Aurora Serverless v1. Para obtener más información, consulte Uso de la API de datos de Amazon RDS.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--enable-http-endpoint` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `EnableHttpEndpoint` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Database authentication (Autenticación de bases de datos) La autenticación de bases de datos que desea usar. Para MySQL: • Elija Password authentication (Autenticación de contraseña) para autenticar solo a los usuarios de la base de datos con contraseñas de base de datos. • Elija Password and IAM database authentication (Contraseña y autenticación de base de datos de IAM) para autenticar a los usuarios de la base de datos con contraseñas y credenciales de usuario a través de usuarios y roles de IAM. Para obtener más información,	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con AWS CLI, ejecute modify-db-clúster y establezca las siguientes opciones: • Para la autenticación IAM, establezca la opción `--enable-iam-database-authentication|--no-enable-iam-database-authentication` . • Para la autenticación Kerberos, establezca las opciones `--domain` y `--domain-iam-role-name` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
consulte Autenticación de bases de datos de IAM . Para PostgreSQL: • Elija IAM database authentication (Autenticación de base de datos de IAM) para autenticar a los usuarios de la base de datos con contraseñas y credenciales de la base de datos a través de usuarios y roles. Para obtener más información, consulte Autenticación de bases de datos de IAM . • Elija Kerberos authentication (Autenticación Kerberos) para autenticar contraseñas de base de datos y credenciales de usuario mediante la autenticación Kerberos. Para obtener más	Con la API de RDS, llame a ModifyDBclúster y establezca los siguientes parámetros: • Para la autenticación IAM, establezca el parámetro `EnableIAMDatabaseAuthentication` . • Para la autenticación Kerberos, establezca los parámetros `Domain` y `DomainIAMRoleName` .		

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
información, consulte Uso de la autenticación Kerberos con Aurora PostgreSQL.			
Database port (Puerto de base de datos) El puerto que desea utilizar para obtener acceso al clúster de base de datos.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--port`. Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `Port`.	El clúster de base de datos completo	Se produce una interrupción durante este cambio. Todas las instancias de base de datos en el clúster de base de datos se reinician inmediata mente.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
DB clúster identifier (Identificador de clúster de base de datos) Identificador de clúster de base de datos. Este valor se almacena como una cadena en minúsculas. Al cambiar el identificador del clúster de base de datos, cambian los puntos de conexión del clúster de base de datos. Los puntos de conexión de las instancias de base de datos del clúster de base de datos no cambian.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--new-db-cluster-identifier`. Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `NewDBClusterIdentifier`.	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Grupo de parámetros de clúster de base de datos El grupo de parámetros de clúster de base de datos que desea asociar al clúster de base de datos. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--db-cluster-parameter-group-name` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `DBClusterParameterGroupName` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio. Cuando cambia el grupo de parámetros, los cambios en algunos parámetros se aplican a las instancias de bases de datos en el clúster de base de datos inmediatamente, sin reinicio. Los cambios en otros parámetros se aplican únicamente después de reiniciar las instancias de bases de datos en el clúster de base de datos.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
DB instance class (Clase de instancia de base de datos) Clase de instancia de base de datos que desea utilizar. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--db-instance-class` . Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca el parámetro `DBInstanceClass` .	Solo la instancia de base de datos especificada	Se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
DB Instance Identifier (Identificador de instancias de bases de datos) El identificador de instancias de base de datos. Este valor se almacena como una cadena en minúsculas.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--new-db-instance-identifier`. Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca el parámetro `NewDBInstanceIdentifier`.	Solo la instancia de base de datos especificada	Se produce un tiempo de inactividad durante este cambio. RDS reinicia la instancia de base de datos para actualizar lo siguiente: • Aurora MySQL: columna `SERVER_ID` de la tabla `information_schema.replica_host_status` • Aurora PostgreSQL: columna `server_id` de la función `aurora_replica_status()`

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
DB Parameter Group (Grupo de parámetros de base de datos) El grupo de parámetros de la base de datos que desea asociar a la instancia de base de datos. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--db-parameter-group-name` . Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca el parámetro `DBParameterGroupName` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio. Al asociar un nuevo grupo de parámetros de base de datos con una instancia de base de datos, los parámetros estáticos y dinámicos modificados se aplican solo después de reiniciar la instancia de base de datos. Sin embargo, si modifica los parámetros dinámicos en el grupo de parámetros de base de datos después de asociarlos a la instancia de base de datos, dichos cambios se aplican inmediatamente sin reiniciar. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora y Reinicio de un clúster de base de datos de Amazon

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
			Aurora o de una instancia de base de datos de Amazon Aurora.
Deletion protection (Protección contra eliminación) Seleccione Enable deletion protection (Habilitar la protección contra la eliminación) para evitar que se elimine el clúster de base de datos. Para obtener más información, consulte Protección contra eliminación para clústeres de Aurora.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--deletion-protection|--no-deletion-protection` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `DeletionProtection` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Engine version (Versión del motor) La versión del motor de base de datos que desea utilizar. Antes de actualizar el clúster de base de datos de producción, recomendamos que pruebe el proceso de actualización en un clúster de base de datos prueba para comprobar la duración y validar las aplicaciones.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--engine-version` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `EngineVersion` .	El clúster de base de datos completo	Se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Enhanced monitoring (Supervisión mejorada) Enable enhanced monitoring (Habilitat supervisión mejorada) para habilitar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta la instancia de base de datos. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca las opciones `--monitoring-role-arn` y `--monitoring-interval` . Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca los parámetros `MonitoringRoleArn` y `MonitoringInterval` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Log exports (Exportaciones de registros) Seleccione los tipos de registro que desee publicar en registros de Amazon Cloudwatch Para obtener más información, consulte Archivos de registro de base de datos de Aurora MySQL.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--cloudwatch-logs-export-configuration` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `CloudwatchLogsExportConfiguration` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Maintenance window (Periodo de mantenimiento) El intervalo de tiempo durante el que se produce el mantenimiento del sistema. El mantenimiento del sistema incluye actualizaciones, si procede. El periodo de mantenimiento se expresa mediante una hora de inicio en tiempo universal coordinado (UTC) y una duración en horas. Si establece un intervalo que incluya la hora actual, debe haber al menos 30 minutos entre la hora actual y el final del intervalo, para asegurarse de que se apliquen los cambios pendientes. Puede establecer el periodo de mantenimiento de manera	Para cambiar el periodo de mantenimiento del clúster de la base de datos con la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Para cambiar el periodo de mantenimiento de la instancia de base de datos con la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Para cambiar el periodo de mantenimiento del clúster de base de datos con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--preferred-maintenance-window` .	El clúster de base de datos completo o una instancia de base de datos individual	Si hay una o varias acciones pendientes que provocan una interrupción y el periodo de mantenimiento se cambia para incluir la hora actual, las acciones pendientes se aplican inmediatamente, y se produce una interrupción.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
independiente para el clúster de base de datos y para cada instancia de base de datos en el clúster de base de datos. Cuando el ámbito de una modificación es el clúster de base de datos completo, la modificación se realiza durante el periodo de mantenimiento del clúster de base de datos. Cuando el ámbito de una modificación es una instancia de base de datos, modificación se realiza durante el periodo de mantenimiento de esa instancia de base de datos. La ventana de mantenimiento y la ventana de copia de seguridad de la instancia de base de datos no se pueden solapar. Para obtener más información,	Para cambiar el periodo de mantenimiento de una instancia de base de datos con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--preferred-maintenance-window` . Para cambiar el periodo de mantenimiento del clúster de base de datos con la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `PreferredMaintenanceWindow` . Para cambiar el periodo de mantenimiento de una instancia de base de datos con la API de RDS, realice una llamada a `ModifyDBInstance` y establezca		

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
consulte Ventana de mantenimiento de Amazon RDS.	el parámetro `PreferredMaintenanceWindow`.		

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Administrar las credenciales maestras en AWS Secrets Manager Seleccione Manage master credentials in AWS Secrets Manager (Administrar credenciales maestras en AWS Secrets Manager) para administrar la contraseña del usuario maestro en un secreto en Secrets Manager. De forma opcional, elija la clave KMS para proteger el secreto. Elija entre las claves de KMS de su cuenta o bien introduzca la clave de otra cuenta. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-cluster` y establezca las opciones `--manage-master-user-password | --no-manage-master-user-password` y `--master-user-secret-kms-key-id` . Para cambiar la contraseña del usuario maestro inmediatamente, defina la opción `--rotate-master-user-password` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca los parámetros `ManageMas`	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Si Aurora ya administra la contraseña de usuario maestra del clúster de base de datos, puede rotar la contraseña del usuario maestro seleccionando Rotate secret immediately (Rotar el secreto inmediatamente). Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.	`terUserPassword` y `MasterUserSecretKmsKeyId` . Para cambiar la contraseña del usuario maestro inmediatamente, defina el parámetro `RotateMasterUserPassword` en `true`.		

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Tipo de red Protocolos de direccionamiento IP admitidos por el clúster de base de datos. IPv4 para especificar que los recursos se pueden comunicar con el clúster de base de datos solo a través del protocolo de direcciones IPv4. Modo de pila dual para especificar que los recursos se pueden comunicar con el clúster de base de datos mediante IPv4, IPv6 o ambos. Utilice el modo de pila doble si tiene recursos que deben comunicarse con su clúster de base de datos a través del protocolo de direccionamiento IPv6. Para usar el modo de doble pila, asegúrese de que haya al menos	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--network-type` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `NetworkType` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
dos subredes que abarquen dos zonas de disponibilidad que admitan los protocolos de red IPv4 e IPv6. Además, asegúrese de asociar un bloque CIDR IPv6 a todas las subredes del grupo de subredes de base de datos que especifique. Para obtener más información, consulte Direccionamiento IP de Amazon Aurora.			

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
New master password (Nueva contraseña maestra) La contraseña para el usuario maestro. • Para Aurora MySQL, la contraseña debe tener entre 8 y 41 caracteres ASCII imprimibles. • Para Aurora PostgreSQL, debe tener entre 8 y 99 caracteres ASCII imprimibles. • No puede contener /, ", @ o un espacio.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--master-user-password` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `MasterUserPassword` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Performance Insights Si se habilita Performance Insights, una herramienta que monitoriza la carga de las instancias de base de datos para que pueda y soluciona r los problemas de rendimiento de la base de datos. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance _` y establezca la opción `--enable-performance-insights|--no-enable-performance-insights` . Mediante la API de RDS, realice una llamada a `ModifyDBInstance _` y establezca el parámetro `EnablePerformanceInsights` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Performance InsightsAWS KMS key (Información sobre rendimiento) El identificador de AWS KMS key para el cifrado de datos de Performance Insights. El identificador de la clave de KMS es el nombre de recurso de Amazon (ARN), el identificador de la clave o el alias de clave de la clave de KMS. Para obtener más información, consulte Activación y desactivación de Información de rendimiento de Aurora.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--performance-insights-kms-key-id` . Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca el parámetro `PerformanceInsightsKMSKeyId` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Performance Insights retention period (Periodo de retención de información sobre rendimiento) El tiempo, en días, durante los que se conservan los datos de información sobre rendimiento. La configuración de retención en la capa gratuita es Default (7 days) (Predeter minado [7 días]). Para retener los datos de rendimiento durante más tiempo, especifiq ue de 1 a 24 meses. Para obtener más información acerca de los periodos de retención, consulte Precios y retención de datos de Performance Insights. Para obtener más información, consulte Activación y desactiva ción de Información de rendimiento de Aurora.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--performance-insights-rete ntion-period` . Mediante la API de RDS, realice una llamada a `ModifyDBI nstance` y establezca el parámetro `Performan ceInsight sRetentio nPeriod` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Promotion tier (Capa de promoción) Un valor que especifica el orden en que se promueven las réplicas de Aurora a la instancia principal en un clúster de base de datos tras un error de la instancia principal existente. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--promotion-tier`. Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca el parámetro `PromotionTier` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Public access (Acceso público) Publicly accesible (Accesible públicamente) para proporcionar una dirección IP pública a la instancia de base de datos, lo que significa que es accesible desde fuera de la VPC. Para que sea accesible públicamente, la instancia de base de datos también debe estar en una subred pública de la VPC. Not publicly accesible (No es accesible públicamente) para que la instancia de base de datos sea accesible solo desde dentro de la VPC. Para obtener más información, consulte Cómo ocultar un clúster de base de datos en una VPC desde Internet..	Uso de la Consola de administración de AWS, Modificación de una instancia de base de datos en un clúster de base de datos. Con la AWS CLI, ejecute `modify-db-instance` y establezca la opción `--publicly-accessible|--no-publicly-accessible` . Mediante la API de RDS, realice una llamada a `ModifyDBInstance` y establezca el parámetro `PubliclyAccessible` .	Solo la instancia de base de datos especificada	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Para conectarse a una instancia de base de datos desde fuera de su Amazon VPC, la instancia de base de datos debe ser accesible públicamente, el acceso debe concederse mediante las reglas entrantes del grupo de seguridad de la instancia de base de datos y deben cumplirse otros requisitos. Para obtener más información, consulte No puede conectarse a la instancia de base de datos de Amazon RDS. Si su instancia de base de datos no es accesible públicamente, también puede usar una conexión Site-to-site VPN AWS o una conexión a Direct Connect para acceder a ella desde una red privada. Para obtener más informaci			

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
ón, consulte Privacidad del tráfico entre redes.			
Configuración de la capacidad de Serverless v2 Es la capacidad de la base de datos de un clúster de base de datos de Aurora Serverless v2 medida en unidades de capacidad de Aurora (ACU). Para obtener más información, consulte Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--serverless-v2-scaling-configuration` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `ServerlessV2ScalingConfiguration` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio. El cambio se produce inmediatamente. Este ajuste omite la configuración de aplicación inmediata.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Security group (Grupo de seguridad) El grupo de seguridad que desea asociar al clúster de base de datos. Para obtener más información, consulte Control de acceso con grupos de seguridad.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--vpc-security-group-ids` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `VpcSecurityGroupIds` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración y descripción	Método	Ámbito	Notas acerca del tiempo de inactividad
Target Backtrack window (Periodo de Backtrack de destino) La cantidad de tiempo que desea poder realizar búsqueda de datos anteriores en su clúster de base de datos, en segundos. Esta configuración está disponible solo para Aurora MySQL y solo si el clúster de base de datos se creó con Backtrack habilitado.	Uso de la Consola de administración de AWS, Modificación del clúster de base de datos con la consola, CLI y API. Con la AWS CLI, ejecute `modify-db-cluster` y establezca la opción `--backtrack-window` . Mediante la API de RDS, realice una llamada a `ModifyDBCluster` y establezca el parámetro `BacktrackWindow` .	El clúster de base de datos completo	No se produce una interrupción durante este cambio.

Configuración que no se aplica a los clústeres de base de datos de Amazon Aurora

Las siguientes configuraciones en el comando de la AWS CLI `modify-db-cluster` y la operación `ModifyDBCluster` de la API de RDS no se aplican a los clústeres de base de datos de Amazon Aurora.

> **Note**
>
> No puede usar la Consola de administración de AWS para modificar esta configuración de los clústeres de base de datos de Aurora.

Configuración de la AWS CLI	Configuración de la API de RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`

Configuración que no se aplica a las instancias de base de datos de Amazon Aurora

Las siguientes configuraciones en el comando de la AWS CLI `modify-db-instance` y la operación `ModifyDBInstance` de la API de RDS no se aplican a las instancias de base de datos de Amazon Aurora.

> **Note**
>
> No puede usar la Consola de administración de AWS para modificar esta configuración para las instancias de base de datos de Aurora.

Configuración de la AWS CLI	Configuración de la API de RDS
`--allocated-storage`	`AllocatedStorage`
`--allow-major-version-upgrade|--no-allow-major-version-upgrade`	`AllowMajorVersionUpgrade`
`--copy-tags-to-snapshot|--no-copy-tags-to-snapshot`	`CopyTagsToSnapshot`
`--domain`	`Domain`
`--db-security-groups`	`DBSecurityGroups`
`--db-subnet-group-name`	`DBSubnetGroupName`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--multi-az|--no-multi-az`	`MultiAZ`
`--iops`	`Iops`
`--license-model`	`LicenseModel`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--processor-features`	`ProcessorFeatures`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--use-default-processor-features|--no-use-default-processor-features`	`UseDefaultProcessorFeatures`

Adición de réplicas de Aurora a un clúster de base de datos

Un clúster de base de datos de Aurora con replicación tiene una instancia de base de datos principal y hasta 15 réplicas de Aurora. Instancia de base de datos principal que admite operaciones de lectura y escritura y realiza todas las modificaciones de los datos en el volumen de clúster. Las réplicas de Aurora se conectan con el mismo volumen de almacenamiento que la instancia de base de datos principal y solo admiten operaciones de lectura. Utilice las réplicas de Aurora para descargar las cargas de trabajo de lectura desde la instancia de base de datos principal. Para obtener más información, consulte Réplicas de Aurora.

Las réplicas Amazon Aurora tienen las siguientes limitaciones:

- No se puede crear una réplica de Aurora para un clúster de base de datos Aurora Serverless v1. Aurora Serverless v1 tiene una única instancia de base de datos que se escala hacia arriba y hacia abajo automáticamente para admitir todas las operaciones de lectura y escritura de las bases de datos.

  No obstante, puede añadir instancias de lector a clústeres de base de datos de Aurora Serverless v2 Para obtener más información, consulte Adición de un lector Aurora Serverless v2.

Es recomendable que distribuya la instancia principal y las réplicas de Aurora de su Aurora clúster de base de datos entre varias zonas de disponibilidad para mejorar la disponibilidad del clúster de base de datos. Para obtener más información, consulte Disponibilidad por región.

Para eliminar una réplica de Aurora de un clúster de base de datos Aurora, elimine la réplica de Aurora siguiendo las instrucciones de Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora.

> **Note**
>
> Amazon Aurora también admite la replicación con una base de datos de externa o una instancia de base de datos de RDS La instancia de base de datos de RDS debe estar en la misma región de AWS que Amazon Aurora. Para obtener más información, consulte Replicación con Amazon Aurora.

Puede agregar réplicas de Aurora a un clúster de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para agregar una réplica de Aurora a un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione el clúster de base de datos en el que desee añadir a la nueva instancia de base de datos.
3. Asegúrese de que tanto el clúster como la instancia principal estén en el estado Disponible. Si el clúster de base de datos o la instancia principal están en un estado de transición como Creando, no puede agregar una réplica.

   Si el clúster no tiene una instancia primaria, cree una mediante el comando create-db-instance de la AWS CLI. Esta situación puede surgir si utilizó la CLI para restaurar una instantánea del clúster de base de datos y, a continuación, ve el clúster en la Consola de administración de AWS.
4. En Actions (Acciones), elija Add reader (Añadir lector).

   Aparecerá la página Add reader (Añadir lector).
5. En la página Add reader (Añadir lector), especifique las opciones para su réplica de Aurora. La siguiente tabla muestra la configuración de una réplica de Aurora.

Para esta opción	Haga lo siguiente
Availability zone (Zona de disponibilidad	Determine si desea especificar una zona de disponibilidad concreta. La lista incluye solo las zonas de disponibilidad asignadas al grupo de subredes de base de datos que eligió al crear el clúster de base de datos. Para obtener más información acerca de las zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.
Accesible públicamente	Seleccione `Yes` para asignar a la réplica de Aurora una dirección IP pública; de lo contrario, seleccione No. Para obtener más información acerca del modo de ocultar réplicas de Aurora del acceso público, consulte

Para esta opción	Haga lo siguiente
	Cómo ocultar un clúster de base de datos en una VPC desde Internet..
Encryption (Cifrado	Seleccione `Enable encryption` para habilitar el cifrado en reposo para esta réplica de Aurora. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
DB instance class (Clase de instancia de base de datos	Seleccione una clase de instancia de base de datos que defina los requisitos de procesamiento y memoria de la réplica de Aurora. Para obtener más informaci ón acerca de las opciones de clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora.
Aurora replica source (Origen de la réplica de Aurora	Seleccione el identificador de la instancia principal para la que se debe crear una réplica de Aurora.
Identificador de la instancia de la base de datos	Ingrese un nombre para la instancia que sea único para su cuenta en la región de AWS que ha seleccion ado. Puede optar por agregar información al nombre, como la región de AWS y el motor de base de datos que ha seleccionado, por ejemplo,**`aurora-read-instance1`** .
Priority (Prioridad	Elija una prioridad de conmutación por error para la instancia. Si no selecciona un valor, el ajuste predeterminado es tier-1. Esta prioridad determina el orden en que se promueven las réplicas de Aurora cuando el sistema se recupera de un error en la instancia principal. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.
Database port (Puerto de base de datos	El puerto de una réplica de Aurora coincide con el puerto del clúster de base de datos.

Para esta opción	Haga lo siguiente
DB Parameter Group (Grupo de parámetros de base de datos	Seleccione un grupo de parámetros. Aurora tiene un grupo de parámetros predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.
Performance Insights	La casilla de verificación Turn on Performance Insights (Activar Performance Insights) está activada de forma predeterminada. El valor no se hereda de la instancia de escritor. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.
Enhanced monitoring (Monitorización mejorada	Elija Enable enhanced monitoring (Habilitar monitorización mejorada) a fin de habilitar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta su clúster de base de datos. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
Monitoring Role (Rol de supervisión	Solo está disponible si Enhanced Monitoring (Monitorización mejorada) se establece en Enable enhanced monitoring (Habilitar monitorización mejorada). Elija la función de IAM que ha creado para permitir que Amazon RDS se comunique con registros de Amazon Cloudwatch, o bien elija Default (Predeterminado) para que RDS cree un rol denominado `rds-monitoring-role` . Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.

Para esta opción	Haga lo siguiente
Granularity (Grado de detalle	Solo está disponible si Enhanced Monitoring (Monitorización mejorada) se establece en Enable enhanced monitoring (Habilitar monitorización mejorada). Defina el intervalo, en segundos, en el que se recopilan las métricas para el clúster de base de datos.
Auto minor version upgrade (Actualización automática de versiones secundarias	Seleccione Enable auto minor version upgrade (Habilitar actualización automática a versiones secundarias) si desea habilitar su clúster de base de datos de Aurora para recibir actualizaciones de las versiones secundarias del motor de base de datos automáticamente cuando estén disponibles. El ajuste Auto minor version upgrade (Actualización automática a versiones secundarias) solo se aplica a los clústeres de base de datos de Aurora PostgreSQL y Aurora MySQL. Para los clústeres de Aurora MySQL 2.x, esta configuración actualiza los clústeres a una versión máxima de 2.07.2. Para obtener más información acerca de las actualizaciones de motor de Aurora PostgreSQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL. Para obtener más información acerca de las actualizaciones de motor de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

6. Elija Add reader (Añadir lector) para crear la réplica de Aurora.

AWS CLI

Para crear una réplica de Aurora en un clúster de base de datos, ejecute el comando create-db-instance de la AWS CLI. Incluya el nombre del clúster de base de datos como opción de `--db-`

`cluster-identifier`. Si lo desea, puede especificar una zona de disponibilidad para la réplica de Aurora usando el parámetro `--availability-zone`, como se muestra en los siguientes ejemplos.

Por ejemplo, el comando siguiente crea una nueva réplica de Aurora compatible con MySQL 5.7 llamada `sample-instance-us-west-2a`.

Para Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

El comando siguiente crea una nueva réplica de Aurora compatible con MySQL 5.7 llamada `sample-instance-us-west-2a`.

Para Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

El comando siguiente crea una nueva réplica de Aurora compatible con PostgreSQL llamada `sample-instance-us-west-2a`.

Para Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large \
    --availability-zone us-west-2a
```

Para Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large ^
    --availability-zone us-west-2a
```

API de RDS

Para crear una réplica de Aurora en su clúster de base de datos, realice una llamada a la operación CreateDBInstance. Incluya el nombre del clúster de base de datos como parámetro `DBClusterIdentifier`. Opcionalmente, puede especificar una zona de disponibilidad para la réplica de Aurora con el parámetro `AvailabilityZone`.

Para obtener información sobre Amazon EC2 Auto Scaling Amazon Aurora con réplicas de Aurora, consulte las siguientes secciones.

Temas

- Amazon Aurora Auto Scaling con réplicas de Aurora
- Cómo añadir una política de escalado automático a un clúster de base de datos de Amazon Aurora
- Cómo editar una política de escalado automático para un clúster de base de datos de Amazon Aurora
- Cómo eliminar una política de escalado automático de su clúster de base de datos de Amazon Aurora

Amazon Aurora Auto Scaling con réplicas de Aurora

Para cumplir sus requisitos de conectividad y carga de trabajo, Auto Scaling de Aurora ajusta de forma dinámica el número de réplicas de Aurora (instancias de base de datos de lectura) aprovisionadas para un clúster de base de datos de Aurora. El Auto Scaling de Aurora está disponible para Aurora MySQL y Aurora PostgreSQL. Auto Scaling de Aurora permite a su clúster de base de datos Aurora hacer frente a los aumentos repentinos de conectividad o carga de trabajo. Cuando la conectividad o carga de trabajo disminuye, Auto Scaling de Aurora quita réplicas de Aurora innecesarias para evitar que tenga que pagar por instancias de base de datos aprovisionadas que no se utilicen.

Puede definir y aplicar una política de escalado a un clúster de base de datos Aurora. La política de escalado define el número mínimo y máximo de réplicas de Aurora que Auto Scaling de Aurora puede administrar. En función de la política, Auto Scaling de Aurora aumenta o reduce el número de réplicas de Aurora en respuesta a las cargas de trabajo reales, determinadas mediante las métricas de Amazon CloudWatch y los valores de destino.

> **Note**
>
> Aurora Auto Scaling no se aplica a la carga de trabajo de la instancia de base de datos del escritor. Aurora Auto Scaling solo ayuda con la carga de trabajo en las instancias de lector.

Puede usar la Consola de administración de AWS para aplicar una política de escalado en función de una métrica predefinida. También puede utilizar la API de Auto Scaling de AWS CLI o Aurora para aplicar una política de escalado basada en una métrica predefinida o personalizada.

Temas

- Antes de empezar
- Políticas de Auto Scaling de Aurora
- Identificadores y etiquetado de instancias de base de datos
- Aurora Auto Scaling e Información sobre rendimiento

Antes de empezar

Antes de usar Aurora Auto Scaling con un clúster de base de datos de Aurora, primero debe crear un clúster de base de datos de Aurora con una instancia de base de datos principal (escritor). Para

obtener más información acerca de la creación de un clúster de base de datos Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora.

Aurora Auto Scaling solo escala un clúster de base de datos si este se encuentra en un estado disponible.

Cuando Auto Scaling de Aurora añade una nueva réplica de Aurora, la nueva réplica de Aurora es la misma clase de instancia de base de datos que la usada por la instancia principal. Para obtener más información acerca de las clases de instancias de bases de datos, consulte Clases de instancia de base de datos de Amazon Aurora. Además, la capa de promoción para las nuevas réplicas de Aurora se establecen en la última prioridad, que es 15 de forma predeterminada. Eso significa que durante una conmutación por error, una réplica con una mejor prioridad, como una creada manualmente, se promocionaría primero. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

Auto Scaling de Aurora solo quita las réplicas de Aurora que ha creado.

Para poder beneficiarse de Auto Scaling de Aurora, sus aplicaciones deben admitir conexiones a nuevas réplicas de Aurora. Y para hacerlo, recomendamos que utilice el punto de enlace de lector de Aurora. Puede usar un controlador como el controlador JDBC de AWS. Para obtener más información, consulte Conexión a un clúster de base de datos Amazon Aurora.

> **Note**
>
> Las bases de datos globales Aurora actualmente no admiten Aurora Auto Scaling para clústeres de bases de datos secundarios.

Políticas de Auto Scaling de Aurora

Auto Scaling de Aurora usa una política de escalado para ajustar el número de réplicas de Aurora en un clúster de base de datos Aurora. Auto Scaling de Aurora tiene los siguientes componentes:

- Una función vinculada a un servicio
- Una métrica de destino
- Capacidad mínima y máxima
- Un periodo de recuperación

Temas

- Rol vinculado a un servicio
- Métrica de destino
- Capacidad mínima y máxima
- Periodo de recuperación
- Activar o desactivar actividades de escalado descendente
- Cómo añadir, editar o eliminar políticas de escalado automático

Rol vinculado a un servicio

Auto Scaling de Aurora usa el rol vinculado a un servicio `AWSServiceRoleForApplicationAutoScaling_RDSCluster`. Para obtener más información, consulte Roles vinculados a servicios de Auto Scaling de aplicaciones en la Guía del usuario de Auto Scaling de aplicaciones.

Métrica de destino

En este tipo de política, una métrica predefinida o personalizada y un valor de destino de la métrica se especifica en una configuración de la política de escalado de seguimiento de destino. Auto Scaling de Aurora crea y administra las alarmas de CloudWatch que desencadenan la política de escalado y calcula el ajuste de escalado en función de la métrica y el valor objetivo. La política de escalado amplía o reduce las réplicas de Aurora en función de las necesidades para mantener la métrica en el valor objetivo especificado o en un valor próximo. Además de mantener la métrica próxima al valor de destino, la política de escalado de seguimiento de destino también se ajusta a las fluctuaciones de la métrica producidas por una carga de trabajo en constante cambio. Esta política también minimiza las fluctuaciones rápidas del número de réplicas de Aurora disponibles de su clúster de base de datos.

Por ejemplo, adopte una política de escalado que use la métrica de utilización de la CPU media predefinida. Esta política puede mantener la utilización de la CPU en el porcentaje de utilización especificado o en un valor próximo, como el 40 por ciento.

> **Note**
>
> Para cada clúster de base de datos Aurora, puede crear solo una política de Auto Scaling para cada métrica de destino.

Al configurar el escalado automático de Aurora, el valor de la métrica de destino se calcula como la media de todas las instancias de lectura del clúster. Este cálculo se realiza como se indica a continuación:

- Incluye todas las instancias de lectura del clúster de Aurora, independientemente de si se administran mediante escalado automático o se agregan manualmente.
- Incluye instancias asociadas a puntos de conexión personalizados. Los puntos de conexión personalizados no influyen en el cálculo de las métricas de destino.
- No incluye la instancia de escritura del clúster.

Las métricas se derivan de CloudWatch mediante las siguientes dimensiones:

- `DBClusterIdentifier`
- `Role=READER`

Por ejemplo, considere un clúster de Aurora MySQL con la siguiente configuración:

- Instancias manuales (no controladas por escalado automático):
  - Escritor con un 50 % de uso de la CPU
  - Lector 1 (punto de conexión personalizado: `custom-reader-1`) con un 90 % de utilización de la CPU
  - Lector 2 (punto de conexión personalizado: `custom-reader-2`) con un 90 % de utilización de la CPU
- Instancia de escalado automático:
  - Lector 3 (agregado mediante escalado automático) con un 10 % de uso de la CPU

En este escenario, la métrica de destino de la política de escalado automático se calcula de la siguiente manera:

```
Target metric = (CPU utilization of reader 1 + reader 2 + reader 3) / total number of
 readers

Target metric = (90 + 90 + 10) / 3 = 63.33%
```

La política de escalado automático utiliza este valor para evaluar si se debe reducir horizontalmente o escalar horizontalmente en función del umbral definido.

Considere lo siguiente:

- Aunque los puntos de conexión personalizados determinan cómo se redirige el tráfico a lectores específicos, no excluyen a los lectores del cálculo de las métricas.
- Las instancias manuales siempre se incluyen en los cálculos de las métricas de destino.
- Para evitar un comportamiento de escalado inesperado, asegúrese de que la configuración de escalado automático tenga en cuenta todas las instancias de lectura del clúster.
- Si un clúster no tiene lectores, la métrica no se calcula y las alarmas de la política de escalado automático permanecen inactivas. Para que la política de escalado automático funcione eficazmente, debe haber al menos un lector presente en todo momento.

Capacidad mínima y máxima

Puede especificar el número máximo de réplicas de Aurora que Auto Scaling de aplicaciones va a administrar. Este valor debe establecerse en 0–15 y debe ser igual o superior al valor especificado para el número mínimo de réplicas de Aurora.

También puede especificar el número mínimo de réplicas de Aurora que Auto Scaling de aplicaciones va a administrar. Este valor debe establecerse en 0–15 y debe ser igual o inferior al valor especificado para el número máximo de réplicas de Aurora.

Debe haber al menos una instancia de base de datos de lector para que Aurora Auto Scaling funcione. Si el clúster de base de datos no tiene una instancia de lectura y usted establece la capacidad mínima en 0, Aurora Auto Scaling no funcionará.

> **Note**
>
> La capacidad mínima y máxima se establece para un clúster de base de datos Aurora. Los valores especificados se aplican a todas las políticas asociadas al clúster de base de datos Aurora.

Periodo de recuperación

Puede ajustar la capacidad de respuesta de una política de escalado de seguimiento de destino añadiendo periodos de recuperación que afecten a la ampliación o reducción de su clúster de base de datos Aurora. Un periodo de recuperación bloquea solicitudes de escalado descendente o

ascendente posteriores hasta que vence el periodo. Estos bloques ralentizan las eliminaciones de réplicas de Aurora en su clúster de base de datos Aurora para solicitudes de escalado descendente y la creación de réplicas de Aurora para solicitudes de escalado ascendente.

Puede especificar los siguientes periodos de recuperación:

- Una actividad de escalado descendente reduce el número de réplicas de Aurora en su clúster de base de datos Aurora. Un periodo de recuperación de escalado descendente especifica la cantidad de tiempo, en segundos, tras completarse una actividad de escalado descendente antes de que pueda comenzar otra actividad de escalado descendente.
- Una actividad de escalado ascendente incrementa el número de réplicas de Aurora en su clúster de base de datos Aurora. Un periodo de recuperación de escalado ascendente especifica la cantidad de tiempo, en segundos, tras completarse una actividad de escalado ascendente antes de que pueda comenzar otra actividad de escalado ascendente.

> **Note**
>
> Se ignora un periodo de recuperación de escalado horizontal si una solicitud de escalado horizontal posterior es para un número mayor de réplicas de Aurora que la primera solicitud.

Si no establece el periodo de recuperación de escalado horizontal o vertical, el valor por defecto de cada uno es de 300 segundos.

Activar o desactivar actividades de escalado descendente

Puede habilitar o deshabilitar actividades de escalado descendente para una política. La habilitación de actividades de escalado descendente permite a la política de escalado eliminar réplicas de Aurora. Al habilitarse actividades de escalado descendente, el periodo de recuperación de escalado descendente de la política de escalado se aplica a las actividades de escalado descendente. La deshabilitación de actividades de escalado descendente impide a la política de escalado eliminar réplicas de Aurora.

> **Note**
>
> Las actividades de escalado ascendente siempre se habilitan de modo que la política de escalado pueda crear réplicas de Aurora según sea necesario.

Cómo añadir, editar o eliminar políticas de escalado automático

Puede añadir, editar o eliminar políticas de escalado automático utilizando la Consola de administración de AWS, la AWS CLI o la API de escalado automático de aplicaciones. Para obtener más información sobre cómo añadir, editar o eliminar políticas de escalado automático, consulte las siguientes secciones.

- Cómo añadir una política de escalado automático a un clúster de base de datos de Amazon Aurora
- Cómo editar una política de escalado automático para un clúster de base de datos de Amazon Aurora
- Cómo eliminar una política de escalado automático de su clúster de base de datos de Amazon Aurora

Identificadores y etiquetado de instancias de base de datos

Cuando se agrega una réplica mediante Aurora Auto Scaling, su ID de instancia de base de datos tiene el prefijo `application-autoscaling-`, por ejemplo, `application-autoscaling-61aabbcc-4e2f-4c65-b620-ab7421abc123`.

La siguiente etiqueta se agrega automáticamente a la instancia de base de datos. Puede verla en la pestaña Etiquetas de la página de detalles de la instancia de base de datos.

Tag	Valor
application-autoscaling:resourceId	clúster:mynewclúster-clúster

Para obtener más información acerca de las etiquetas de recursos Amazon RDS, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.

Aurora Auto Scaling e Información sobre rendimiento

Puede utilizar Información sobre rendimiento para supervisar las réplicas que ha añadido Aurora Auto Scaling, igual que con cualquier instancia de base de datos de lector de Aurora.

Para obtener más información acerca de cómo usar Información sobre rendimiento para supervisar los clústeres de base de datos de Aurora, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

Cómo añadir una política de escalado automático a un clúster de base de datos de Amazon Aurora

Puede agregar una política de escalado utilizando la Consola de administración de AWS, AWS CLI o la API de Auto Scaling de aplicaciones.

> **Note**
>
> Para ver un ejemplo que agrega una política de escalado mediante CloudFormation, consulte Declaración de una política de escalado para un clúster de base de datos de Aurora en la Guía del usuario de AWS CloudFormation.

Consola

Puede agregar una política de escalado a un clúster de base de datos de Aurora mediante la Consola de administración de AWS.

Para añadir una política de Auto Scaling a un clúster de base de datos Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el clúster de base de datos Aurora para el que desea añadir una política.
4. Seleccione la pestaña Logs & events (Registros y eventos).
5. En la sección Auto scaling policies (Políticas de Auto Scaling), elija Add (Añadir).

   Aparecerá el cuadro de diálogo Add Auto Scaling policy (Añadir política de Auto Scaling).
6. En Policy name (Nombre de la política), escriba un nombre para la política.
7. Para la métrica de destino, elija una de las siguientes opciones:

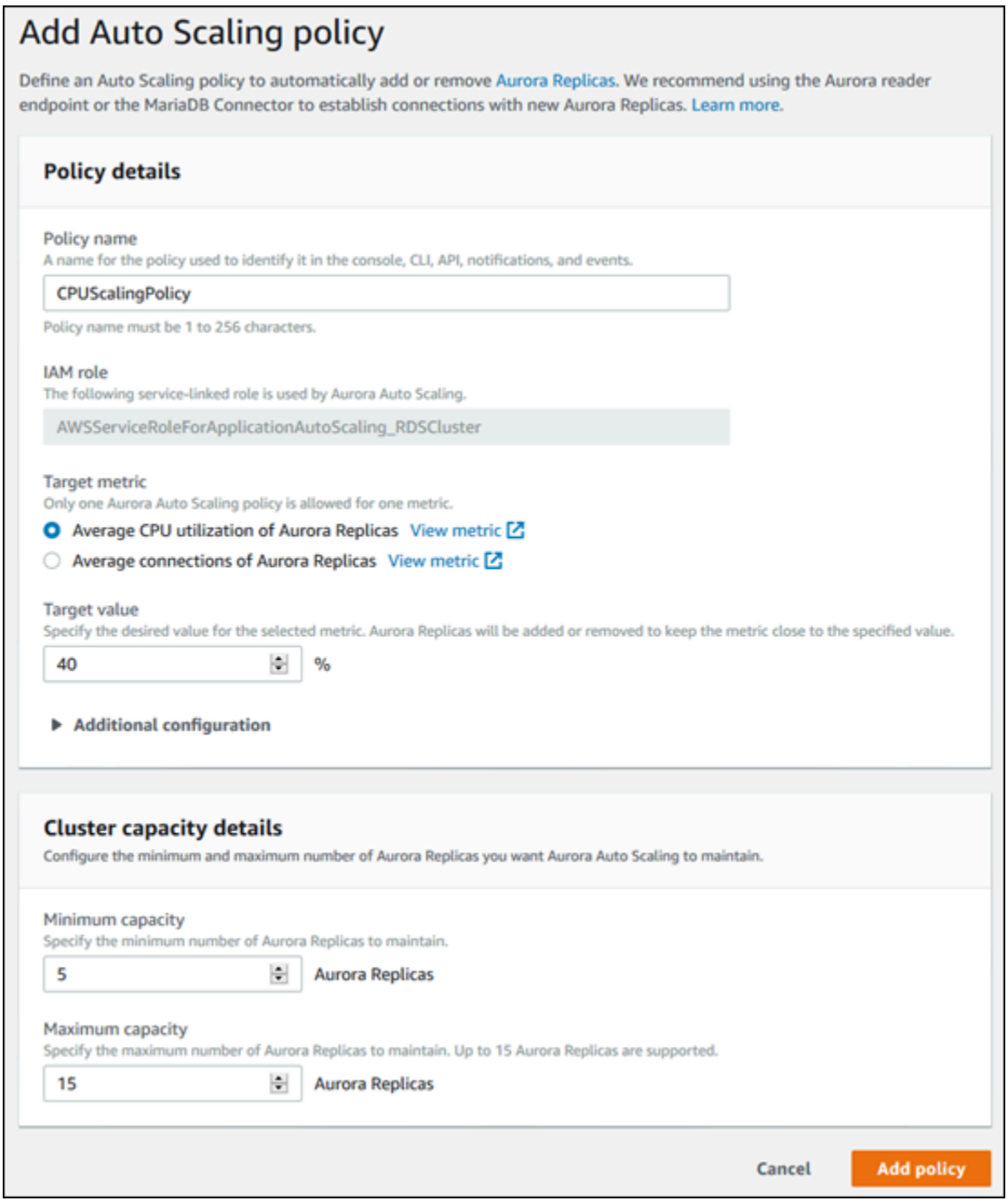
   - Average CPU utilization of Aurora Replicas (Utilización media de CPU de réplicas de Aurora) para crear una política basada en el uso medio de la CPU.
   - Average connections of Aurora Replicas (Promedio de conexiones de réplicas de Aurora) para crear una política basada en el número medio de conexiones a réplicas de Aurora.
8. Para el valor de destino, escriba una de las siguientes opciones:

- Si eligió Average CPU utilization of Aurora Replicas (Utilización media de CPU de réplicas de Aurora) en el paso anterior, escriba el porcentaje de utilización de CPU que desea mantener en las réplicas de Aurora.
- Si eligió Average connections of Aurora Replicas (Promedio de conexiones de réplicas de Aurora) en el paso anterior, escriba el número de conexiones que desea mantener.

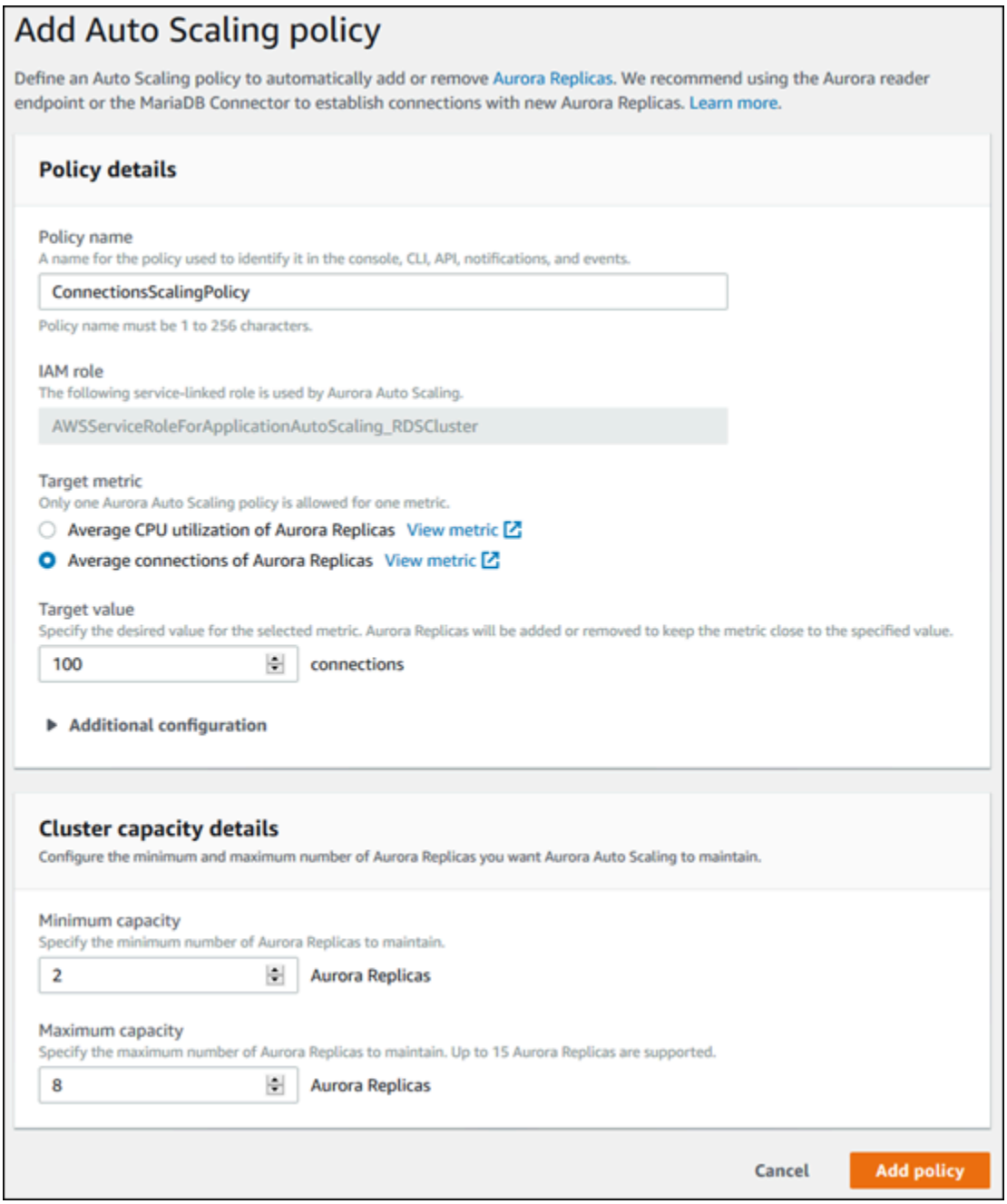
Las réplicas de Aurora se añaden o quitan para mantener la métrica en un valor próximo al especificado.

9. (Opcional) Expanda Additional Configuration (Configuración adicional) para crear un periodo de recuperación de escalado vertical u horizontal.
10. Para Minimum capacity (Capacidad mínima), escriba el número mínimo de réplicas de Aurora que debe mantener la política de Auto Scaling de Aurora.
11. Para Maximum capacity (Capacidad máxima), escriba el número máximo de réplicas de Aurora que debe mantener la política de Auto Scaling de Aurora.
12. Elija Add policy (Agregar política).

El siguiente cuadro de diálogo crea una política de Auto Scaling basada en un uso medio de la CPU del 40 por ciento. En la política se especifica un mínimo de 5 réplicas de Aurora y un máximo de 15 réplicas de Aurora.



En el siguiente cuadro de diálogo se crea una política de Auto Scaling basada en un número medio de conexiones de 100. En la política se especifica un mínimo de dos réplicas de Aurora y un máximo de ocho réplicas de Aurora.



AWS CLI o API Application Auto Scaling

Puede aplicar una política de escalado en función de una métrica predefinida o una personalizada. Para ello, puede usar AWS CLI o la API de Auto Scaling de aplicaciones. El primer paso consiste en registrar su clúster de base de datos Aurora con Auto Scaling de aplicaciones.

Registro de un clúster de base de datos Aurora

Antes de que pueda usar Auto Scaling de Aurora con un clúster de base de datos Aurora, puede registrar su clúster de base de datos Aurora con Auto Scaling de aplicaciones. Esto se hace para definir la dimensión y los límites de escalado que se van a aplicar a ese clúster. La aplicación Auto Scaling escala dinámicamente el clúster de base de datos de Aurora a lo largo de la `rds:cluster:ReadReplicaCount` dimensión escalable, que representa el número de réplicas de Aurora.

Para registrar su clúster de base de datos Aurora, puede usar la AWS CLI o la API de Auto Scaling de aplicaciones.

AWS CLI

Para registrar un clúster de base de datos de Aurora, utilice el comando `register-scalable-target` de AWS CLI con los siguientes parámetros:

- `--service-namespace` – ajuste este valor en `rds`.
- `--resource-id`: identificador de recurso para el clúster de base de datos Aurora. Para este parámetro, el tipo de recurso es `cluster` y el identificador único es el nombre del clúster de base de datos Aurora, por ejemplo `cluster:myscalablecluster`.
- `--scalable-dimension` – ajuste este valor en `rds:cluster:ReadReplicaCount`.
- `--min-capacity`: el número mínimo de instancias de base de datos de lector que Auto Scaling de aplicaciones va a administrar. Para obtener información sobre la relación entre `--min-capacity`, `--max-capacity`, y el número de instancias de bases de datos del clúster, consulte Capacidad mínima y máxima.
- `--max-capacity`: el número máximo de instancias de base de datos de lector que Auto Scaling de aplicaciones va a administrar. Para obtener información sobre la relación entre `--min-capacity`, `--max-capacity`, y el número de instancias de bases de datos del clúster, consulte Capacidad mínima y máxima.

Example

En el siguiente ejemplo, registra un clúster de base de datos Aurora denominado `myscalablecluster`. El registro indica que el clúster de base de datos debe escalarse dinámicamente para tener de una a ocho réplicas de Aurora.

Para Linux, macOS o:Unix

```
aws application-autoscaling register-scalable-target \
    --service-namespace rds \
    --resource-id cluster:myscalablecluster \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --min-capacity 1 \
    --max-capacity 8 \
```

En:Windows

```
aws application-autoscaling register-scalable-target ^
    --service-namespace rds ^
    --resource-id cluster:myscalablecluster ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --min-capacity 1 ^
    --max-capacity 8 ^
```

API Application Auto Scaling

Para registrar un clúster de base de datos de Aurora con Auto Scaling de aplicaciones, use la operación `RegisterScalableTarget` de la API de Auto Scaling de aplicaciones con los siguientes parámetros:

- `ServiceNamespace` – ajuste este valor en `rds`.
- `ResourceID`: identificador de recurso para el clúster de base de datos Aurora. Para este parámetro, el tipo de recurso es `cluster` y el identificador único es el nombre del clúster de base de datos Aurora, por ejemplo `cluster:myscalablecluster`.
- `ScalableDimension` – ajuste este valor en `rds:cluster:ReadReplicaCount`.
- `MinCapacity`: el número mínimo de instancias de base de datos de lector que Auto Scaling de aplicaciones va a administrar. Para obtener información sobre la relación entre `MinCapacity`, `MaxCapacity`, y el número de instancias de bases de datos del clúster, consulte Capacidad mínima y máxima.
- `MaxCapacity`: el número máximo de instancias de base de datos de lector que Auto Scaling de aplicaciones va a administrar. Para obtener información sobre la relación entre `MinCapacity`, `MaxCapacity`, y el número de instancias de bases de datos del clúster, consulte Capacidad mínima y máxima.

Example

En el siguiente ejemplo, registra un clúster de base de datos Aurora denominado `myscalablecluster` con la API de Auto Scaling de aplicaciones. Este registro indica que el clúster de base de datos debe escalarse dinámicamente para tener de una a ocho réplicas de Aurora.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "MinCapacity": 1,
    "MaxCapacity": 8
}
```

Definición de una política de escalado para un clúster de base de datos Aurora

Una configuración de la política de escalado de seguimiento de destino está representada por un bloque JSON en el que se definen las métricas y los valores de destino. Puede guardar una configuración de la política de escalado como bloque JSON en un archivo de texto. Puede utilizarr ese archivo de texto al invocar AWS CLI o la API de Auto Scaling de aplicaciones. Para obtener más información acerca de la sintaxis de configuración de la política, consulte `TargetTrackingScalingPolicyConfiguration` en la referencia de la API de Auto Scaling de aplicaciones.

Las siguientes opciones están disponibles para definir una configuración de la política de escalado de seguimiento de destino.

Temas

- Uso de una métrica predefinida
- Uso de una métrica personalizada

- Uso de periodos de recuperación
- Desactivación de actividad de escalado descendente

Uso de una métrica predefinida

Mediante las métricas predefinidas, puede definir rápidamente una política de escalado de seguimiento de destino para un clúster de base de datos Aurora que funciona bien tanto con el seguimiento de destino como con el escalado dinámico en Auto Scaling de Aurora.

Actualmente, Aurora admite las siguientes métricas predefinidas en Auto Scaling de Aurora:

- RDSReaderAverageCPUUtilization: el valor medio de la métrica de `CPUUtilization` en CloudWatch en todas las réplicas de Aurora del clúster de base de datos Aurora.
- RDSReaderAverageDatabaseConnections: el valor medio de la métrica de `DatabaseConnections` en CloudWatch en todas las réplicas de Aurora del clúster de base de datos Aurora.

Para obtener más información sobre las métricas de `CPUUtilization` y `DatabaseConnections`, consulte Métricas de Amazon CloudWatch para Amazon Aurora.

Para usar una métrica predefinida en su política de escalado, puede crear una configuración de seguimiento de destino para su política de escalado. Esta configuración debe incluir `PredefinedMetricSpecification` para la métrica predefinida y `TargetValue` para el valor de destino de esa métrica.

Example

En el siguiente ejemplo se describe una configuración de la política típica para el escalado de seguimiento de destino de un clúster de base de datos Aurora. En esta configuración, la métrica predefinida `RDSReaderAverageCPUUtilization` se usa para ajustar el clúster de base de datos Aurora en función del uso medio de la CPU del 40 por ciento en todas las réplicas de Aurora.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    }
}
```

Uso de una métrica personalizada

Mediante las métricas personalizadas, puede definir una política de escalado de seguimiento de destino que cumpla sus requisitos personalizados. Puede definir una métrica personalizada en función de cualquier métrica de Aurora que cambie en proporción al escalado.

No todas las métricas de Aurora funcionan para el seguimiento de destino. La métrica debe ser una métrica de utilización válida y describir el nivel de actividad de una instancia. El valor de la métrica debe aumentar o reducirse en proporción al número de réplicas de Aurora del clúster de base de datos Aurora. Este aumento o reducción proporcionales son necesarios para usar los datos de las métricas a fin de ampliar o reducir proporcionalmente el número de réplicas de Aurora.

Example

En el siguiente ejemplo se describe una configuración de seguimiento de destino para una política de escalado. En esta configuración, una métrica personalizada ajusta un clúster de base de datos Aurora en función de un uso medio de la CPU del 50 % en todas las réplicas de Aurora de un clúster de base de datos Aurora denominado `my-db-cluster`.

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/RDS",
        "Dimensions": [
            {"Name": "DBClusterIdentifier","Value": "my-db-cluster"},
            {"Name": "Role","Value": "READER"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

Uso de periodos de recuperación

Puede especificar un valor, en segundos, a fin de que `ScaleOutCooldown` añada un periodo de recuperación para el escalado ascendente de su clúster de base de datos Aurora. De forma similar, puede añadir un valor, en segundos, a fin de que `ScaleInCooldown` añada un periodo de recuperación para el escalado descendente de su clúster de base de datos Aurora. Para obtener más información acerca de `ScaleInCooldown` y `ScaleOutCooldown`, consulte

TargetTrackingScalingPolicyConfiguration en la referencia de la API de Auto Scaling de aplicaciones.

Example

En el siguiente ejemplo se describe una configuración de seguimiento de destino para una política de escalado. En esta configuración, la métrica predefinida RDSReaderAverageCPUUtilization se usa para ajustar un clúster de base de datos de Aurora en función del uso promedio de la CPU del 40 por ciento en todas las réplicas de Aurora de ese clúster de base de datos de Aurora. La configuración proporciona un periodo de recuperación de escalado descendente de 10 minutos y un periodo de recuperación de escalado ascendente de 5 minutos.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Desactivación de actividad de escalado descendente

Puede evitar que la configuración de la política de escalado de seguimiento de destino escale de forma descendente su clúster de base de datos Aurora deshabilitando la actividad de escalado descendente. La deshabilitación de la actividad de escalado descendente evita que la política de escalado elimine réplicas de Aurora, a la vez que permite a la política de escalado crearlas según sea necesario.

Puede especificar un valor booleano para que DisableScaleIn habilite o deshabilite la actividad de escalado descendente para su clúster de base de datos Aurora. Para obtener más información acerca de DisableScaleIn, consulte TargetTrackingScalingPolicyConfiguration en la referencia de la API de Auto Scaling de aplicaciones.

Example

En el siguiente ejemplo se describe una configuración de seguimiento de destino para una política de escalado. En esta configuración, la métrica predefinida RDSReaderAverageCPUUtilization ajusta un clúster de base de datos Aurora en función del uso medio de la CPU del 40 % en todas las

réplicas de Aurora de ese clúster de base de datos Aurora. La configuración deshabilita la actividad de escalado descendente para la política de escalado.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Aplicación de una política de escalado a un clúster de base de datos Aurora

Tras registrar su clúster de base de datos Aurora con Auto Scaling de aplicaciones y definir una política de escalado, puede aplicar esta al clúster de base de datos Aurora registrado. Para aplicar una política de escalado a un clúster de base de datos Aurora, puede usar la AWS CLI o la API de Auto Scaling de aplicaciones.

AWS CLI

Para aplicar una política de escalado a un clúster de base de datos de Aurora, use el comando `put-scaling-policy` de AWS CLI con los siguientes parámetros:

- `--policy-name`: el nombre de la política de escalado.
- `--policy-type` – ajuste este valor en `TargetTrackingScaling`.
- `--resource-id`: identificador de recurso para el clúster de base de datos Aurora. Para este parámetro, el tipo de recurso es `cluster` y el identificador único es el nombre del clúster de base de datos Aurora, por ejemplo `cluster:myscalablecluster`.
- `--service-namespace` – ajuste este valor en `rds`.
- `--scalable-dimension` – ajuste este valor en `rds:cluster:ReadReplicaCount`.
- `--target-tracking-scaling-policy-configuration`: configuración de la política de escalado de seguimiento de destino que se usará para el clúster de base de datos Aurora.

Example

En el siguiente ejemplo, aplica una política de escalado de seguimiento de destino denominada `myscalablepolicy` a un clúster de base de datos Aurora llamado `myscalablecluster` con

Auto Scaling de aplicaciones. Para ello, puede usar una configuración de la política guardada en un archivo denominado `config.json`.

Para Linux, macOS o:Unix

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --target-tracking-scaling-policy-configuration file://config.json
```

En:Windows

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --target-tracking-scaling-policy-configuration file://config.json
```

API Application Auto Scaling

Para aplicar una política de escalado a un clúster de base de datos de Aurora con la API de Auto Scaling de aplicaciones, utilice la operación `PutScalingPolicy` de la API de Auto Scaling de aplicaciones con los siguientes parámetros:

- `PolicyName`: el nombre de la política de escalado.
- `ServiceNamespace` – ajuste este valor en `rds`.
- `ResourceID`: identificador de recurso para el clúster de base de datos Aurora. Para este parámetro, el tipo de recurso es `cluster` y el identificador único es el nombre del clúster de base de datos Aurora, por ejemplo `cluster:myscalablecluster`.
- `ScalableDimension` – ajuste este valor en `rds:cluster:ReadReplicaCount`.
- `PolicyType` – ajuste este valor en `TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration`: configuración de la política de escalado de seguimiento de destino que se usará para el clúster de base de datos Aurora.

Example

En el siguiente ejemplo, aplica una política de escalado de seguimiento de destino denominada `myscalablepolicy` a un clúster de base de datos Aurora llamado `myscalablecluster` con Auto Scaling de aplicaciones. Puede usar una configuración de la política en función de la métrica predefinida `RDSReaderAverageCPUUtilization`.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
        }
    }
}
```

Cómo editar una política de escalado automático para un clúster de base de datos de Amazon Aurora

Puede editar una política de escalado utilizando la Consola de administración de AWS, la AWS CLI o la API de Auto Scaling de aplicaciones.

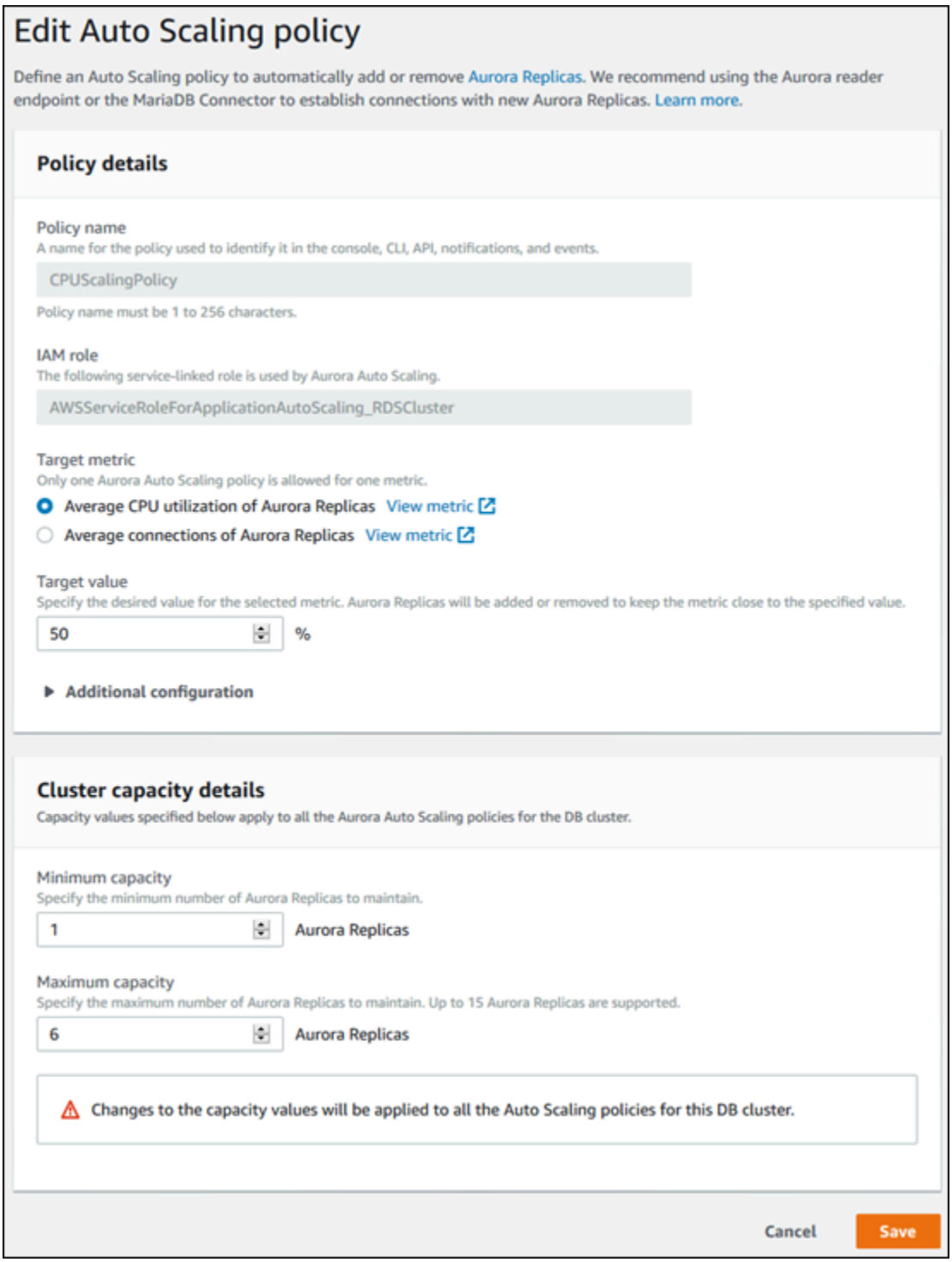
Consola

Puede editar una política de escalado mediante la Consola de administración de AWS.

Para editar una política de Auto Scaling para un clúster de base de datos Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos Aurora con la política de Auto Scaling que desea editar.
4. Seleccione la pestaña Logs & events (Registros y eventos).
5. En la sección Auto Scaling Policies (Políticas de Auto Scaling), elija la política de Auto Scaling y, a continuación, seleccione Edit (Editar).
6. Realice cambios en la política.
7. Seleccione Save.

A continuación, se muestra un cuadro de diálogo Edit Auto Scaling policy (Editar política de Auto Scaling) de ejemplo.



AWS CLI o API Application Auto Scaling

Puede utilizar AWS CLI o la API de Auto Scaling de aplicaciones para editar una política de escalado de la misma forma que aplica una política de escalado:

- Al usar la AWS CLI, especifique el nombre de la política que desea editar en el parámetro `--policy-name`. Especifique nuevos valores para los parámetros que desea cambiar.
- Al usar la API de Auto Scaling de aplicaciones, especifique el nombre de la política que desea editar en el parámetro `PolicyName`. Especifique nuevos valores para los parámetros que desea cambiar.

Para obtener más información, consulte Aplicación de una política de escalado a un clúster de base de datos Aurora.

Cómo eliminar una política de escalado automático de su clúster de base de datos de Amazon Aurora

Puede eliminar una política de escalado utilizando la Consola de administración de AWS, AWS CLI o la API de Auto Scaling de aplicaciones.

Consola

Puede eliminar una política de escalado mediante la Consola de administración de AWS.

Para eliminar una política de Auto Scaling para un clúster de base de datos Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos Aurora con la política de Auto Scaling que desea eliminar.
4. Seleccione la pestaña Logs & events (Registros y eventos).
5. En la sección Auto Scaling Policies (Políticas de Auto Scaling), elija la política de Auto Scaling y, a continuación, seleccione Delete (Eliminar).

AWS CLI

Para eliminar una política de escalado de su clúster de base de datos de Aurora, use el comando `delete-scaling-policy` de AWS CLI con los siguientes parámetros:

- `--policy-name`: el nombre de la política de escalado.

- `--resource-id`: identificador de recurso para el clúster de base de datos Aurora. Para este parámetro, el tipo de recurso es `cluster` y el identificador único es el nombre del clúster de base de datos Aurora, por ejemplo `cluster:myscalablecluster`.
- `--service-namespace` – ajuste este valor en `rds`.
- `--scalable-dimension` – ajuste este valor en `rds:cluster:ReadReplicaCount`.

Example

En el siguiente ejemplo, elimina una política de escalado de seguimiento de destino denominada `myscalablepolicy` de un clúster de base de datos Aurora llamado `myscalablecluster`.

Para Linux, macOS o:Unix

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

En:Windows

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
```

API Application Auto Scaling

Para eliminar una política de escalado de su clúster de base de datos Aurora, use la operación `DeleteScalingPolicy` de la API de Auto Scaling de aplicaciones con los siguientes parámetros:

- `PolicyName`: el nombre de la política de escalado.
- `ServiceNamespace` – ajuste este valor en `rds`.

- `ResourceID`: identificador de recurso para el clúster de base de datos Aurora. Para este parámetro, el tipo de recurso es `cluster` y el identificador único es el nombre del clúster de base de datos Aurora, por ejemplo `cluster:myscalablecluster`.
- `ScalableDimension` – ajuste este valor en `rds:cluster:ReadReplicaCount`.

Example

En el siguiente ejemplo, elimina una política de escalado de seguimiento de destino denominada `myscalablepolicy` de un clúster de base de datos Aurora llamado `myscalablecluster` con la API de Auto Scaling de aplicaciones.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount"
}
```

Administración del rendimiento y el escalado para clústeres de base de datos Aurora

Puede utilizar las siguientes opciones para administrar el desempeño y el escalado de clústeres e instancias de bases de datos Aurora:

Temas

- Escalado del almacenamiento
- Escalado de instancia
- Escalado de lectura
- Administración de conexiones
- Administración de planes de ejecución de consultas

Escalado del almacenamiento

El almacenamiento de Aurora se escala automáticamente con los datos del volumen de clúster. A medida que los datos crecen, el volumen de almacenamiento del clúster se amplía hasta un máximo de 128 tebibytes (TiB) o 64 TiB. El tamaño máximo depende de la versión del motor de base de datos. Para obtener información sobre qué tipos de datos se incluyen en el volumen del clúster, consulte Almacenamiento de Amazon Aurora. Para obtener información detallada sobre el tamaño máximo de una versión específica, consulte Límites de tamaño de Amazon Aurora.

El tamaño del volumen de clúster se evalúa cada hora para determinar los costos de almacenamiento. Para obtener información sobre los precios, consulte la página de precios de Aurora.

Aunque un volumen de clúster de Aurora puede aumentar en tamaño a muchos tebibytes, solo se le cobrará el espacio que utilice en el volumen. El mecanismo para determinar el espacio de almacenamiento facturado depende de la versión del clúster de Aurora.

- Cuando se eliminan datos de Aurora del volumen del clúster, el espacio facturado general disminuye en una cantidad comparable. Este comportamiento dinámico de cambio de tamaño se produce cuando los espacios de tabla subyacentes se eliminan o se reorganizan para requerir menos espacio. De esta manera, podrá reducir los gastos de almacenamiento eliminando las tablas y bases de datos que ya no necesite. El cambio de tamaño dinámico se aplica a ciertas

versiones de Aurora. Estas son las versiones de Aurora en las que el volumen del clúster cambia de tamaño dinámicamente a medida que se eliminan los datos:

Motor de base de datos	Versiones con redimensionamiento dinámico
Aurora MySQL	• Versión 3 (compatible con MySQL 8.0): todas las versiones admitidas • Versión 2 (compatible con MySQL 5.7): 2.11 y posteriores
Aurora PostgreSQL	Todas las versiones compatibles
Aurora Serverless v2	Todas las versiones compatibles
Aurora Serverless v1	Todas las versiones compatibles

- En versiones de Aurora anteriores a las de la lista anterior, el volumen del clúster puede reutilizar el espacio liberado al eliminar datos, pero el volumen en sí nunca disminuye de tamaño.
- Esta característica se está implementando por fases en las regiones de AWS donde está disponible Aurora. Dependiendo de la región donde se encuentre el clúster, es posible que esta característica no esté disponible todavía.

El cambio de tamaño dinámico se aplica a operaciones que eliminan o redimensionan físicamente los espacios de tablas dentro del volumen del clúster. Por lo tanto, se aplica a instrucciones SQL como `DROP TABLE`, `DROP DATABASE`, `TRUNCATE TABLE` y `ALTER TABLE ... DROP PARTITION`. No se aplica a la eliminación de filas utilizando la instrucción `DELETE`. Si elimina un gran número de filas de una tabla, puede ejecutar la instrucción `OPTIMIZE TABLE` de Aurora MySQL o utilizar después la extensión `pg_repack` de Aurora PostgreSQL para reorganizar la tabla y cambiar dinámicamente el volumen del clúster.

Para Aurora MySQL, se aplican los siguientes aspectos:

- Tras actualizar el clúster de base de datos a una versión de motor de base de datos que admita el cambio de tamaño dinámico, y cuando la característica esté habilitada en esa Región de AWS específica, se podrá recuperar cualquier espacio que se libere posteriormente mediante determinadas instrucciones SQL, como `DROP TABLE`.

Si la característica está deshabilitada de forma explícita en una Región de AWS concreta, es posible que el espacio solo se pueda reutilizar (y no se pueda recuperar) incluso en versiones que admitan el cambio de tamaño dinámico.

La característica se habilitó para versiones específicas del motor de base de datos (1.23.0–1.23.4, 2.09.0–2.09.3 y 2.10.0) entre noviembre de 2020 y marzo de 2022, y está habilitada de forma predeterminada en todas las versiones posteriores.

- Una tabla se almacena internamente en uno o más fragmentos contiguos de distintos tamaños. Mientras se ejecutan operaciones `TRUNCATE TABLE`, el espacio correspondiente al primer fragmento se puede reutilizar y no se puede recuperar. Los demás fragmentos se pueden recuperar. Durante las operaciones `DROP TABLE`, se puede recuperar el espacio correspondiente a todo el espacio de la tabla.
- El parámetro `innodb_file_per_table` afecta a cómo se organiza el almacenamiento de tablas. Cuando las tablas forman parte del tablespace del sistema, la eliminación de la tabla no reduce el tamaño del tablespace del sistema. Por lo tanto, asegúrese establecer `innodb_file_per_table` en 1 para clústeres de base de datos de Aurora MySQL para aprovechar al máximo el cambio de tamaño dinámico.
- Para la versión 2.11 y versiones posteriores, el espacio de tablas temporal de InnoDB se elimina y se vuelve a crear al reiniciar. Esto libera al sistema el espacio ocupado por el espacio de tablas temporal y, a continuación, el volumen del clúster cambia de tamaño. Para aprovechar al máximo la característica de cambio de tamaño dinámico, le recomendamos que actualice su clúster de base de datos a la versión 2.11 o una versión posterior.

> **Note**
>
> La característica de cambio de tamaño dinámico no recupera espacio inmediatamente cuando se eliminan las tablas de los tablespaces, sino de forma gradual a un ritmo de aproximadamente 10 TB por día. El espacio en el espacio de tablas del sistema no se recupera porque el espacio de tablas del sistema nunca se elimina. El espacio libre no recuperado de un espacio de tablas se reutiliza cuando una operación necesita espacio en ese espacio de tablas. La característica de cambio de tamaño dinámico puede recuperar espacio de almacenamiento solo cuando el clúster tenga el estado disponible.

Puede comprobar cuánto espacio de almacenamiento está utilizando un clúster monitoreando la métrica `VolumeBytesUsed` en CloudWatch. Para obtener más información acerca de la facturación del almacenamiento, consulte Cómo se factura el almacenamiento de datos de Aurora.

- En la Consola de administración de AWS, puede ver esta cifra en un gráfico consultando la pestaña `Monitoring` en la página de detalles del clúster.
- Con la AWS CLI, puede ejecutar un comando similar al siguiente ejemplo de Linux. Sustituya sus propios valores por las horas de inicio y finalización y el nombre del clúster.

```
aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '6 hours ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" \
  --statistics Average Maximum Minimum \
  --dimensions Name=DBClusterIdentifier,Value=my_cluster_identifier
```

El resultado de este comando debería ser similar al siguiente.

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T21:25:00+00:00",
            "Average": 182871982080.0,
            "Minimum": 182871982080.0,
            "Maximum": 182871982080.0,
            "Unit": "Bytes"
        }
    ]
}
```

Los siguientes ejemplos muestran cómo puede realizar un seguimiento del uso de almacenamiento de un clúster de Aurora a lo largo del tiempo mediante comandos de AWS CLI en un sistema Linux. Los parámetros `--start-time` y `--end-time` definen el intervalo de tiempo general como un día. El parámetro `--period` solicita las mediciones a intervalos de una hora. No tiene sentido elegir un valor de `--period` pequeño, porque las métricas se recopilan a intervalos, no de forma continua. Además, las operaciones de almacenamiento de Aurora a veces continúan durante algún tiempo en segundo plano después de que finalice la instrucción SQL pertinente.

El primer ejemplo devuelve la salida en el formato JSON predeterminado. Los puntos de datos se devuelven en orden arbitrario, no ordenados por marca de tiempo. Puede importar estos datos JSON en una herramienta de gráficos para ordenar y visualizar.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T19:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T00:40:00+00:00",
            "Maximum": 198573719552.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T05:40:00+00:00",
            "Maximum": 206827454464.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-04T17:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
... output omitted ...
```

Este ejemplo devuelve los mismos datos que el anterior. El parámetro `--output` representa los datos en formato de texto no cifrado compacto. El comando `aws cloudwatch` canaliza su salida al comando `sort`. El parámetro `-k` del comando `sort` ordena la salida por el tercer campo, que es la marca de hora en formato UTC (Tiempo universal coordinado).

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id \
```

```
  --output text | sort -k 3
VolumeBytesUsed
DATAPOINTS  182872522752.0  2020-08-04T17:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T18:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T19:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T20:41:00+00:00 Bytes
DATAPOINTS  187667791872.0  2020-08-04T21:41:00+00:00 Bytes
DATAPOINTS  190981029888.0  2020-08-04T22:41:00+00:00 Bytes
DATAPOINTS  195587244032.0  2020-08-04T23:41:00+00:00 Bytes
DATAPOINTS  201048915968.0  2020-08-05T00:41:00+00:00 Bytes
DATAPOINTS  205368492032.0  2020-08-05T01:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T02:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T03:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T04:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T05:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T06:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T07:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T08:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T09:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T10:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T11:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T12:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T13:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T14:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T15:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T16:41:00+00:00 Bytes
```

La salida ordenada muestra cuánto almacenamiento se utilizó al inicio y al final del período de monitoreo. También puede encontrar los puntos durante ese período cuando Aurora asigna más almacenamiento para el clúster. En el siguiente ejemplo se utilizan comandos Linux para volver a dar formato a los valores `VolumeBytesUsed` inicial y final como gigabytes (GB) y como gibibytes (GiB). Los gigabytes representan unidades medidas en potencias de 10 y se utilizan comúnmente en explicaciones sobre el almacenamiento de discos duros rotacionales. Gibibytes representan unidades medidas en potencias de 2. Las mediciones y los límites de almacenamiento de Aurora se indican normalmente en las unidades de potencia de 2, como gibibytes y tebibytes.

```
$ GiB=$((1024*1024*1024))
$ GB=$((1000*1000*1000))
$ echo "Start: $((182872522752/$GiB)) GiB, End: $((206833664000/$GiB)) GiB"
Start: 170 GiB, End: 192 GiB
$ echo "Start: $((182872522752/$GB)) GB, End: $((206833664000/$GB)) GB"
```

```
Start: 182 GB, End: 206 GB
```

La métrica `VolumeBytesUsed` indica cuánto almacenamiento de información en el clúster está incurriendo en cargos. Por lo tanto, es mejor minimizar este número cuando resulte práctico. Sin embargo, esta métrica no incluye parte del almacenamiento que Aurora utiliza internamente en el clúster y que no cobra. Si el clúster se acerca al límite de almacenamiento y puede quedarse sin espacio, es más útil monitorear la métrica `AuroraVolumeBytesLeftTotal` e intentar maximizar ese número. En el ejemplo siguiente se ejecuta un cálculo similar al anterior, pero para `AuroraVolumeBytesLeftTotal` en lugar de `VolumeBytesUsed`.

```
$ aws cloudwatch get-metric-statistics --metric-name "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_old_cluster_id \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS      140530528288768.0       2023-02-23T19:25:00+00:00       Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((69797067915264 / $TB)) TB remaining for this cluster"
69 TB remaining for this cluster
$ echo "$((69797067915264 / $TiB)) TiB remaining for this cluster"
63 TiB remaining for this cluster
```

Para un clúster que ejecute Aurora MySQL versión 2.09 o versiones posteriores, o Aurora PostgreSQL, el tamaño libre notificado por `VolumeBytesUsed` aumenta cuando se añaden datos y disminuye cuando se eliminan datos. El siguiente ejemplo muestra cómo. Este informe muestra el tamaño máximo y mínimo de almacenamiento de un clúster a intervalos de 15 minutos a medida que se crean y eliminan tablas con datos temporales. El informe muestra el valor máximo antes del valor mínimo. Por lo tanto, para comprender cómo cambió el uso del almacenamiento dentro del intervalo de 15 minutos, interprete los números de derecha a izquierda.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Maximum Minimum --dimensions
 Name=DBClusterIdentifier,Value=my_new_cluster_id
  --output text | sort -k 4
VolumeBytesUsed
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T20:49:00+00:00 Bytes
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T21:19:00+00:00 Bytes
```

```
DATAPOINTS 22022176768.0 14545305600.0 2020-08-05T21:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 15614263296.0 2020-08-05T23:19:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-05T23:49:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-06T00:19:00+00:00 Bytes
```

En el ejemplo siguiente, se muestra cómo con un clúster que ejecuta Aurora MySQL versión 2.09 o versiones posteriores, o Aurora PostgreSQL, el tamaño libre notificado por `AuroraVolumeBytesLeftTotal` refleja el límite de tamaño de 128 TiB.

```
$ aws cloudwatch get-metric-statistics --region us-east-1 --metric-name
 "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBClusterIdentifier,Value=pq-57 \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS 140515818864640.0 2020-08-05T20:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:26:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:56:00+00:00 Count
DATAPOINTS 140514866757632.0 2020-08-05T22:26:00+00:00 Count
DATAPOINTS 140511020580864.0 2020-08-05T22:56:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:26:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-06T00:26:00+00:00 Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((140515818864640 / $TB)) TB remaining for this cluster"
140 TB remaining for this cluster
$ echo "$((140515818864640 / $TiB)) TiB remaining for this cluster"
127 TiB remaining for this cluster
```

Escalado de instancia

Puede escalar el clúster de base de datos de Aurora como considere necesario modificando la clase de instancia de base de datos para cada instancia de base de datos del clúster de base de datos. Aurora admite varias clases de instancia de base de datos optimizadas para Aurora, dependiendo de la compatibilidad del motor de base de datos.

Motor de base de datos	Escalado de instancia
MySQL de Amazon Aurora	Consulte Escalado de las instancias de base de datos Aurora MySQL
PostgreSQL de Amazon Aurora	Consulte Escalado de las instancias de base de datos Aurora PostgreSQL

Escalado de lectura

Puede realizar el escalado de lectura de su clúster de base de datos de Aurora creando un máximo de 15 réplicas de Aurora en el clúster de base de datos. Cada réplica de Aurora devuelve los mismos datos desde el volumen de clúster con un retardo de réplica mínimo, normalmente mucho menos de 100 milisegundos una vez que la instancia principal ha escrito una actualización. A medida que el tráfico de lectura aumenta, puede crear réplicas de Aurora adicionales y conectarlas directamente para distribuir la carga de lectura del clúster de base de datos. Las réplicas de Aurora no tienen que ser de la misma clase de instancia de base de datos que la instancia principal.

Para obtener más información acerca de la adición de réplicas de Aurora a un clúster de base de datos, consulte Adición de réplicas de Aurora a un clúster de base de datos.

Administración de conexiones

El número máximo de conexiones permitidas a una instancia de base de datos Aurora viene determinado por el parámetro `max_connections` del grupo de parámetros de nivel de instancia para la instancia de base de datos. El valor predeterminado de dicho parámetro varía en función de la clase de instancia de base de datos utilizada para la instancia de base de datos y la compatibilidad del motor de base de datos.

Motor de base de datos	Valor predeterminado de max_connections (máximo de conexiones)
MySQL de Amazon Aurora	Consulte Número máximo de conexiones a una instancia de base de datos Aurora MySQL
PostgreSQL de Amazon Aurora	Consulte Número máximo de conexiones a una instancia de base de datos Aurora PostgreSQL.

> **Tip**
>
> Si sus aplicaciones abren y cierran conexiones con frecuencia, o mantienen abierto un gran número de conexiones de larga duración, le recomendamos que utilice Amazon RDS Proxy. El RDS Proxy es un proxy de base de datos totalmente administrado y de alta disponibilidad que utiliza agrupación de conexiones para compartir conexiones de base de datos de forma segura y eficiente. Para obtener más información acerca de RDS Proxy, consulteAmazon RDS Proxy para Aurora.

Administración de planes de ejecución de consultas

Si emplea una administración de planes de consultas para Aurora PostgreSQL obtiene control sobre qué planes ejecutará el optimizador. Para obtener más información, consulte Administración de planes de ejecución de consultas para Aurora PostgreSQL.

Clonación de un volumen de clúster de base de datos de Amazon Aurora

Con la clonación de Aurora, puede crear un nuevo clúster que utilice inicialmente las mismas páginas de datos que el original y sea un volumen independiente. El proceso está diseñado para ser rápido y rentable. El nuevo clúster con su volumen de datos asociado se conoce como clon. La creación de un clon es más rápido y más eficiente en el espacio que copiar físicamente los datos mediante una técnica diferente, como la restauración de una instantánea.

Temas

- Información general de la clonación de Aurora
- Limitaciones de la clonación de Aurora
- Cómo funciona la clonación de Aurora
- Creación de un clon de Amazon Aurora
- Clonación entre VPC con Amazon Aurora
- Clonación entre cuentas con AWS RAM y Amazon Aurora

Información general de la clonación de Aurora

Aurora utiliza un protocolo de copia en escritura para crear un clon. Este mecanismo utiliza un espacio adicional mínimo para crear un clon inicial. Cuando se crea el clon por primera vez, Aurora guarda una sola copia de los datos que utiliza el clúster de base de datos de Aurora de origen y el nuevo clúster de base de datos de Aurora (clonado). El almacenamiento adicional solo se asigna cuando el clúster de base de datos de Aurora de origen o el clon del clúster de base de datos de Aurora realizan cambios en los datos (en el volumen de almacenamiento de Aurora). Para obtener más información sobre el protocolo de copia en escritura, consulte Cómo funciona la clonación de Aurora.

La clonación de Aurora es especialmente útil para configurar rápidamente entornos de prueba mediante sus datos de producción, sin riesgo de corrupción de datos. Puede utilizar clones para muchos tipos de aplicaciones de corta duración, como las siguientes:

- Experimente con cambios potenciales (por ejemplo, cambios de esquema y cambios de grupo de parámetros) para evaluar todos los impactos.

- Realice operaciones intensivas de carga de trabajo, como exportar datos o ejecutar consultas analíticas en el clon.
- Cree una copia del clúster de base de datos de producción para desarrollo, pruebas u otros fines.

Puede crear más de un clon desde el mismo clúster de base de datos de Aurora. También puede crear varios clones desde otro clon.

Después de crear un clon de Aurora, puede configurar las instancias de base de datos de Aurora de forma diferente al clúster de base de datos de Aurora de origen. Por ejemplo, es posible que no necesite un clon con fines de desarrollo para cumplir con los mismos requisitos de alta disponibilidad que el clúster de base de datos Aurora de producción de origen. En este caso, puede configurar el clon con una única instancia de base de datos de Aurora en lugar de las múltiples instancias de base de datos utilizadas por el clúster de base de datos de Aurora.

Cuando crea un clon con una configuración de implementación diferente a la de origen, el clon se crea con la versión secundaria más reciente del motor de base de datos Aurora.

Cuando crea clones desde sus clústeres de base de datos de Aurora, los clones se crean en su cuenta de:AWS la misma cuenta que posee el clúster de base de datos de Aurora de origen. Sin embargo, también puede compartir clústeres y clones de base de datos de Aurora DB aprovisionados y de Aurora Serverless v2 con otras cuentas de AWS. Para obtener más información, consulte Clonación entre cuentas con AWS RAM y Amazon Aurora.

Cuando termine de utilizar el clon para realizar pruebas, desarrollo u otros fines, puede eliminarlo.

Limitaciones de la clonación de Aurora

La clonación de Aurora tiene las siguientes limitaciones:

- Puede crear tantos clones como desee, hasta el número máximo de clústeres de base de datos permitido en la Región de AWS.
- Puede crear hasta 15 clones con protocolo de copia en escritura. Después de crear 15 clones, el siguiente clon que se cree es una copia completa. El protocolo de copia completa actúa como una recuperación en un momento dado.
- No se puede crear un clon en una región de AWS distinta a la del clúster de base de datos de Aurora de origen.
- No se puede crear un clon desde un clúster de base de datos de Aurora sin la característica de consulta paralela a un clúster que utiliza consulta paralela. Para llevar datos a un clúster que utiliza

la consulta paralela, cree una instantánea del clúster original y restáurela al clúster donde está habilitada la característica de consulta paralela.

- No se puede crear un clon desde un clúster de base de datos de Aurora que no tiene instancias de base de datos. Solo se pueden clonar clústeres de base de datos de Aurora que tengan al menos una instancia de base de datos.
- Se puede crear un clon en una Virtual Private Cloud (VPC) diferente de la del clúster de base de datos de Aurora. Sin embargo, las subredes de esas VPC deben estar asignadas al mismo conjunto de zonas de disponibilidad.
- Puede crear un clon aprovisionado de Aurora desde un clúster de base de datos de Aurora aprovisionado.
- Los clústeres con instancias de Aurora Serverless v2 siguen las mismas reglas que los clústeres aprovisionados.
- En:Aurora Serverless v1
  - Puede crear un clon aprovisionado desde un clúster de base de datos de Aurora Serverless v1.
  - Puede crear un clon de Aurora Serverless v1 desde un clúster de base de datos aprovisionado o de Aurora Serverless v1.
  - No se puede crear un clon de Aurora Serverless v1 a partir de un clúster de base de datos de Aurora aprovisionado no cifrado.
  - La clonación entre cuentas actualmente no admite la clonación de clústeres de base de datos de Aurora Serverless v1. Para obtener más información, consulte Limitaciones de la clonación entre cuentas.
  - Un clúster de base de datos de Aurora Serverless v1 clonado tiene el mismo comportamiento y limitaciones que cualquier clúster de base de datos de Aurora Serverless v1. Para obtener más información, consulte Uso de Amazon Aurora Serverless v1.
  - Los clústeres de base de datos de Aurora Serverless v1 están siempre cifrados. Cuando clona un clúster de base de datos de de Aurora Serverless v1 en un clúster de base de datos de Aurora aprovisionado, el clúster de de base de datos de Aurora aprovisionado está cifrado. Puede elegir la clave de cifrado, pero no puede desactivar el cifrado. Para crear un clon aprovisionado de base de datos de Aurora a un Aurora Serverless v1, debe hacerlo a partir de un clúster de base de datos de Aurora cifrado y aprovisionado.

Cómo funciona la clonación de Aurora

La clonación de Aurora funciona en la capa de almacenamiento de un clúster de base de datos de Aurora. Utiliza un protocolo copy-on-write que es rápido y eficiente en el espacio en términos de los medios permanentes subyacentes que soportan el volumen de almacenamiento de Aurora. Puede obtener más información sobre volúmenes de clúster de Aurora en Información general del almacenamiento de Amazon Aurora.

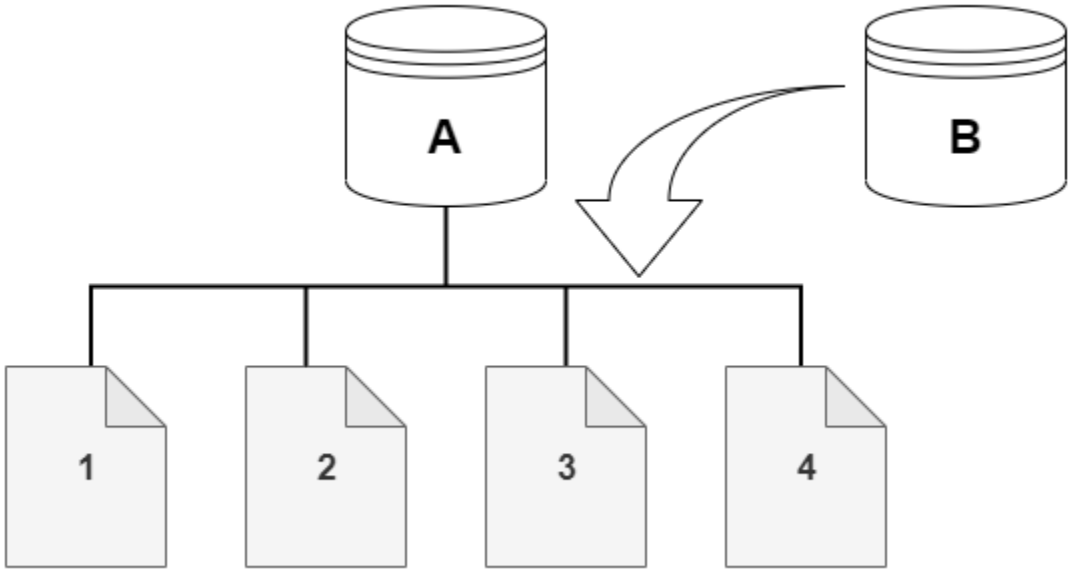
Temas

- Descripción del protocolo de copia en escritura
- Eliminación de un volumen del clúster de origen

Descripción del protocolo de copia en escritura

Un clúster de base de datos de Aurora almacena datos en páginas en el volumen de almacenamiento de Aurora subyacente.

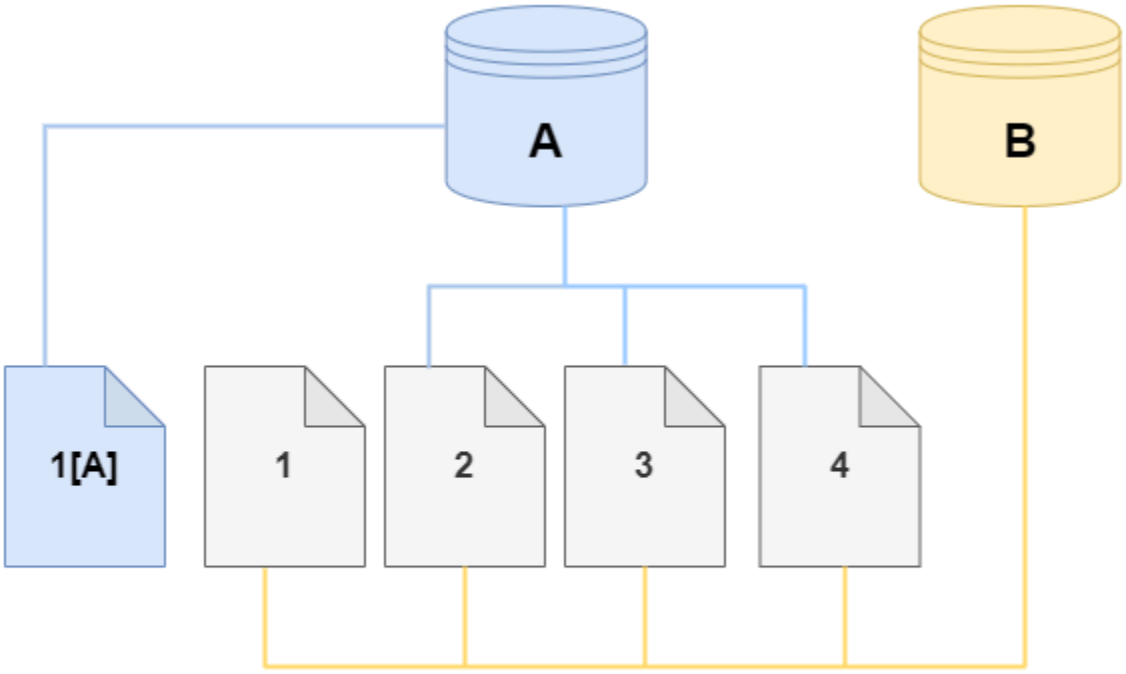
Por ejemplo, en el siguiente diagrama puede encontrar un clúster de base de datos de Aurora (A) que tiene cuatro páginas de datos, 1, 2, 3 y 4. Imagine que un clon, B, se crea desde del clúster de base de datos de Aurora. Cuando se crea el clon, no se copian datos. Más bien, el clon apunta al mismo conjunto de páginas que el clúster de base de datos de Aurora de origen.



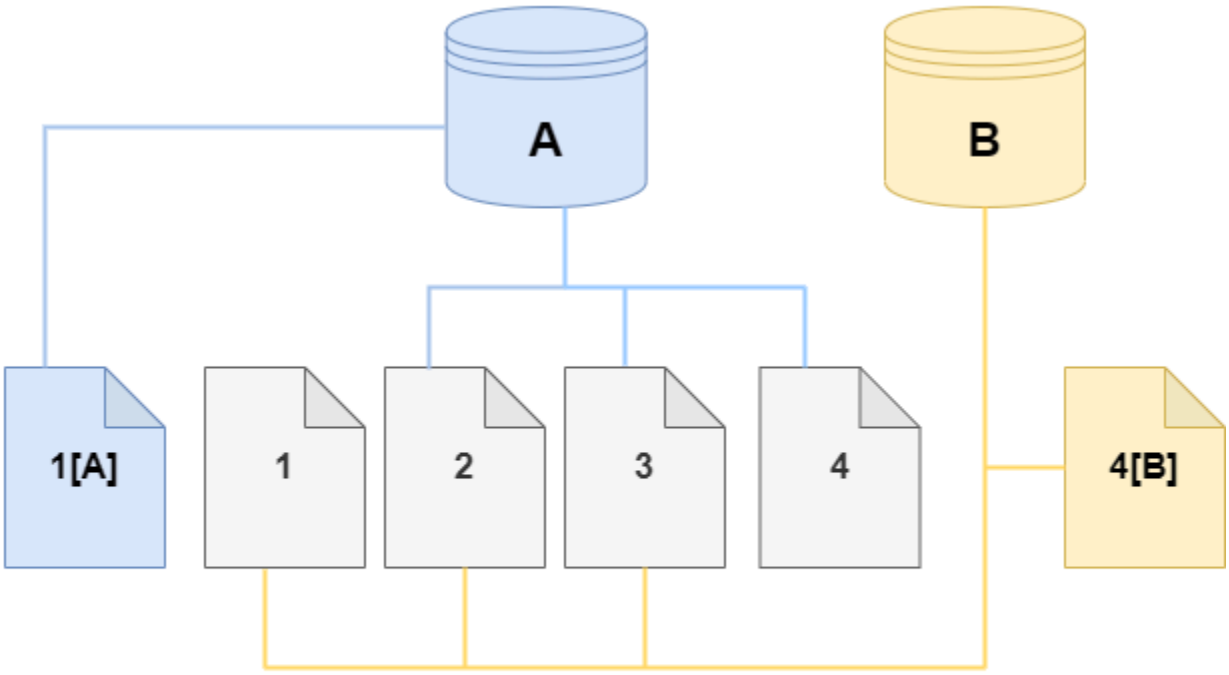
Cuando se crea el clon, generalmente no se necesita almacenamiento adicional. El protocolo de copia en escritura utiliza el mismo segmento en los medios de almacenamiento físico que el

segmento de origen. Solo se requiere almacenamiento adicional si la capacidad del segmento de origen no es suficiente para todo el segmento de clones. Si ese es el caso, el segmento de origen se copia en otro dispositivo físico.

En los diagramas siguientes, puede encontrar un ejemplo del protocolo de copia en escritura en acción utilizando el mismo clúster A y su clon B, como se muestra anteriormente. Supongamos que realiza un cambio en su clúster de base de datos de Aurora (A) que da lugar a un cambio en los datos almacenados en la página 1. En lugar de escribir en la página original 1, Aurora crea una nueva página 1 [A]. El volumen del clúster de base de datos de Aurora para el clúster (A) ahora apunta a la página 1 [A], 2, 3 y 4, mientras que el clon (B) sigue haciendo referencia a las páginas originales.



En el clon, se realiza un cambio en la página 4 del volumen de almacenamiento. En lugar de escribir en la página original 4, Aurora crea una nueva página 4 [B]. El clon ahora apunta a las páginas 1, 2, 3 y a la página 4 [B], mientras que el clúster (A) continúa apuntando a 1 [A], 2, 3 y 4.



A medida que se producen más cambios a lo largo del tiempo en el clon y el volumen del clúster de base de datos de Aurora de origen, necesitará cada vez más almacenamiento para capturar y almacenar los cambios.

Eliminación de un volumen del clúster de origen

Inicialmente, el volumen de clon comparte las mismas páginas de datos que el volumen original a partir del cual se crea el clon. Mientras exista el volumen original, el volumen del clon solo se considera propietario de las páginas que el clon ha creado o modificado. Por lo tanto, la métrica `VolumeBytesUsed` del volumen del clon comienza siendo pequeña y solo aumenta a medida que los datos divergen entre el clúster original y el clon. En el caso de páginas idénticas entre el volumen de origen y el clon, los cargos de almacenamiento se aplican únicamente al clúster original. Para obtener más información acerca de la métrica `VolumeBytesUsed`, consulte Métricas de nivel de clúster para Amazon Aurora.

Cuando elimina un volumen de clúster de origen que tiene uno o más clones asociados, los datos en los volúmenes del clúster de los clones no cambian. Aurora conserva las páginas que antes

pertenecían al volumen del clúster de origen. Aurora redistribuye la facturación del almacenamiento de las páginas que pertenecían al clúster eliminado. Por ejemplo, supongamos que un clúster original tenía dos clones y, después, se eliminó el clúster original. La mitad de las páginas de datos que pertenecían al clúster original ahora pertenecerían a un clon. La otra mitad de las páginas pertenecería al otro clon.

Si elimina el clúster original, a medida que crea o elimina más clones, Aurora sigue redistribuyendo la propiedad de las páginas de datos entre todos los clones que comparten las mismas páginas. Por lo tanto, es posible que observe que el valor de la métrica `VolumeBytesUsed` cambia para el volumen del clúster de un clon. El valor de la métrica puede disminuir a medida que se crean más clones y la propiedad de la página se distribuye entre más clústeres. El valor de la métrica también puede aumentar a medida que se eliminan los clones y se asigna la propiedad de la página a una cantidad menor de clústeres. Para obtener información sobre cómo afectan las operaciones de escritura a las páginas de datos de los volúmenes de clones, consulte Descripción del protocolo de copia en escritura.

Cuando el clúster original y los clones pertenecen a la misma cuenta de AWS, todos los cargos de almacenamiento de esos clústeres se aplican a esa misma cuenta de AWS. Si algunos de los clústeres son clones entre cuentas, eliminar el clúster original puede conllevar cargos de almacenamiento adicionales a las cuentas de AWS que poseen los clones entre cuentas.

Por ejemplo, supongamos que un volumen de clúster tiene 1000 páginas de datos usados antes de crear cualquier clon. Al clonar ese clúster, inicialmente el volumen del clon no tiene ninguna página utilizada. Si el clon modifica 100 páginas de datos, solo esas 100 páginas se almacenan en el volumen del clon y se marcan como usadas. Las otras 900 páginas sin cambios del volumen principal se comparten entre ambos clústeres. En este caso, el clúster principal tiene cargos de almacenamiento para 1000 páginas y el volumen del clon para 100 páginas.

Si elimina el volumen de origen, los cargos de almacenamiento del clon incluyen las 100 páginas modificadas, más las 900 páginas compartidas del volumen original, lo que da un total de 1000 páginas.

Creación de un clon de Amazon Aurora

Puede crear un clon en la misma cuenta de AWS como clúster de base de datos de Aurora de origen. Para ello, puede utilizar la Consola de administración de AWS o la AWS CLI, y los procedimientos siguientes.

Para permitir que otra cuenta de AWS cree un clon o comparta un clon con otra cuenta de AWS, utilice los procedimientos en Clonación entre cuentas con AWS RAM y Amazon Aurora.

Consola

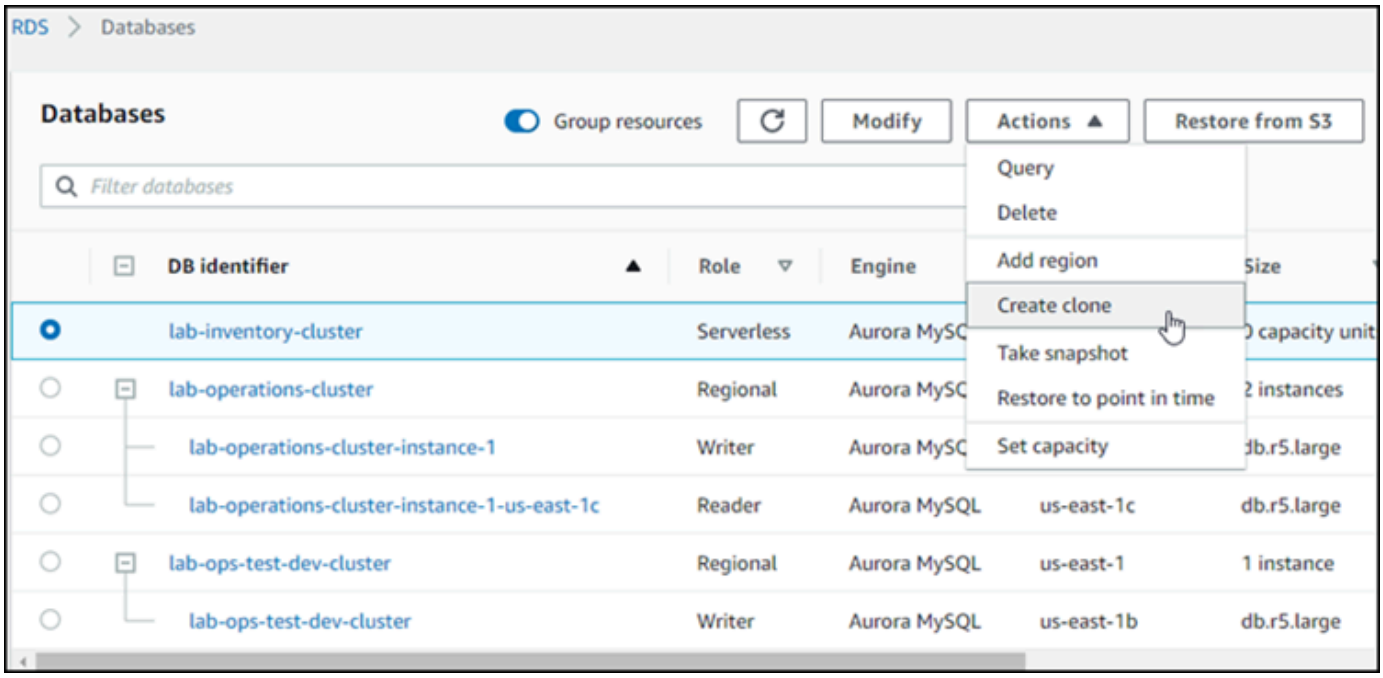
El siguiente procedimiento describe cómo clonar un clúster de base de datos de Aurora mediante la Consola de administración de AWS.

Al crear un clon con la Consola de administración de AWS resulta en un clúster de base de datos de Aurora con una instancia de base de datos de Aurora.

Estas instrucciones se aplican a los clústeres de base de datos que pertenecen a la misma AWS cuenta que crea la clonación. Si el clúster de base de datos es propiedad de otra AWS cuenta, consulte Clonación entre cuentas con AWS RAM y Amazon Aurora en su lugar.

Para crear un clon de un clúster de base de datos propiedad de la AWS cuenta mediante el comando Consola de administración de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija su clúster de base de datos de Aurora de la lista y para Actions (Acciones), elija Create clone (Crear clon).



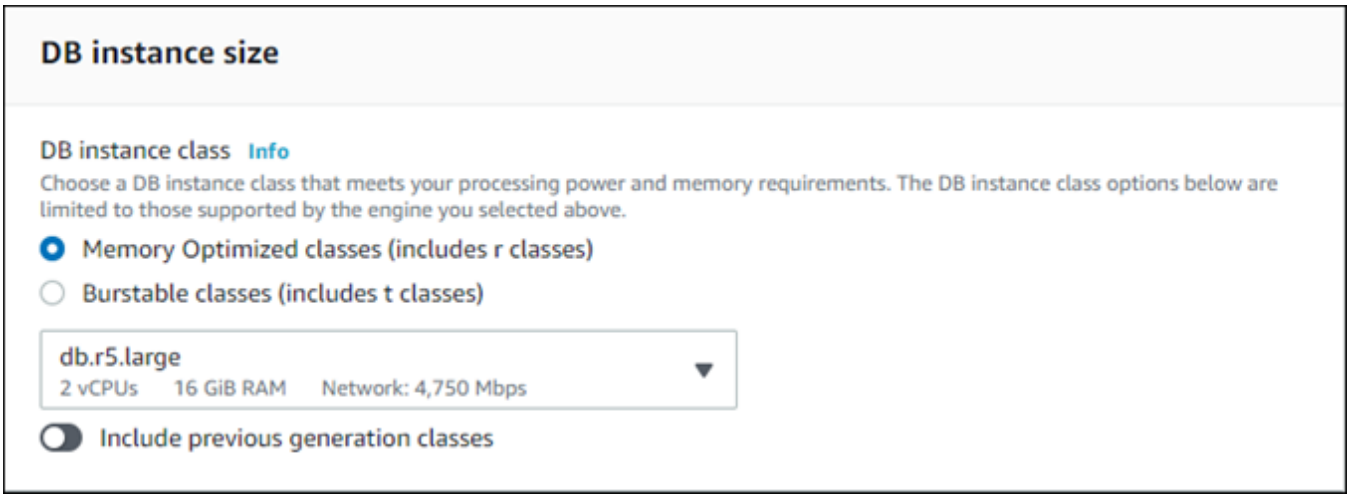
Se abre la página Crear clon donde puede configurar Configuración, Conectividad y otras opciones para el clon del clúster de base de datos de Aurora.

4. Para el identificador de instancia de base de datos, indique el nombre que desea asignar a su clúster de base de datos de Aurora clonado.
5. Para los clústeres de bases de datos de Aurora Serverless v1, elija Aprovisionado o Sin servidor como Tipo de capacidad.

   Puede elegir Serverless (Sin servidor) solo si el clúster de base de datos de Aurora de origen es una clúster de base de datos de Aurora Serverless v1 o es un clúster de base de datos de Aurora aprovisionado que está cifrado.
6. Para los clústeres de bases de datos de Aurora Serverless v2 o aprovisionados, elija Aurora I/O-Optimized o Aurora Standard para Configuración de almacenamiento del clúster.

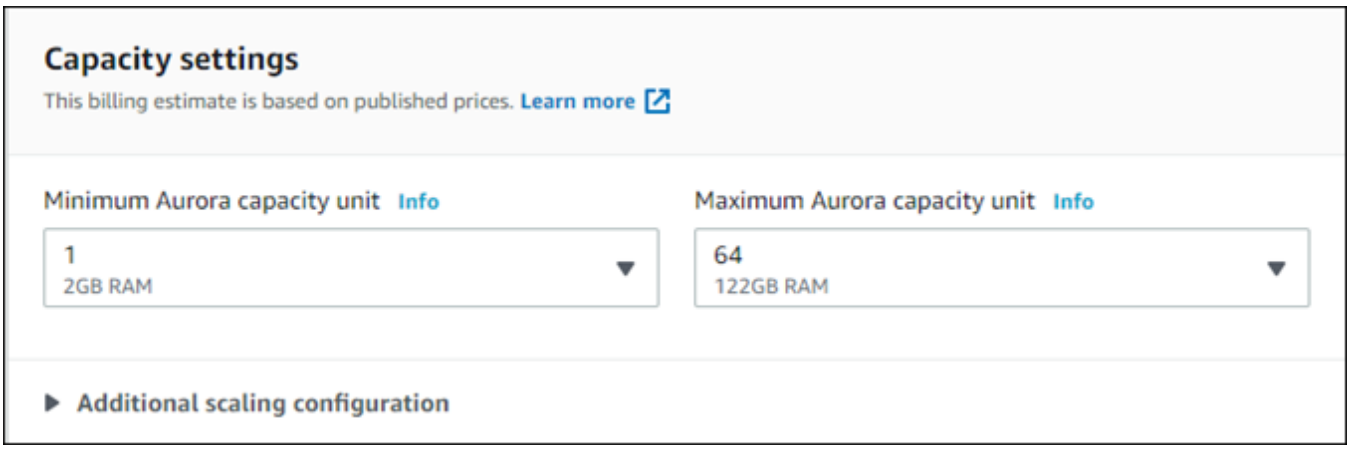
   Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.
7. Elija el tamaño de la instancia de base de datos o la capacidad del clúster de base de datos:

   - Para un clon aprovisionado, elija una Clase de instancia de base de datos.



     Puede aceptar la configuración proporcionada o puede usar una clase de instancia de base de datos diferente para su clon.

   - Para un clon de Aurora Serverless v1 o Aurora Serverless v2, elija la Configuración de capacidad.



Puede aceptar la configuración proporcionada o cambiarla para su clon.

8. Seleccione el resto de ajustes según sea necesario para su clon. Para obtener más información sobre la configuración del clúster y de la instancia de base de datos de Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora.
9. Elija Crear clon.

Cuando se crea el clon, aparece junto con los otros clústeres de base de datos de Aurora en la sección Databases (Bases de datos) de la consola y muestra su estado actual. Su clon está listo para utilizar cuando su estado es Available (Disponible).

AWS CLI

El uso de la AWS CLI para clonar el clúster de base de datos de Aurora implica pasos separados para crear el clúster de clones y agregarle una o más instancias de base de datos.

El comando `restore-db-cluster-to-point-in-time` de la AWS CLI que utiliza produce un clúster de base de datos de Aurora con los mismos datos de almacenamiento que el clúster original, pero ninguna instancia de base de datos de Aurora. Se crean instancias de base de datos por separado después de que el clon esté disponible. Puede elegir el número de instancias de base de datos y sus clases de instancias para dar al clon una capacidad de procesamiento mayor o menor que la del clúster original. Los pasos del proceso son los siguientes:

1. Cree el clon mediante el comando restore-db-clúster-to-point-in-time de la CLI.
2. Cree la instancia de base de datos de escritor para el clon con el comando de la CLI create-db-instance.
3. (Opcional) Ejecute comandos de la CLI create-db-instance adicionales para agregar una o más instancias de lector al clúster de clones. El uso de instancias de lector ayuda a mejorar los

aspectos de alta disponibilidad y escalabilidad de lectura del clon. Puede omitir este paso si tiene pensado utilizar el clon para el desarrollo y las pruebas.

Temas

- Creación del clon
- Comprobación del estado y obtención de detalles del clon
- Crear la instancia de base de datos de Aurora para su clon
- Parámetros para utilizar durante la clonación

Creación del clon

Utilice el comando de la CLI `restore-db-cluster-to-point-in-time` para crear el clúster de clones inicial.

Creación de un clon desde un clúster de base de datos de Aurora de origen

- Utilice el comando CLI `restore-db-cluster-to-point-in-time`. Especifique los valores de los siguientes parámetros. En este caso típico, el clon utiliza el mismo modo de motor que el clúster original, ya sea aprovisionado o Aurora Serverless v1.
  - `--db-cluster-identifier`: elija un nombre significativo para su clon. Se asigna un nombre al clon cuando se utiliza el comando restore-db-clúster-to-point-in-time de la CLI. A continuación, pase el nombre del clon en el comando create-db-instance de la CLI.
  - `--restore-type`: utilice `copy-on-write` para crear un clon del clúster de base de datos de origen. Sin este parámetro, `restore-db-cluster-to-point-in-time` restaura el clúster de base de datos de Aurora en lugar de crear un clon.
  - `--source-db-cluster-identifier`: utilice el nombre del clúster de base de datos de Aurora de origen que desea clonar.
  - `--use-latest-restorable-time`: este valor apunta a los datos de volumen restaurables más recientes para el clúster de base de datos de origen. Úselo para crear clones.

El siguiente ejemplo crea un clon del clúster de denominado `my-clone` desde un clúster denominado `my-source-cluster`.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

El comando devuelve el objeto JSON que contiene detalles del clon. Compruebe que su clúster de base de datos clonado está disponible antes de intentar crear la instancia de base de datos para su clon. Para obtener más información, consulte Comprobación del estado y obtención de detalles del clon.

Por ejemplo, supongamos que tiene un clúster llamado `tpch100g` que desea clonar. En el siguiente ejemplo de Linux, se crea un clúster clonado denominado `tpch100g-clone`, una instancia de escritor de Aurora Serverless v2 denominada `tpch100g-clone-instance` y una instancia de lector aprovisionada denominada `tpch100g-clone-instance-2` para el nuevo clúster.

No es necesario proporcionar algunos parámetros, como `--master-username` y `--master-user-password`. Aurora determina automáticamente los parámetros del clúster original. Es necesario especificar el motor de base de datos que se va a utilizar. Por lo tanto, el ejemplo prueba el nuevo clúster para determinar el valor correcto a utilizar para el parámetro `--engine`.

En este ejemplo también se incluye la opción `--serverless-v2-scaling-configuration` al crear el clúster de clones. De esta forma, puede agregar instancias de Aurora Serverless v2 al clon aunque el clúster original no usara Aurora Serverless v2.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier tpch100g \
  --db-cluster-identifier tpch100g-clone \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
  --restore-type copy-on-write \
  --use-latest-restorable-time

$ aws rds describe-db-clusters \
```

```
  --db-cluster-identifier tpch100g-clone \
    --query '*[].[Engine]' \
    --output text
aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.serverless \
  --engine aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance-2 \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.r6g.2xlarge \
  --engine aurora-mysql
```

Para crear un clon en un modo de motor distinto al del clúster de base de datos de Aurora de origen.

- Este procedimiento solo se aplica a las versiones de motor anteriores que admitan Aurora Serverless v1. Supongamos que tiene un clúster de Aurora Serverless v1 y quiere crear un clon que sea un clúster aprovisionado. En ese caso, utilice el comando de la CLI `restore-db-cluster-to-point-in-time` y especifique valores de parámetros similares a los del ejemplo anterior, además de estos parámetros adicionales:

  - `--engine-mode`: utilice este parámetro solo para crear clones que sean de un modo de motor diferente al del clúster de base de datos de Aurora de origen. Este parámetro solo se aplica a las versiones de motor anteriores que admitan Aurora Serverless v1. Elija el valor con el que se va a pasar `--engine-mode` de la siguiente manera:

    - Utilice `--engine-mode provisioned` para crear un clon de clúster de base de datos de Aurora aprovisionado desde un clúster de base de datos de Aurora Serverless.

      > **Note**
      >
      > Si tiene pensado utilizar Aurora Serverless v2 con un clúster que se clonó desde Aurora Serverless v1, siga especificando el modo de motor del clon como `provisioned`. A continuación, debe realizar pasos adicionales de actualización y migración.

- Utilice `--engine-mode serverless` para crear un clon de Aurora Serverless v1 desde un clúster de base de datos de Aurora aprovisionado. Cuando se especifica el modo de motor `serverless`, también puede elegir la `--scaling-configuration`.

- `--scaling-configuration`: (opcional) se utiliza con `--engine-mode serverless` para configurar la capacidad mínima y máxima de un clon de Aurora Serverless v1. Si no utiliza este parámetro, Aurora crea un clon de Aurora Serverless v1 con los valores de capacidad de Aurora Serverless v1 predeterminados del motor de base de datos.

El siguiente ejemplo crea un clon aprovisionado denominado `my-clone` desde un clúster de base de datos de Aurora Serverless v1 denominado `my-source-cluster`.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --engine-mode provisioned \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --engine-mode provisioned ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Estos comandos devuelven el objeto JSON que contiene detalles del clon que necesita para crear la instancia de base de datos. No puede hacer eso hasta que el estado del clon (el clúster vacío de base de datos de Aurora) tenga el estado Available (Disponible).

> **Note**
>
> El comando restore-db-clúster-to-point-in-time de la CLI de AWS solo restaura el clúster de la base de datos, no las instancias de base de datos dicho clúster. Ejecute el comando create-db-instance para crear instancias de base de datos para el clúster de base de datos restaurado. Con ese comando, especifique el identificador del clúster de base de datos

restaurado como parámetro `--db-cluster-identifier`. Solo puede crear instancias de base de datos después de que se haya completado el comando `restore-db-cluster-to-point-in-time` y de que el clúster de base de datos esté disponible.
Suponga que comienza con un clúster de Aurora Serverless v1 y tiene la intención de migrarlo a un clúster de Aurora Serverless v2. Como paso inicial de la migración, debe crear un clon aprovisionado del clúster de Aurora Serverless v1. Para ver el procedimiento completo, incluidas las actualizaciones de la versión necesarias, consulte Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.

Comprobación del estado y obtención de detalles del clon

Puede utilizar el siguiente comando para verificar el estado del clúster de clones recién creado.

```
$ aws rds describe-db-clusters --db-cluster-identifier my-clone --query '*[].[Status]'
 --output text
```

O puede obtener el estado y los otros valores que necesita paracrear la instancia de base de datos para su clon mediante el uso de la siguiente consulta de la AWS CLI.

Para Linux, macOS o Unix:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}'
```

Para Windows:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}"
```

Esta consulta devuelve un resultado similar al siguiente:

```
[
  {
        "Status": "available",
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.04.1",
```

```
        "EngineMode": "provisioned"
    }
]
```

Crear la instancia de base de datos de Aurora para su clon

Use el comando create-db-instance de la CLI para crear la instancia de base de datos para su clon de Aurora Serverless v2 o aprovisionado. No se crea una instancia de base de datos para un clon de Aurora Serverless v1.

La instancia de base de datos hereda las propiedades `--master-username` y `--master-user-password` del clúster de base de datos de origen.

El siguiente ejemplo crea una instancia de base de datos para un clon aprovisionado.

Para Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.r6g.2xlarge \
    --engine aurora-mysql
```

Para Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.r6g.2xlarge ^
    --engine aurora-mysql
```

En el siguiente ejemplo, se crea una instancia de base de datos de Aurora Serverless v2 para un clon que utiliza una versión del motor que admite Aurora Serverless v2.

Para Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.serverless \
  --engine aurora-postgresql
```

Para Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.serverless ^
    --engine aurora-mysql
```

Parámetros para utilizar durante la clonación

En la siguiente tabla se resumen los diversos parámetros utilizados con `restore-db-cluster-to-point-in-time` para clonar clústeres de base de datos de Aurora.

Parámetro	Descripción
`--source-db-cluster-identifier`	Utilice el nombre del clúster de base de datos de Aurora de origen que desea clonar.
`--db-cluster-identifier`	Elija un nombre significativo para su clon al crearlo con el comando `restore-db-cluster-to-point-in-time` . A continuación, pase este nombre al comando `create-db-instance` .
`--restore-type`	Especifique `copy-on-write` como el `--restore-type` para crear un clon del clúster de base de datos de origen en lugar de restaurar el clúster de base de datos de Aurora de origen.
`--use-latest-restorable-time`	Este valor apunta a los datos de volumen restaurables más recientes para el clúster de base de datos de origen. Úselo para crear clones.
`--serverless-v2-scaling-configuration`	(Versiones más recientes que admiten Aurora Serverless v2) Utilice este parámetro para configurar la capacidad mínima y máxima de un clon de Aurora Serverless v2. Si no especifica este parámetro, no podrá crear ninguna instancia de Aurora Serverless v2 en el clúster de clones hasta que modifique el clúster para agregar este atributo.
`--engine-mode`	(Versiones anteriores que admiten solo Aurora Serverless v1) Utilice este parámetro para crear clones que sean de un tipo diferente al del

Parámetro	Descripción
	clúster de base de datos de Aurora de origen, con uno de los siguientes valores: • Utilice `provisioned` para crear un clon aprovisionado desde un clúster de base de datos de Aurora Serverless v1. • Utilice `serverless` para crear un clon de Aurora Serverless v1 desde un clúster de base de datos aprovisionado o de Aurora Serverless v2. Cuando se especifica el modo de motor `serverless`, también puede elegir la `--scaling-configuration`.
`--scaling-configuration`	(Versiones anteriores que admiten solo Aurora Serverless v1) Utilice este parámetro para configurar la capacidad mínima y máxima de un clon de Aurora Serverless v1. Si no especifica este parámetro, Aurora crea el clon con los valores de capacidad predeterminados del motor de base de datos.

Para obtener información sobre la clonación entre VPC y entre cuentas, consulte las siguientes secciones.

Temas

- Clonación entre VPC con Amazon Aurora
- Clonación entre cuentas con AWS RAM y Amazon Aurora

Clonación entre VPC con Amazon Aurora

Imagine que quiere imponer diferentes controles de acceso a la red en el clúster original y en el clon. Por ejemplo, podría utilizar la clonación para hacer una copia de un clúster Aurora de producción en una VPC diferente para el desarrollo y las pruebas. O bien podría crear un clon como parte de una migración de subredes públicas a subredes privadas para mejorar la seguridad de su base de datos.

En las siguientes secciones, se muestra cómo configurar la red del clon para que el clúster original y el clon puedan acceder a los mismos nodos de almacenamiento de Aurora, incluso desde subredes

o VPC diferentes. Verificar los recursos de la red con antelación puede evitar errores durante la clonación que podrían ser difíciles de diagnosticar.

Si no conoce la forma en que Aurora interactúa con las VPC, las subredes y los grupos de subredes de base de datos, consulte primero VPC de Amazon y Amazon Aurora. Puede consultar los tutoriales de esa sección para crear este tipo de recursos en la consola de AWS y comprender cómo encajan entre sí.

Dado que los pasos implican cambiar entre los servicios de RDS y EC2, los ejemplos utilizan comandos de AWS CLI para ayudarle a entender cómo automatizar dichas operaciones y guardar el resultado.

Temas

- Antes de empezar
- Recopilación de información sobre el entorno de red
- Creación de recursos de red para el clon
- Creación de un clon de Aurora con una nueva configuración de red
- Traslado de un clúster de subredes públicas a subredes privadas
- Ejemplo de extremo a extremo de creación de un clon entre VPC

Antes de empezar

Antes de empezar a configurar un clon entre VPC, asegúrese de tener preparados los siguientes recursos:

- Un clúster de base de datos Aurora para usarlo como origen de clonación. Si es la primera vez que crea un clúster de base de datos Aurora, consulte los tutoriales de Introducción a Amazon Aurora para configurar un clúster mediante el motor de base de datos MySQL o PostgreSQL.
- Una segunda VPC, si tiene pensado crear un clon entre VPC. Si no tiene una VPC que usar para el clon, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4) o Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble).

Recopilación de información sobre el entorno de red

Con la clonación entre VPC, el entorno de red puede diferir en gran medida entre el clúster original y su clon. Antes de crear el clon, recopile y registre información sobre la VPC, las subredes, el

grupo de subredes de base de datos y las AZ utilizadas en el clúster original. De esta forma, puede minimizar las probabilidades de que surjan problemas. Si se produce un problema en la red, no tendrá que interrumpir ninguna actividad de solución de problemas para buscar información de diagnóstico. Las siguientes secciones muestran ejemplos de CLI para recopilar este tipo de información. Puede guardar los detalles en el formato que le resulte útil para consultar al crear el clon y solucionar cualquier problema.

- Paso 1: comprobación de las zonas de disponibilidad del clúster original
- Paso 2: comprobación del grupo de subredes de base de datos del clúster original
- Paso 3: comprobación de las subredes del clúster original
- Paso 4: comprobación de las zonas de disponibilidad de las instancias de base de datos en el clúster original
- Paso 5: comprobación de las VPC que puede utilizar para el clon

Paso 1: comprobación de las zonas de disponibilidad del clúster original

Antes de crear el clon, compruebe qué zonas de disponibilidad utiliza el clúster original para su almacenamiento. Como se explica en Almacenamiento de Amazon Aurora, el almacenamiento de cada clúster Aurora está asociado exactamente a tres zonas de disponibilidad (AZ). Como Clústeres de base de datos de Amazon Aurora aprovecha la separación entre la computación y el almacenamiento, esta regla es válida independientemente de cuántas instancias haya en el clúster.

Por ejemplo, ejecute un comando de la CLI como el siguiente y sustituya el nombre de su propio clúster por *my_cluster*. El siguiente ejemplo genera una lista ordenada alfabéticamente por el nombre de la AZ.

```
aws rds describe-db-clusters \
  --db-cluster-identifier my_cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' \
  --output text
```

En el siguiente ejemplo, se muestra el resultado del comando `describe-db-clusters` anterior. Demuestra que el almacenamiento del clúster Aurora siempre usa tres AZ.

```
us-east-1c
us-east-1d
us-east-1e
```

Para crear un clon en un entorno de red que no cuente con todos los recursos necesarios para conectarse a estas AZ, debe crear subredes asociadas al menos a dos de esas AZ y, a continuación, crear un grupo de subredes de base de datos que contenga esas dos o tres subredes. Los siguientes ejemplos muestran cómo hacerlo.

Paso 2: comprobación del grupo de subredes de base de datos del clúster original

Si desea utilizar el mismo número de subredes para el clon que en el clúster original, puede obtener el número de subredes del grupo de subredes de base de datos del clúster original. Un grupo de subredes de base de datos Aurora contiene al menos dos subredes, cada una asociada a una AZ diferente. Anote las AZ a las que están asociadas las subredes.

En el siguiente ejemplo, se muestra cómo encontrar el grupo de subredes de base de datos del clúster original y, a continuación, retroceder hasta las AZ correspondientes. Sustituya el nombre de su clúster por *`my_cluster`* en el primer comando. Sustituya el nombre del grupo de subredes de base de datos por *`my_subnet`* en el segundo comando.

```
aws rds describe-db-clusters --db-cluster-identifier my_cluster \
  --query '*[].DBSubnetGroup' --output text

aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetAvailabilityZone.Name]' --output text
```

El resultado de un ejemplo podría ser similar al siguiente para un clúster con un grupo de subredes de base de datos que contenga dos subredes. En este caso, `two-subnets` es un nombre que se especificó al crear el grupo de subredes de base de datos.

```
two-subnets

us-east-1d
us-east-1c
```

Para un clúster con un grupo de subredes de base de datos que contenga tres subredes, el resultado podría ser similar al siguiente.

```
three-subnets

us-east-1f
us-east-1d
```

```
us-east-1c
```

Paso 3: comprobación de las subredes del clúster original

Si necesita más detalles sobre las subredes del clúster original, ejecute comandos de AWS CLI similares a los siguientes. Puede examinar los atributos de las subredes, como los rangos de direcciones IP, el propietario, etc. De esta forma, puede determinar si desea utilizar subredes diferentes en la misma VPC o crear subredes con características similares en una VPC diferente.

Busque los ID de subred de todas las subredes que están disponibles en su VPC.

```
aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc \
  --query '*[].[SubnetId]' --output text
```

Busque las subredes exactas que se utilizan en el grupo de subredes de base de datos.

```
aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetIdentifier]' --output text
```

A continuación, especifique las subredes que quiere investigar en una lista, como en el siguiente comando. Sustituya los nombres de las subredes por *my_subnet_1*, etc.

```
aws ec2 describe-subnets \
  --subnet-ids '["my_subnet_1","my_subnet2","my_subnet3"]'
```

El siguiente ejemplo muestra el resultado parcial de un comando `describe-subnets` de este tipo. El resultado muestra algunos de los atributos importantes que puede ver en cada subred, como su AZ y la VPC de la que forma parte.

```
{
    'Subnets': [
        {
            'AvailabilityZone': 'us-east-1d',
            'AvailableIpAddressCount': 54,
            'CidrBlock': '10.0.0.64/26',
            'State': 'available',
            'SubnetId': 'subnet-000a0bca00e0b0000',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
        },
```

```
        {
            'AvailabilityZone': 'us-east-1c',
            'AvailableIpAddressCount': 55,
            'CidrBlock': '10.0.0.0/26',
            'State': 'available',
            'SubnetId': 'subnet-4b4dbfe4d4a4fd4c4',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
```

Paso 4: comprobación de las zonas de disponibilidad de las instancias de base de datos en el clúster original

Puede utilizar este procedimiento para comprender las AZ utilizadas para las instancias de base de datos del clúster original. De esta forma, puede configurar exactamente las mismas AZ para las instancias de base de datos del clon. También puede utilizar más o menos instancias de base de datos en el clon, en función de si el clon se utiliza para la producción, el desarrollo y las pruebas, etc.

Para cada instancia del clúster original, ejecute un comando como el siguiente. Asegúrese de que la instancia haya terminado de crearse y de que esté en el estado `Available` primero. Sustituya el identificador de instancia por *`my_instance`*.

```
aws rds describe-db-instances --db-instance-identifier my_instance \
  --query '*[].AvailabilityZone' --output text
```

El siguiente ejemplo muestra el resultado de ejecutar el comando `describe-db-instances` anterior. El clúster Aurora tiene cuatro instancias de base de datos. Por lo tanto, ejecutamos el comando cuatro veces y lo sustituimos por un identificador de instancia de base de datos diferente cada vez. El resultado muestra cómo se distribuyen esas instancias de base de datos en un máximo de tres AZ.

```
us-east-1a
us-east-1c
us-east-1d
us-east-1a
```

Después de crear el clon y de agregarle instancias de base de datos, puede especificar estos mismos nombres de las AZ en los comandos `create-db-instance`. Puede hacerlo para configurar las instancias de base de datos en el nuevo clúster configurado exactamente para las mismas AZ que en el clúster original.

Paso 5: comprobación de las VPC que puede utilizar para el clon

Si tiene pensado crear el clon en una VPC diferente a la original, puede obtener una lista de los ID de VPC disponibles para su cuenta. También puede realizar este paso si necesita crear subredes adicionales en la misma VPC que el clúster original. Al ejecutar el comando para crear una subred, se especifica el ID de VPC como parámetro.

Para enumerar todas las VPC de la cuenta, ejecute el siguiente comando de CLI:

```
aws ec2 describe-vpcs --query '*[].[VpcId]' --output text
```

En el siguiente ejemplo, se muestra el resultado de muestra del comando `describe-vpcs` anterior. El resultado demuestra que hay cuatro VPC en la cuenta de AWS actual que se pueden usar como origen o destino para la clonación entre VPC.

```
vpc-fd111111
vpc-2222e2cd2a222f22e
vpc-33333333a33333d33
vpc-4ae4d4de4a4444dad
```

Puede usar la misma VPC como destino para el clon o una VPC diferente. Si el clúster original y el clon están en la misma VPC, puede reutilizar el mismo grupo de subredes de base de datos para el clon. También puede crear un grupo de subredes de base de datos diferente. Por ejemplo, el nuevo grupo de subredes de base de datos podría usar subredes privadas, mientras que el grupo de subredes de base de datos del clúster original podría usar subredes públicas. Si crea el clon en una VPC diferente, asegúrese de que haya suficientes subredes en la nueva VPC y de que las subredes estén asociadas a las AZ correctas del clúster original.

Creación de recursos de red para el clon

Si al recopilar la información de la red ha descubierto que se necesitan recursos de red adicionales para el clon, puede crear esos recursos antes de intentar configurar el clon. Por ejemplo, podría necesitar crear más subredes, subredes asociadas a zonas de disponibilidad concretas o un nuevo grupo de subredes de base de datos.

- Paso 1: creación de las subredes para el clon
- Paso 2: creación del grupo de subredes de base de datos para el clon

Paso 1: creación de las subredes para el clon

Si necesita crear nuevas subredes para el clon, ejecute un comando similar al siguiente. Es posible que tenga que hacerlo al crear el clon en una VPC diferente o al realizar algún otro cambio en la red, como usar subredes privadas en lugar de subredes públicas.

AWS genera automáticamente el ID de la subred. Sustituya el nombre de la VPC del clon por *my_vpc*. Elija el rango de direcciones para la opción `--cidr-block` para permitir al menos 16 direcciones IP en el rango. Puede incluir cualquier otra propiedad que desee especificar. Ejecute el comando `aws ec2 create-subnet help` para ver todas las opciones.

```
aws ec2 create-subnet --vpc-id my_vpc \
  --availability-zone AZ_name --cidr-block IP_range
```

El siguiente ejemplo muestra algunos atributos importantes de una subred recién creada.

```
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1b',
        'AvailableIpAddressCount': 59,
        'CidrBlock': '10.0.0.64/26',
        'State': 'available',
        'SubnetId': 'subnet-44b4a44f4e44db444',
        'VpcId': 'vpc-555fc5df555e555dc',
        ...
        }
}
```

Paso 2: creación del grupo de subredes de base de datos para el clon

Si va a crear el clon en una VPC diferente o en un conjunto diferente de subredes dentro de la misma VPC, debe crear un nuevo grupo de subredes de base de datos y especificarlo al crear el clon.

Asegúrese de conocer todos los siguientes detalles. Puede encontrarlos todos en el resultado de los ejemplos anteriores.

1. VPC del clúster original. Para obtener instrucciones, consulte Paso 3: comprobación de las subredes del clúster original.
2. VPC del clon, si lo está creando en una VPC diferente. Para obtener instrucciones, consulte Paso 5: comprobación de las VPC que puede utilizar para el clon.

3. Tres zonas de disponibilidad (AZ) asociadas al almacenamiento de Aurora del clúster original. Para obtener instrucciones, consulte Paso 1: comprobación de las zonas de disponibilidad del clúster original.
4. Dos o tres AZ asociadas al grupo de subredes de base de datos del clúster original. Para obtener instrucciones, consulte Paso 2: comprobación del grupo de subredes de base de datos del clúster original.
5. Los ID de subred y las AZ asociadas de todas las subredes de la VPC que tiene pensado usar para el clon. Utilice el mismo comando `describe-subnets` que en Paso 3: comprobación de las subredes del clúster original, sustituyendo el ID de VPC de la VPC de destino.

Compruebe cuántas AZ están asociadas al almacenamiento del clúster original y a las subredes de la VPC de destino. Para crear el clon correctamente, debe haber dos o tres AZ en común. Si tiene menos de dos AZ en común, vuelva a Paso 1: creación de las subredes para el clon. Cree una, dos o tres subredes nuevas asociadas a las AZ asociadas al almacenamiento del clúster original.

Elija subredes en la VPC de destino que estén asociadas a las mismas AZ que el almacenamiento de Aurora en el clúster original. Lo ideal sería elegir tres zonas de disponibilidad (AZ). De este modo, dispondrá de la máxima flexibilidad para distribuir las instancias de base de datos del clon en varias zonas de disponibilidad y conseguir una alta disponibilidad de los recursos de computación.

Ejecute un comando similar al siguiente para crear el nuevo grupo de subredes de base de datos. Sustituya los ID de las subredes en la lista. Si especifica los ID de subred mediante variables de entorno, asegúrese de citar la lista de parámetros `--subnet-ids`, de forma que se conserven las comillas dobles alrededor de los ID.

```
aws rds create-db-subnet-group --db-subnet-group-name my_subnet_group \
  --subnet-ids '["my_subnet_1","my_subnet_2","my_subnet3"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets for clone'
```

El siguiente ejemplo muestra el resultado parcial del comando `create-db-subnet-group`.

```
{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'my_subnet_group',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets for clone',
        'VpcId': 'vpc-555fc5df555e555dc',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
```

```
                'SubnetIdentifier': 'my_subnet_1',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1c'
                },
                'SubnetStatus': 'Active'
            },
            {
                'SubnetIdentifier': 'my_subnet_2',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1d'
                },
                'SubnetStatus': 'Active'
            }
            ...
        ],
        'SupportedNetworkTypes': [
            'IPV4'
        ]
    }
}
```

En este momento, todavía no ha creado el clon. Ha creado todos los recursos de subred y VPC relevantes para poder especificar los parámetros adecuados para los comandos `restore-db-cluster-to-point-in-time` y `create-db-instance` al crear el clon.

Creación de un clon de Aurora con una nueva configuración de red

Una vez que se haya asegurado de que se ha establecido la configuración correcta de VPC, subredes, AZ y grupos de subredes para que la utilice el nuevo clúster, podrá realizar la operación de clonación propiamente dicha. Los siguientes ejemplos de CLI destacan opciones como `--db-subnet-group-name`, `--availability-zone` y `--vpc-security-group-ids` que se especifican en los comandos para configurar el clon y sus instancias de base de datos.

- Paso 1: especificación del grupo de subredes de base de datos para el clon
- Paso 2: especificación de la configuración de red para las instancias del clon
- Paso 3: establecimiento de la conectividad de un sistema cliente a un clon

Paso 1: especificación del grupo de subredes de base de datos para el clon

Al crear el clon, puede configurar todos los ajustes correctos de VPC, subred y AZ especificando un grupo de subredes de base de datos. Utilice los comandos de los ejemplos anteriores para

comprobar todas las relaciones y asignaciones que entran en el grupo de subredes de base de datos.

Por ejemplo, los siguientes comandos muestran cómo clonar un clúster original en un clon. En el primer ejemplo, el clúster de origen está asociado a dos subredes y el clon a tres subredes. El segundo ejemplo muestra el caso contrario: la clonación de un clúster con tres subredes a un clúster con dos subredes.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-3-subnets \
  --db-cluster-identifier cluster-cloned-to-2-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name two-subnets
```

Si tiene pensado utilizar instancias Aurora sin servidor v2 en el clon, incluya una opción `--serverless-v2-scaling-configuration` al crear el clon, tal y como se muestra. De este modo, podrá utilizar la clase `db.serverless` al crear instancias de base de datos en el clon. También puede modificar el clon más adelante para añadir este atributo de configuración de escalado. Los números de capacidad de este ejemplo permiten que cada instancia sin servidor v2 en el clúster escale entre 2 y 32 unidades de capacidad (ACU) de Aurora. Para obtener información sobre la característica Aurora sin servidor v2 y sobre cómo elegir el rango de capacidad, consulte Uso de Aurora Serverless v2.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-2-subnets \
  --db-cluster-identifier cluster-cloned-to-3-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name three-subnets \
  --serverless-v2-scaling-configuration 'MinCapacity=2,MaxCapacity=32'
```

Independientemente del número de subredes utilizadas por las instancias de base de datos, el almacenamiento de Aurora para el clúster de origen y el clon está asociado a tres AZ. En el siguiente ejemplo, se enumeran las AZ asociadas al clúster original y al clon, para los dos comandos `restore-db-cluster-to-point-in-time` de los ejemplos anteriores.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-with-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
```

```
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-with-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d
```

Como al menos dos de las AZ se superponen entre cada par de clústeres originales y clonados, ambos clústeres pueden acceder al mismo almacenamiento subyacente de Aurora.

Paso 2: especificación de la configuración de red para las instancias del clon

Al crear instancias de base de datos en el clon, de forma predeterminada, estas heredan el grupo de subredes de base de datos del propio clúster. De esta forma, Aurora asigna automáticamente cada instancia a una subred concreta y la crea en la AZ asociada a la subred. Esta opción resulta práctica, sobre todo para los sistemas de desarrollo y prueba, ya que no es necesario realizar un seguimiento de los ID de subred ni de las AZ al agregar nuevas instancias al clon.

Como alternativa, puede especificar la zona de disponibilidad (AZ) al crear una instancia de base de datos Aurora para el clon. La AZ que especifique debe ser del conjunto de las AZ asociadas al clon. Si el grupo de subredes de base de datos que utiliza para el clon solo contiene dos subredes, solo podrá elegir entre las AZ asociadas a esas dos subredes. Esta opción ofrece flexibilidad y resiliencia para sistemas de alta disponibilidad, ya que puede asegurarse de que la instancia de escritor y la instancia de lector en espera estén en diferentes zonas de disponibilidad. O bien si añade lectores adicionales al clúster, puede asegurarse de que estén distribuidos en tres zonas de disponibilidad.

De esta forma, incluso en el caso inusual de que se produzca un error en la zona de distribución, seguirá teniendo una instancia de escritor y otra de lector en otras dos zonas de disponibilidad.

En el siguiente ejemplo, se agrega una instancia de base de datos aprovisionada a un clúster de Aurora PostgreSQL clonado que usa un grupo de subredes de base de datos personalizado.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_postgresql_clone \
  --db-instance-identifier my_postgres_instance \
  --db-subnet-group-name my_new_subnet \
  --engine aurora-postgresql \
  --db-instance-class db.t4g.medium
```

El siguiente ejemplo muestra el resultado parcial de un comando de este tipo.

```
{
  'DBInstanceIdentifier': 'my_postgres_instance',
  'DBClusterIdentifier': 'my_aurora_postgresql_clone',
  'DBInstanceClass': 'db.t4g.medium',
  'DBInstanceStatus': 'creating'
  ...
}
```

El siguiente ejemplo agrega una instancia de base de datos Aurora sin servidor v2 a un clon de Aurora MySQL que usa un grupo de subredes de base de datos personalizado. Para poder utilizar instancias de sin servidor v2, asegúrese de especificar la opción `--serverless-v2-scaling-configuration` para el comando `restore-db-cluster-to-point-in-time`, tal y como se muestra en los ejemplos anteriores.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_mysql_clone \
  --db-instance-identifier my_mysql_instance \
  --db-subnet-group-name my_other_new_subnet \
  --engine aurora-mysql \
  --db-instance-class db.serverless
```

El siguiente ejemplo muestra el resultado parcial de un comando de este tipo.

```
{
  'DBInstanceIdentifier': 'my_mysql_instance',
  'DBClusterIdentifier': 'my_aurora_mysql_clone',
  'DBInstanceClass': 'db.serverless',
```

```
  'DBInstanceStatus': 'creating'
  ...
}
```

Paso 3: establecimiento de la conectividad de un sistema cliente a un clon

Si ya se está conectando a un clúster de Aurora desde un sistema cliente, es posible que desee permitir el mismo tipo de conectividad a un clon nuevo. Por ejemplo, podría conectarse al clúster original desde una instancia de Amazon Cloud9 o EC2. Para permitir conexiones desde los mismos sistemas cliente o desde otros nuevos que cree en la VPC de destino, configure grupos de subredes de base de datos y grupos de seguridad de VPC equivalentes a los de la VPC. A continuación, especifique el grupo de subredes y los grupos de seguridad al crear el clon.

Los siguientes ejemplos configuran un clon de Aurora sin servidor v2. Esa configuración se basa en la combinación de `--engine-mode provisioned` y `--serverless-v2-scaling-configuration` al crear el clúster de base de datos y, después, `--db-instance-class db.serverless` al crear cada instancia de base de datos del clúster. El modo de motor `provisioned` es el predeterminado, por lo que puede omitir esa opción si lo prefiere.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier serverless-sql-postgres\
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-subnet-group-name 'default-vpc-1234' \
  --vpc-security-group-ids 'sg-4567' \
  --serverless-v2-scaling-configuration 'MinCapacity=0.5,MaxCapacity=16' \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

A continuación, al crear las instancias de base de datos en el clon, especifique la misma opción `--db-subnet-group-name`. Si lo desea, puede incluir la opción `--availability-zone` y especificar una de las AZ asociadas a las subredes de ese grupo de subredes. Esa AZ también debe ser una de las AZ asociadas al clúster original.

```
aws rds create-db-instance \
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-instance-identifier serverless-sql-postgres-clone-instance \
  --db-instance-class db.serverless \
  --db-subnet-group-name 'default-vpc-987zyx654' \
  --availability-zone 'us-east-1c' \
  --engine aurora-postgresql
```

Traslado de un clúster de subredes públicas a subredes privadas

Puede usar la clonación para migrar un clúster entre subredes públicas y privadas. Podría hacerlo al agregar capas de seguridad adicionales a su aplicación antes de implementarla en producción. Para este ejemplo, debe tener configuradas las subredes privadas y la puerta de enlace de NAT antes de iniciar el proceso de clonación con Aurora.

Para los pasos relacionados con Aurora, puede seguir los mismos pasos generales que en los ejemplos anteriores para Recopilación de información sobre el entorno de red yCreación de un clon de Aurora con una nueva configuración de red. La diferencia principal es que, incluso si tiene subredes públicas que se asignan a todas las AZ del clúster original, ahora debe comprobar que tiene suficientes subredes privadas para un clúster de Aurora y que esas subredes están asociadas a todas las mismas AZ que se utilizan para el almacenamiento de Aurora en el clúster original. Al igual que en otros casos de uso de clonación, puede crear el grupo de subredes de base de datos del clon con tres o dos subredes privadas asociadas a las AZ requeridas. Sin embargo, si usa dos subredes privadas en el grupo de subredes de base de datos, debe tener una tercera subred privada asociada a la tercera AZ utilizada para el almacenamiento de Aurora en el clúster original.

Puede consultar esta lista de comprobación para verificar que se cumplen todos los requisitos para realizar este tipo de operación de clonación.

- Registre las tres AZ asociadas al clúster original. Para obtener instrucciones, consulte Paso 1: comprobación de las zonas de disponibilidad del clúster original.
- Registre las dos o tres AZ asociadas a las subredes públicas en el grupo de subredes de base de datos del clúster original. Para obtener instrucciones, consulte Paso 3: comprobación de las subredes del clúster original.
- Cree subredes privadas que se asignen a las tres AZ asociadas al clúster original. Realice también cualquier otra configuración de red, como crear una puerta de enlace de NAT, para poder comunicarse con las subredes privadas. Para obtener instrucciones detalladas, consulte Crear una subred en la Guía del usuario de Amazon Virtual Private Cloud.
- Cree un nuevo grupo de subredes de base de datos que contenga tres o dos de las subredes privadas que están asociadas a las AZ desde el primer punto. Para obtener instrucciones, consulte Paso 2: creación del grupo de subredes de base de datos para el clon.

Cuando se cumplan todos los requisitos previos, puede pausar la actividad de la base de datos en el clúster original mientras crea el clon y cambiar la aplicación para usarlo. Tras crear el clon y verificar que puede conectarse a él, ejecutar el código de la aplicación, etc., podrá dejar de utilizar el clúster original.

Ejemplo de extremo a extremo de creación de un clon entre VPC

Para crear un clon en una VPC diferente a la original, se siguen los mismos pasos generales que en los ejemplos anteriores. Dado que el ID de VPC es una propiedad de las subredes, en realidad no se especifica el ID de VPC como parámetro al ejecutar cualquiera de los comandos de la CLI de RDS. La principal diferencia es que es más probable que necesite crear nuevas subredes, nuevas subredes asignadas a zonas de disponibilidad específicas, un grupo de seguridad de VPC y un nuevo grupo de subredes de base de datos. Esto es especialmente cierto si se trata del primer clúster de Aurora que se crea en esa VPC.

Puede consultar esta lista de comprobación para verificar que se cumplen todos los requisitos para realizar este tipo de operación de clonación.

- Registre las tres AZ asociadas al clúster original. Para obtener instrucciones, consulte Paso 1: comprobación de las zonas de disponibilidad del clúster original.
- Registre las tres o dos AZ asociadas a las subredes en el grupo de subredes de base de datos del clúster original. Para obtener instrucciones, consulte Paso 2: comprobación del grupo de subredes de base de datos del clúster original.
- Cree subredes que se asignen a las tres AZ asociadas al clúster original. Para obtener instrucciones, consulte Paso 1: creación de las subredes para el clon.
- Realice cualquier otra configuración de red, como configurar un grupo de seguridad de VPC, para los sistemas cliente, los servidores de aplicaciones, etc., para poder comunicarse con las instancias de base de datos del clon. Para obtener instrucciones, consulte Control de acceso con grupos de seguridad.
- Cree un nuevo grupo de subredes de base de datos que contenga tres o dos de las subredes que están asociadas a las AZ desde el primer punto. Para obtener instrucciones, consulte Paso 2: creación del grupo de subredes de base de datos para el clon.

Cuando se cumplan todos los requisitos previos, puede pausar la actividad de la base de datos en el clúster original mientras crea el clon y cambiar la aplicación para usarlo. Tras crear el clon y verificar que puede conectarse a él, ejecutar el código de la aplicación, etc., podrá plantearse si va a mantener tanto el original como los clones ejecutándose, o bien si va a dejar de utilizar el clúster original.

Los siguientes ejemplos de Linux muestran la secuencia de operaciones de AWS CLI para clonar un clúster de base de datos Aurora de una VPC a otra. Algunos campos que no son relevantes para los ejemplos no se muestran en el resultado del comando.

En primer lugar, comprobamos los ID de las VPC de origen y destino. El nombre descriptivo que se asigna a una VPC al crearla se representa como una etiqueta en los metadatos de la VPC.

```
$ aws ec2 describe-vpcs --query '*[].[VpcId,Tags]'
[
    [
        'vpc-0f0c0fc0000b0ffb0',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-source'
            }
        ]
    ],
    [
        'vpc-9e99d9f99a999bd99',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-dest'
            }
        ]
    ]
]
```

El clúster original ya existe en la VPC de origen. Para configurar el clon con el mismo conjunto de AZ para el almacenamiento de Aurora, verificamos las AZ utilizadas por el clúster original.

```
$ aws rds describe-db-clusters --db-cluster-identifier original-cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

Nos aseguramos de que haya subredes que correspondan a las AZ utilizadas por el clúster original: `us-east-1c`, `us-east-1d` y `us-east-1f`.

```
$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
  --availability-zone us-east-1c --cidr-block 10.0.0.128/28
{
    'Subnet': {
```

```
        'AvailabilityZone': 'us-east-1c',
        'SubnetId': 'subnet-3333a33be3ef3e333',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1d --cidr-block 10.0.0.160/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1d',
        'SubnetId': 'subnet-4eeb444cd44b4d444',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1f --cidr-block 10.0.0.224/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1f',
        'SubnetId': 'subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}
```

Este ejemplo confirma que hay subredes que se asignan a las AZ necesarias en la VPC de destino.

```
aws ec2 describe-subnets --query 'sort_by(*[] | [?VpcId == `vpc-9e99d9f99a999bd99`] |
[].{SubnetId:SubnetId,VpcId:VpcId,AvailabilityZone:AvailabilityZone},
 &AvailabilityZone)' --output table

-------------------------------------------------------------------------
|                           DescribeSubnets                             |
+------------------+---------------------------+-------------------------+
| AvailabilityZone |         SubnetId          |          VpcId          |
+------------------+---------------------------+-------------------------+
|  us-east-1a      |  subnet-000ff0e00000c0aea  |  vpc-9e99d9f99a999bd99  |
|  us-east-1b      |  subnet-1111d111111ca11b1  |  vpc-9e99d9f99a999bd99  |
|  us-east-1c      | subnet-3333a33be3ef3e333   |  vpc-9e99d9f99a999bd99  |
|  us-east-1d      | subnet-4eeb444cd44b4d444   |  vpc-9e99d9f99a999bd99  |
|  us-east-1f      | subnet-66eea6666fb66d66c   |  vpc-9e99d9f99a999bd99  |
+------------------+---------------------------+-------------------------+
```

Antes de crear un clúster de base de datos Aurora en la VPC, debe tener un grupo de subredes de base de datos con subredes que se asignen a las AZ utilizadas para el almacenamiento de Aurora. Al crear un clúster normal, puede usar cualquier conjunto de tres zonas de disponibilidad (AZ). Al clonar un clúster existente, el grupo de subredes debe coincidir con al menos dos de las tres zonas de disponibilidad que utiliza para el almacenamiento de Aurora.

```
$ aws rds create-db-subnet-group \
  --db-subnet-group-name subnet-group-in-other-vpc \
  --subnet-ids
 '["subnet-3333a33be3ef3e333","subnet-4eeb444cd44b4d444","subnet-66eea6666fb66d66c"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c'

{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'subnet-group-in-other-vpc',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'subnet-4eeb444cd44b4d444',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1d' }
            },
            {
                'SubnetIdentifier': 'subnet-3333a33be3ef3e333',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1c' }
            },
            {
                'SubnetIdentifier': 'subnet-66eea6666fb66d66c',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1f' }
            }
        ]
    }
}
```

Ahora las subredes y el grupo de subredes de base de datos están en su lugar. En el siguiente ejemplo, se muestra el `restore-db-cluster-to-point-in-time` que clona el clúster. La opción `--db-subnet-group-name` asocia el clon al conjunto correcto de subredes que se asignan al conjunto correcto de AZ del clúster original.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier original-cluster \
  --db-cluster-identifier clone-in-other-vpc \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name subnet-group-in-other-vpc

{
  'DBClusterIdentifier': 'clone-in-other-vpc',
  'DBSubnetGroup': 'subnet-group-in-other-vpc',
  'Engine': 'aurora-postgresql',
  'EngineVersion': '15.4',
  'Status': 'creating',
  'Endpoint': 'clone-in-other-vpc.cluster-c0abcdef.us-east-1.rds.amazonaws.com'
}
```

El siguiente ejemplo confirma que el almacenamiento de Aurora en el clon usa el mismo conjunto de AZ que en el clúster original.

```
$ aws rds describe-db-clusters --db-cluster-identifier clone-in-other-vpc \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

En este momento, puede crear instancias de base de datos para el clon. Asegúrese de que el grupo de seguridad de VPC asociado a cada instancia permita las conexiones desde los rangos de direcciones IP que utiliza para las instancias EC2, los servidores de aplicaciones, etc., que se encuentran en la VPC de destino.

Clonación entre cuentas con AWS RAM y Amazon Aurora

Al usar AWS Resource Access Manager (AWS RAM) con Amazon Aurora, puede compartir clústeres de base de datos de Aurora y clones que pertenecen a su cuenta de AWS con otra cuenta de AWS u organización. La clonación entre cuentas es mucho más rápida en dichas situaciones que crear y restaurar una instantánea de base de datos. Puede crear un clon de uno de sus clústeres de base de datos de Aurora y compartir el clon. O puede compartir su clúster de base de datos de Aurora con otra cuenta de AWS y dejar que el titular de la cuenta cree el clon. El enfoque que elija depende de su caso de uso.

Por ejemplo, puede que necesite compartir regularmente un clon de su base de datos financiera con el equipo de auditoría interna de su organización. En este caso, su equipo de auditoría tiene su propia cuenta de AWS para las aplicaciones que utiliza. Puede darle permiso al equipo de auditoría de la cuenta de AWS para acceder a su clúster de base de datos de Aurora y clonarlo según sea necesario.

Por otro lado, si un proveedor externo audita sus datos financieros, es posible que prefiera crear el clon usted mismo. Luego, conceda al proveedor externo acceso solo al clon.

También puede utilizar la clonación entre cuentas para admitir muchos de los mismos casos de uso para la clonación dentro de la misma cuenta de AWS, como desarrollo y pruebas. Por ejemplo, su organización podría utilizar diferentes cuentas de AWS para la producción, el desarrollo, las pruebas, etc. Para obtener más información, consulte Información general de la clonación de Aurora.

Por lo tanto, podría ser necesario compartir un clon con otra cuenta de AWS o permitirle a otra cuenta de AWS crear clones de los clústeres de base de datos de Aurora. En cualquier caso, comience por usar AWS RAM para crear un objeto compartido. Para obtener información completa sobre cómo compartir recursos de AWS entre cuentas de AWS, consulte la guía del usuario de AWS RAM.

La creación de un clon entre cuentas requiere acciones de la cuenta de AWS propietaria del clúster original y la cuenta de AWS que crea el clon. En primer lugar, el propietario modifica el clúster para permitir que una o más cuentas lo clonen. Si alguna de las cuentas está en una organización de AWS diferente, AWS genera una invitación para compartir. La otra cuenta debe aceptar la invitación antes de continuar. A continuación, cada cuenta autorizada puede clonar el clúster. En todo este proceso, el clúster se identifica mediante su nombre de recurso de Amazon (ARN) único.

Al igual que con la clonación dentro de la misma cuenta de AWS, el espacio de almacenamiento adicional solo si el origen o el clon realizan cambios a los datos. Los cargos por almacenamiento se aplican entonces en ese momento. Si se borra el clúster de origen, los costes de almacenamiento se distribuyen por igual entre el resto de clústeres clonados.

Temas

- Limitaciones de la clonación entre cuentas
- Permitir a otras cuentas de AWS clonar su clúster
- Clonar un clúster que es propiedad de otra cuenta de AWS

Limitaciones de la clonación entre cuentas

La clonación entre cuentas de Aurora tiene las siguientes limitaciones:

- No puede clonar un clúster de Aurora Serverless v1 en cuentas de AWS.
- No puede ver ni aceptar invitaciones a recursos compartidos con la Consola de administración de AWS. Utilice la AWS CLI, la API de Amazon RDS o la consola de AWS RAM para ver y aceptar invitaciones a recursos compartidos.
- Solo puede crear un clon nuevo desde un recurso compartido con la Cuenta de AWS. Esto se aplica si el recurso compartido es un clúster de base de datos de Aurora original o si es un clon creado anteriormente.
- Solo puede crear un clon nuevo desde un clon que se ha compartido con su cuenta de AWS.
- No puede compartir recursos (clones o clústeres de base de datos de Aurora) que se han compartido con su cuenta de AWS.
- Puede crear un máximo de 15 clones entre cuentas desde cualquier clúster de base de datos de Aurora.
- Cada uno de estos 15 clones de diferentes cuentas debe ser propiedad de una cuenta de AWS diferente. Es decir, solo puede crear un clon entre cuentas de un clúster dentro de cualquier cuenta de AWS.
- Después de clonar un clúster, se considera que el clúster original y su clon son los mismos a efectos de aplicar los límites de los clones entre cuentas. No se pueden crear clones entre cuentas tanto del clúster original como del clúster clonado dentro de la misma cuenta de AWS. El número total de clones entre cuentas para el clúster original y cualquiera de sus clones no puede superar los 15.
- No puede compartir un clúster de base de datos de Aurora con otras cuentas de AWS a menos que el clúster esté en un estado `ACTIVE`.
- No puede cambiar el nombre de un clúster de base de datos de Aurora que se ha compartido con otras cuentas de AWS.
- No puede crear un clon entre cuentas de un clúster que esté cifrado con la clave de RDS predeterminada.
- No puede crear clones no cifrados en una cuenta de AWS desde clústeres de base de datos de Aurora cifrados que han sido compartidos por otra cuenta de AWS. El propietario del clúster también debe concederle permiso para obtener acceso a la del clúster fuente AWS KMS key. Ahora bien, puede utilizar una clave distinta cuando cree el clon.

Permitir a otras cuentas de AWS clonar su clúster

Para permitir a otras cuentas de AWS clonar un clúster de su propiedad, utilice AWS RAM para establecer el permiso de uso compartido. Al hacerlo, también se envía una invitación a cada una de las otras cuentas que está en una organización de AWS diferente.

Para que los procedimientos compartan recursos de su propiedad en la consola de AWS RAM, consulte Compartir recursos de su propiedad en la guía del usuario de AWS RAM.

Temas

- Conceder permisos a otras cuentas de AWS para clonar su clúster
- Verificar si un clúster de su propiedad se comparte con otras cuentas de AWS

Conceder permisos a otras cuentas de AWS para clonar su clúster

Si el clúster que está compartiendo está cifrado, también se comparte la AWS KMS key del clúster. Puede permitir que los usuarios o roles de AWS Identity and Access Management (IAM) de una cuenta de AWS utilicen una clave de KMS en una cuenta diferente.

Para hacer esto, primero tiene que agregar la cuenta externa (usuario raíz) a la política de claves de KMS a través de AWS KMS. No se añaden los usuarios o roles individuales a la política de claves, solo la cuenta externa que los posee. Solo puede compartir una clave de KMS que cree, no la clave de servicio de RDS predeterminada. Para obtener más información sobre el control de acceso de las claves de KMS, consulte Autenticación y control de acceso de AWS KMS.

Consola

Para conceder permisos para clonar su clúster

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos que quiera compartir para ver su página Details (Detalles) y elija la pestaña Connectivity & security (Conectividad y seguridad).
4. En la sección Share DB cluster with other accounts (Compartir clúster de base de datos con otras cuentas de AWS), ingrese el ID numérico de la cuenta para la cuenta de AWS que quiera permitir clonar este clúster. Para los ID de cuenta en la misma organización, puede comenzar a escribir en el cuadro y luego elegir en el menú.

> **Important**
>
> En algunos casos, es posible que desee una cuenta que no esté en la misma organización de AWS que su cuenta para clonar un clúster. En estos casos, por razones de seguridad la consola no notifica el propietario de ese ID de cuenta o si existe la cuenta.
> Tenga cuidado de introducir números de cuenta que no estén en la misma organización de AWS que su cuenta de AWS. Verifique inmediatamente que ha compartido con la cuenta pretendida.

5. En la página de confirmación, verifique que el ID de cuenta que ha especificado sea correcto. Introduzca `share` en el cuadro de confirmación para confirmar.

   En la página Details (Detalles), aparece una entrada que muestra el ID de cuenta de AWS especificado en Accounts that this DB cluster is shared with (Cuentas con las que se comparte este clúster de base de datos). La columna Status (Estado) inicialmente muestra el estado Pending (Pendiente).

6. Póngase en contacto con el propietario de la otra cuenta de AWS o inicie sesión en dicha cuenta si es propietario de ambas. Pida al propietario de la otra cuenta que acepte la invitación de uso compartido y clone el clúster de base de datos, como se describe a continuación.

AWS CLI

Para conceder permisos para clonar su clúster

1. Recopile la información de los parámetros requeridos. Necesita el ARN de su clúster y el ID numérico de la otra cuenta de AWS.
2. Ejecute el comando AWS RAM de la CLI `create-resource-share`.

   Para Linux, macOS o Unix:

```
aws ram create-resource-share --name descriptive_name \
  --region region \
  --resource-arns cluster_arn \
  --principals other_account_ids
```

   Para Windows:

```
aws ram create-resource-share --name descriptive_name ^
  --region region ^
  --resource-arns cluster_arn ^
  --principals other_account_ids
```

Para incluir varios ID de cuenta para el parámetro `--principals`, separe los ID entre sí con espacios. Para especificar si los ID de cuenta permitidos pueden estar fuera de la organización de AWS, incluya el parámetro `--allow-external-principals` o `--no-allow-external-principals` para `create-resource-share`.

AWS RAMAPI de

Para conceder permisos para clonar su clúster

1. Recopile la información de los parámetros requeridos. Necesita el ARN de su clúster y el ID numérico de la otra cuenta de AWS.
2. Llame a la operación CreateResourceShare de la API de AWS RAM y especifique los siguientes valores:
   - Especifique los ID de cuenta de una o más cuentas de AWS en el parámetro `principals`.
   - Especifique los ARN de uno o más clústeres de base de datos de Aurora en el parámetro `resourceArns`.
   - Especifique si los ID de cuenta permitidos pueden estar fuera de la organización de AWS o no, incluyendo un valor booleano para el parámetro `allowExternalPrincipals`.

Recreación de un clúster que utiliza la clave predeterminada de RDS

Si el clúster cifrado que desea compartir utiliza la clave predeterminada de RDS, asegúrese de volver a crear el clúster. Para ello, cree una instantánea manual del clúster de base de datos, utilice una AWS KMS key y, a continuación, restaure el clúster a un clúster nuevo. A continuación, comparta el nuevo clúster. Para ello, siga estos pasos:

Para volver a crear un clúster cifrado que utiliza la clave de RDS predeterminada

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).

3. Elija una instantánea.
4. En Actions (Acciones), elija Copy Snapshot (Copiar instantánea) y, a continuación, elija Enable encryption (Habilitar cifrado).
5. En AWS KMS key, elija la nueva clave de cifrado que desee utilizar.
6. Restaure la instantánea copiada. Para ello, siga el procedimiento en Restauración de una instantánea de clúster de base de datos. La nueva instancia de base de datos utiliza su clave de cifrado nueva.
7. (Opcional) Elimine el clúster de base de datos antiguo si ya no lo necesita más. Para ello, siga el procedimiento en Eliminación de una instantánea de clúster de base de datos. Antes de hacerlo, confirme que su nuevo clúster tiene todos los datos necesarios y que su aplicación puede acceder a ellos correctamente.

Verificar si un clúster de su propiedad se comparte con otras cuentas de AWS

Puede comprobar si otros usuarios tienen permiso para compartir un clúster. Hacerlo puede ayudarle a comprender si el clúster se acerca al límite del número máximo de clones entre cuentas.

Para que los procedimientos compartan recursos utilizando la consola de AWS RAM, consulte Compartir recursos de su propiedad en la guía del usuario de AWS RAM.

AWS CLI

Para descubrir si un clúster de su propiedad se comparte con otras cuentas de AWS

- Llame al comando `list-principals` de la CLI de AWS RAM mediante su ID de cuenta como el propietario del recurso y el ARN de su clúster como el ARN del recurso. Puede ver todas las unidades compartidas con el siguiente comando. Los resultados indican qué cuentas de AWS tienen permiso para clonar el clúster.

```
aws ram list-principals \
    --resource-arns your_cluster_arn \
    --principals your_aws_id
```

AWS RAMAPI de

Para descubrir si un clúster de su propiedad se comparte con otras cuentas de AWS

- Llame a la operación ListPrincipals de la API de AWS RAM. Utilice su ID de cuenta como el propietario del recurso y el ARN de su clúster como el ARN del recurso.

Clonar un clúster que es propiedad de otra cuenta de AWS

Para clonar un clúster propiedad de otra cuenta de AWS, utilice AWS RAM para obtener permiso para crear un clon. Una vez que tenga el permiso requerido, utilice el procedimiento estándar para clonar un clúster de Aurora.

También puede comprobar si un clúster de su propiedad es un clon de un clúster propiedad de una cuenta de AWS diferente.

Para que funcionen los procedimientos con recursos que son propiedad de otros en la consola de AWS RAM, consulte Acceso a los recursos compartidos con usted en la guía del usuario de AWS RAM.

Temas

- Visualización de invitaciones para clonar clústeres propiedad de otras cuentas de AWS
- Aceptación de invitaciones para compartir clústeres propiedad de otras cuentas de AWS
- Clonar un clúster de Aurora que es propiedad de otra cuenta de AWS
- Comprobar si un clúster de base de datos es un clon entre cuentas

Visualización de invitaciones para clonar clústeres propiedad de otras cuentas de AWS

Para trabajar con invitaciones para clonar clústeres propiedad de cuentas de AWS en otras organizaciones de AWS, utilice la AWS CLI, la consola de AWS RAM o la API de AWS RAM. Actualmente, no puede realizar este procedimiento utilizando la consola de Amazon RDS.

Para que funcionen los procedimientos con invitaciones en la consola de AWS RAM, consulte Acceso a los recursos compartidos con usted en la guía del usuario de AWS RAM.

AWS CLI

Para ver invitaciones para clonar clústeres propiedad de otras cuentas de AWS

1. Ejecute el comando AWS RAM de la CLI `get-resource-share-invitations`.

   ```
   aws ram get-resource-share-invitations --region region_name
   ```

   Los resultados del comando anterior muestran todas las invitaciones para clonar clústeres, incluidas las que ya haya aceptado o rechazado.

2. (Opcional) Filtre la lista de forma que vea solo las invitaciones que requieren una acción por su parte. Para hacerlo, añada el parámetro `--query 'resourceShareInvitations[?status==`PENDING`]'`.

AWS RAMAPI de

Para ver invitaciones para clonar clústeres propiedad de otras cuentas de AWS

1. Llame a la operación `GetResourceShareInvitations` de la API de AWS RAM. Esta operación devuelve estas invitaciones, incluidas las que ya haya aceptado o rechazado.
2. (Opcional) Busque solo invitaciones que requieran de una acción por su parte comprobando que el campo de devolución `resourceShareAssociations` tenga un valor de `status` de `PENDING`.

Aceptación de invitaciones para compartir clústeres propiedad de otras cuentas de AWS

Puede aceptar invitaciones para compartir clústeres propiedad de otras cuentas de AWS que estén en diferentes organizaciones de AWS. Para trabajar con estas invitaciones, utilice la AWS CLI, las API de AWS RAM y RDS o la consola de AWS RAM. Actualmente, no puede realizar este procedimiento utilizando la consola de RDS.

Para que funcionen los procedimientos con invitaciones en la consola de AWS RAM, consulte Acceso a los recursos compartidos con usted en la guía del usuario de AWS RAM.

AWS CLI

Para aceptar una invitación para compartir un clúster desde otra cuenta de AWS

1. Busque el ARN de la invitación ejecutando el comando `get-resource-share-invitations` de la CLI de AWS RAM, como se ha mostrado anteriormente.
2. Acepte la invitación llamando al comando `accept-resource-share-invitation` de la CLI de AWS RAM, como se muestra a continuación.

   Para Linux, macOS o Unix:

   ```
   aws ram accept-resource-share-invitation \
     --resource-share-invitation-arn invitation_arn \
     --region region
   ```

   Para Windows:

   ```
   aws ram accept-resource-share-invitation ^
     --resource-share-invitation-arn invitation_arn ^
     --region region
   ```

AWS RAM y API de RDS

Para aceptar invitaciones para compartir el clúster de alguien

1. Busque el ARN de la invitación llamando a la operación `GetResourceShareInvitations` de la API AWS RAM, como se ha mostrado anteriormente.
2. Pase ese ARN como el parámetro `resourceShareInvitationArn` a la operación AcceptResourceShareInvitation de la API de RDS.

Clonar un clúster de Aurora que es propiedad de otra cuenta de AWS

Después de aceptar la invitación de otra cuenta de AWS propiedad del clúster de base de datos, como se ha mostrado anteriormente, puede clonar el clúster.

Consola

Para clonar un clúster de Aurora que es propiedad de otra cuenta de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).

   En la parte superior de la lista de bases de datos, debe ver uno o más elementos con un valor de Role (Role) de `Shared from account #account_id`. Por razones de seguridad, solo puede ver información limitada sobre los clústeres originales. Las propiedades que puede ver son las de motor de base de datos y versión que deben ser las mismas en su clúster clonado.
3. Elija el clúster que pretende clonar.
4. En Actions (Acciones), elija Create alias (Crear alias).
5. Siga el procedimiento de Consola para finalizar la configuración del clúster clonado.
6. Según sea necesario, habilite el cifrado del clúster clonado. Si el clúster que esté clonando está cifrado, debe habilitar el cifrado del clúster clonado. La cuenta de AWS que compartió el clúster con usted también debe compartir la clave de KMS que se utilizó para cifrar el clúster. Puede utilizar la misma clave de KMS para cifrar el clon, o bien puede utilizar su propia clave de KMS. No puede crear un clon entre cuentas para un clúster que esté cifrado con la clave de KMS predeterminada.

   La cuenta que posee la clave de cifrado debe conceder permiso a la cuenta de destino para utilizar la clave mediante una política de claves. Este proceso es similar al utilizado para compartir las instantáneas cifradas, utilizando una política de claves que otorga permiso a la cuenta de destino para utilizar la clave.

AWS CLI

Para clonar un clúster de Aurora que es propiedad de otra cuenta de AWS

1. Acepte la invitación de otra cuenta de AWS propiedad del clúster de base de datos, como se ha mostrado anteriormente.
2. Clone el clúster especificando el ARN completo del clúster de origen en el parámetro `source-db-cluster-identifier` del comando `restore-db-cluster-to-point-in-time` de la CLI de RDS, como se muestra a continuación.

Si el ARN pasado como el `source-db-cluster-identifier` no se ha compartido, se devuelve el mismo error que si no existiera el clúster especificado.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier=arn:aws:rds:arn_details \
  --db-cluster-identifier=new_cluster_id \
  --restore-type=copy-on-write \
  --use-latest-restorable-time
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
  --source-db-cluster-identifier=arn:aws:rds:arn_details ^
  --db-cluster-identifier=new_cluster_id ^
  --restore-type=copy-on-write ^
  --use-latest-restorable-time
```

3. Si el clúster que está clonando está cifrado, cifre su clúster clonado incluyendo un parámetro `kms-key-id`. Este valor `kms-key-id` puede ser el mismo que el utilizado para cifrar el clúster de base de datos original o su propia clave de KMS. Su cuenta debe tener permiso para utilizar dicha clave de cifrado.

   Para Linux, macOS o Unix:

   ```
   aws rds restore-db-cluster-to-point-in-time \
     --source-db-cluster-identifier=arn:aws:rds:arn_details \
     --db-cluster-identifier=new_cluster_id \
     --restore-type=copy-on-write \
     --use-latest-restorable-time \
     --kms-key-id=arn:aws:kms:arn_details
   ```

   Para Windows:

   ```
   aws rds restore-db-cluster-to-point-in-time ^
     --source-db-cluster-identifier=arn:aws:rds:arn_details ^
     --db-cluster-identifier=new_cluster_id ^
     --restore-type=copy-on-write ^
     --use-latest-restorable-time ^
   ```

```
  --kms-key-id=arn:aws:kms:arn_details
```

La cuenta que posee la clave de cifrado debe conceder permiso a la cuenta de destino para utilizar la clave mediante una política de claves. Este proceso es similar al utilizado para compartir las instantáneas cifradas, utilizando una política de claves que otorga permiso a la cuenta de destino para utilizar la clave. A continuación se incluye una política de claves.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
```

```
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> El comando restore-db-cluster-to-point-in-time de la CLI de AWS solo restaura el clúster de base de datos, no las instancias de base de datos de dicho clúster. Para crear instancias de base de datos para el clúster de base de datos restaurado, invoque el comando create-db-instance. Especifique el identificador del clúster de base de datos restaurado en `--db-cluster-identifier`.
>
> Solo puede crear instancias de base de datos después de que se haya completado el comando `restore-db-cluster-to-point-in-time` y de que el clúster de base de datos esté disponible.

API de RDS

Para clonar un clúster de Aurora que es propiedad de otra cuenta de AWS

1. Acepte la invitación de otra cuenta de AWS propiedad del clúster de base de datos, como se ha mostrado anteriormente.
2. Clone el clúster especificando el ARN completo del clúster de origen en el parámetro `SourceDBClusterIdentifier` de la operación `RestoreDBClusterToPointInTime` de la API de RDS.

   Si el ARN pasado como el `SourceDBClusterIdentifier` no se ha compartido, se devuelve el mismo error que se el clúster especificado no exista.

3. Si el clúster que está clonando está cifrado, incluya un parámetro `KmsKeyId` para cifrar el clúster clonado. Este valor `kms-key-id` puede ser el mismo que el utilizado para cifrar el clúster de base de datos original o su propia clave de KMS. Su cuenta debe tener permiso para utilizar dicha clave de cifrado.

   Al clonar un volumen, la cuenta de destino debe tener permiso para utilizar la clave de cifrado utilizada para cifrar el clúster de origen. Aurora cifra el nuevo clúster clonado con la clave de cifrado especificada en `KmsKeyId`.

   La cuenta que posee la clave de cifrado debe conceder permiso a la cuenta de destino para utilizar la clave mediante una política de claves. Este proceso es similar al utilizado para compartir las instantáneas cifradas, utilizando una política de claves que otorga permiso a la cuenta de destino para utilizar la clave. A continuación se incluye una política de claves.

   JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> La operación RestoreDBClusterToPointInTime de la API de RDS solo restaura el clúster de base de datos, no las instancias de base de datos de dicho clúster. Para crear instancias de base de datos para el clúster de base de datos restaurado, invoque la operación CreateDBInstance de la API de RDS. Especifique el identificador del clúster de base de datos restaurado en `DBClusterIdentifier`. Solo puede crear instancias de base de datos después de que se haya completado la operación `RestoreDBClusterToPointInTime` y el clúster de base de datos esté disponible.

Comprobar si un clúster de base de datos es un clon entre cuentas

El objeto `DBClusters` identifica si cada clúster es un clon entre cuentas. Puede ver los clústeres que tienen permiso para clonar utilizando el la opción `include-shared` cuando ejecute el comando `describe-db-clusters` de la CLI de RDS. Sin embargo, no puede ver la mayoría de los detalles de configuración de dichos clústeres.

AWS CLI

Verificar si un clúster de base de datos es un clon entre cuentas

- Llame al comando de CLI de RDS `describe-db-clusters`.

  El siguiente ejemplo muestra cómo los clústeres de base de datos de clonación entre cuentas reales o potenciales aparece en la salida de `describe-db-clusters`. Para clústeres existentes propiedad de su cuenta de AWS, el campo `CrossAccountClone` indica si el clúster es un clon de un clúster de base de datos propiedad de otra cuenta de AWS.

  En algunos casos, una entrada podría tener un número de cuenta de AWS diferente al suyo en el campo `DBClusterArn`. En este caso, dicha entrada representa un clúster propiedad de una cuenta de AWS diferente y que se puede clonar. Dichas entradas tienen pocos cambios aparte de `DBClusterArn`. Al crear el clúster clonado, especifique los mismos valores de `StorageEncrypted`, `Engine` y `EngineVersion` que el clúster original.

```
$aws rds describe-db-clusters --include-shared --region us-east-1
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0
  ]
```

```
}
```

API de RDS

Verificar si un clúster de base de datos es un clon entre cuentas

- Llame a la operación DescribeDBClusters de la API de RDS.

  Para clústeres existentes propiedad de su cuenta de AWS, el campo `CrossAccountClone` indica si el clúster es un clon de un clúster de base de datos propiedad de otra cuenta de AWS. Las entradas con un número de cuenta de AWS diferente en el campo `DBClusterArn` representan clústeres que puede clonar y que son propiedad de otras cuentas de AWS. Estas entradas tienen pocos cambios aparte de `DBClusterArn`. Al crear el clúster clonado, especifique los mismos valores de `StorageEncrypted`, `Engine` y `EngineVersion` que el clúster original.

  El siguiente ejemplo muestra un valor de devolución que demuestra los clústeres clonados tanto reales como potenciales.

```
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0"
```

```
        }
    ]
}
```

Integración de Aurora con otros servicios de AWS

Integre Amazon Aurora con otros servicios de AWS a fin de permitirle ampliar su clúster de base de datos de Aurora para usar funcionalidades adicionales en la nube de AWS.

Temas

- Integración de los servicios de AWS con Amazon Aurora MySQL
- Integración de los servicios de AWS con Amazon Aurora PostgreSQL

Integración de los servicios de AWS con Amazon Aurora MySQL

Amazon Aurora MySQL se integra con otros servicios de AWS con el fin de permitirle ampliar su clúster de base de datos de Aurora MySQL para usar funcionalidades adicionales en la nube de AWS. El clúster de base de datos de Aurora MySQL puede usar los servicios de AWS para hacer lo siguiente:

- Invoque de forma síncrona o asíncrona una función de AWS Lambda mediante las funciones nativas `lambda_sync` o `lambda_async`. O bien invoque de forma asíncrona una función de AWS Lambda con el procedimiento `mysql.lambda_async`.
- Cargar datos desde archivos de texto o XML almacenados en un bucket de Amazon S3 en el clúster de base de datos mediante el comando `LOAD DATA FROM S3` o `LOAD XML FROM S3`.
- Guardar datos en archivos de texto almacenados en un bucket de Amazon S3 desde su clúster de base de datos usando el comando `SELECT INTO OUTFILE S3`.
- Añada o quite de forma automática réplicas de Aurora con Auto Scaling de aplicaciones. Para obtener más información, consulte Amazon Aurora Auto Scaling con réplicas de Aurora.

Para obtener más información acerca de la integración de Aurora MySQL con otros servicios de AWS, consulte Integración de Amazon Aurora MySQL con otros servicios de AWS.

Integración de los servicios de AWS con Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL se integra con otros servicios de AWS con el fin de permitirle ampliar su clúster de base de datos de Aurora PostgreSQL para usar funcionalidades adicionales en la nube de AWS. El clúster de base de datos de Aurora PostgreSQL puede usar los servicios de AWS para hacer lo siguiente:

- Recopilar, ver y evaluar rápidamente el desempeño en las cargas de trabajo de la base de datos relacional con Performance Insights.
- Agregue o quite de forma automática réplicas de Aurora con Amazon EC2 Auto Scaling de Aurora. Para obtener más información, consulte Amazon Aurora Auto Scaling con réplicas de Aurora.

Para obtener más información acerca de la integración de Aurora PostgreSQL con otros servicios de AWS, consulte Integración de Amazon Aurora PostgreSQL con otros servicios de AWS.

Mantenimiento de un clúster de base de datos de Amazon Aurora

Amazon RDS realiza tareas de mantenimiento periódicas en los recursos de Amazon RDS. En los temas siguientes se describen estas acciones de mantenimiento y cómo aplicarlas.

Descripción general de las actualizaciones de mantenimiento de un clúster de base de datos

El mantenimiento suele implicar actualizaciones de los siguientes recursos de su clúster de base de datos:

- Hardware subyacente
- Sistema operativo (SO) subyacente
- Versión del motor de base de datos

Las actualizaciones del sistema operativo suelen deberse a motivos de seguridad. Se recomienda que las haga lo antes posible. Para obtener más información sobre las actualizaciones de sistemas operativos, consulte the section called “Actualizaciones del sistema operativo”.

Temas

- Recursos sin conexión durante las actualizaciones de mantenimiento
- Modificaciones diferidas de instancias de base de datos y clústeres de base de datos
- Consistencia final de la API DescribependingMaintenanceActions

Recursos sin conexión durante las actualizaciones de mantenimiento

Algunos elementos de mantenimiento requieren que Amazon RDS desconecte su clúster de base de datos durante un breve periodo de tiempo. Entre los elementos de mantenimiento que requieren que un recurso esté desconectado están el sistema operativo necesario o la aplicación de parches a la base de datos. Los parches obligatorios que tienen que ver con la seguridad y la fiabilidad de la instancia son los únicos que se programan automáticamente. Estos parches se producen con poca frecuencia, normalmente una vez cada pocos meses. Rara vez se requiere más de una fracción de su período de mantenimiento.

Modificaciones diferidas de instancias de base de datos y clústeres de base de datos

Las modificaciones de instancias y clústeres de base de datos diferidas que haya decidido no aplicar inmediatamente se aplican durante el periodo de mantenimiento. Por ejemplo, puede elegir cambiar clases de instancia de base de datos o grupos de parámetros de clúster o base de datos durante el periodo de mantenimiento. Las modificaciones que especifique mediante la configuración de reinicio pendiente no se muestran en la lista Mantenimiento pendiente. Para obtener más información acerca de la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Para ver las modificaciones pendientes para la siguiente ventana de mantenimiento, utilice el comando describe-db-clústers de la AWS CLI y marque el campo `PendingModifiedValues`.

Consistencia final de la API DescribependingMaintenanceActions

La API `DescribePendingMaintenanceActions` de Amazon RDS sigue un modelo de consistencia final. Esto significa que el resultado del comando `DescribePendingMaintenanceActions` puede que no sea visible inmediatamente para todos los comandos de RDS posteriores. Tenga esto en cuenta cuando utilice `DescribePendingMaintenanceActions` inmediatamente después de usar un comando de API anterior.

La consistencia final puede afectar a la forma en que ha administrado las actualizaciones de mantenimiento. Por ejemplo, si ejecuta el comando `ApplyPendingMaintenanceActions` para actualizar la versión del motor de base de datos de un clúster de base de datos, al final será visible en `DescribePendingMaintenanceActions`. En este escenario, `DescribePendingMaintenanceActions` podría indicar que la acción de mantenimiento no se aplicó a pesar de que sí se hizo.

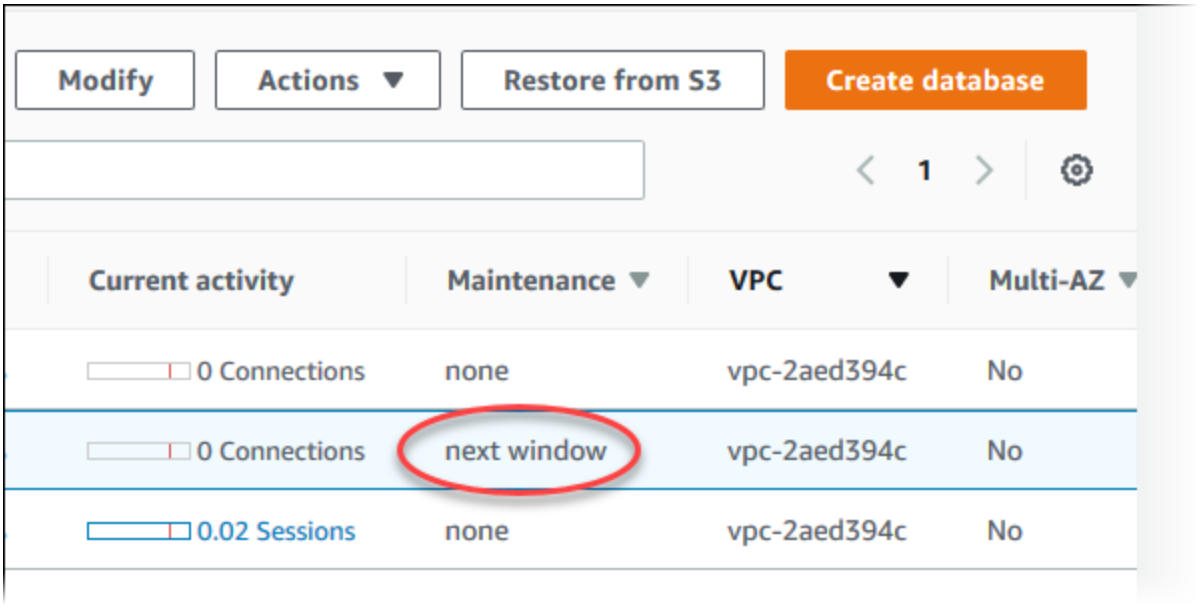
Para administrar la consistencia final, puede hacer lo siguiente:

- Confirme el estado de el clúster de base de datos antes de ejecutar un comando para modificarlo. Ejecute el comando `DescribePendingMaintenanceActions` correspondiente mediante un algoritmo de retroceso exponencial para asegurarse de que dispone de tiempo suficiente para que el comando anterior se propague en el sistema. Para ello, ejecute el comando `DescribePendingMaintenanceActions` varias veces, empezando con un par de segundos de espera y aumentando gradualmente hasta cinco minutos.
- Agregue tiempo de espera entre los comandos siguientes, incluso si un comando `DescribePendingMaintenanceActions` devuelve una respuesta precisa. Aplique un algoritmo

de retroceso exponencial comenzando con un par de segundos de tiempo de espera y aumente gradualmente hasta unos cinco minutos de tiempo de espera.

Visualización de actualizaciones de mantenimiento pendientes

Compruebe si hay disponible una actualización de mantenimiento para un clúster de base de datos, use la consola de RDS, la AWS CLI o la API de RDS. Si hay disponible una actualización, se indicará en la columna Mantenimiento para el clúster de base de datos en la consola Amazon RDS, como se muestra en esta figura.



Si no hay ninguna actualización de mantenimiento disponible para un clúster de base de datos, el valor de columna es none (ninguno).

Si una actualización de mantenimiento está disponible para un clúster, son posibles los siguientes valores de columna:

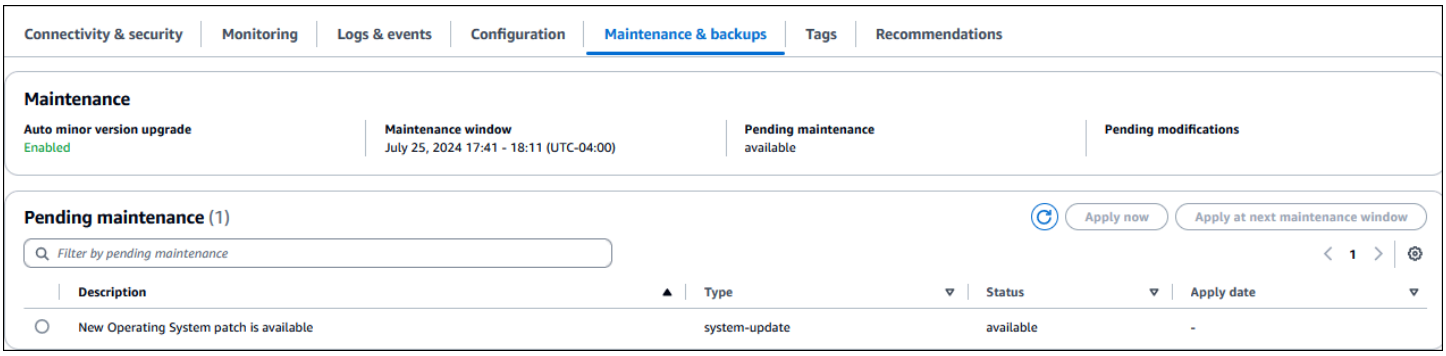
- obligatorio: la acción de mantenimiento se aplicará al recurso y no se podrá aplazar indefinidamente.
- available: la acción de mantenimiento está disponible, pero no se aplicará al recurso automáticamente. Puede aplicarla manualmente.
- siguiente periodo: la acción de mantenimiento se aplicará al recurso durante el siguiente periodo de mantenimiento.
- En curso: la acción de mantenimiento se está aplicando al recurso.

Si hay disponible una actualización, puede realizar una de las acciones siguientes:

- Si el valor de mantenimiento es siguiente periodo, aplace las acciones de mantenimiento eligiendo aplazar actualización en Acciones. No puede aplazar una acción de mantenimiento que ya se haya iniciado.
- Aplicar inmediatamente las operaciones de mantenimiento.
- Aplicar las acciones de mantenimiento durante el siguiente periodo de mantenimiento.
- No realizar ninguna acción.

Realización de una acción mediante la Consola de administración de AWS

1. Seleccione la instancia de base de datos o el clúster para mostrar sus detalles.
2. Seleccione Mantenimiento y copias de seguridad. Aparecerán las acciones de mantenimiento pendientes.
3. Elija la acción que desee realizar y seleccione cuándo aplicarla.



El periodo de mantenimiento determina el momento en que comienzan las operaciones pendientes, pero no limita su tiempo de ejecución total. No existen garantías de que las operaciones de mantenimiento finalicen antes de que termine el periodo de mantenimiento, de modo que pueden continuar más allá de la hora de finalización establecida. Para obtener más información, consulte Ventana de mantenimiento de Amazon RDS.

Para ver si hay disponible una actualización de mantenimiento para un clúster de base de datos, puede ejecutar el comando describe-pending-maintenance-actions de la AWS CLI.

Para obtener información sobre la aplicación de actualizaciones de mantenimiento, consulte Aplicación de actualizaciones a un clúster de base de datos.

Acciones de mantenimiento para Amazon Aurora

Las acciones de mantenimiento siguientes se aplican a los clústeres de bases de datos de Aurora:

- `os-upgrade`: actualización de los sistemas operativos de todas las instancias de base de datos del clúster de bases de datos mediante actualizaciones sucesivas. Para obtener más información, consulte the section called “Actualizaciones del sistema operativo”.
- `system-update`: aplicación de parches al motor de base de datos para Aurora PostgreSQL.

Las acciones de mantenimiento siguientes se aplican a las instancias de base de datos de Aurora:

- `ca-certificate-rotation`: actualización del certificado de Amazon RDS Certificate Authority para la instancia de base de datos.
- `hardware-maintenance`: realización del mantenimiento del hardware subyacente de la instancia de base de datos.
- `system-update`: actualización del sistema operativo de la instancia de base de datos.

Selección de frecuencia de actualizaciones de mantenimiento de Aurora MySQL

Puede controlar si las actualizaciones de Aurora MySQL ocurren con frecuencia o rara vez para cada clúster de base de datos. La mejor opción depende del uso de Aurora MySQL y de las prioridades de las aplicaciones que se ejecuten en Aurora. Para obtener información acerca de las versiones de estabilidad a largo plazo (LTS) de Aurora MySQL que requieren actualizaciones menos frecuentes, consulte Versiones de soporte a largo plazo (LTS) de Aurora MySQL.

Podría elegir actualizar un clúster de Aurora MySQL rara vez si se aplican algunas o todas las condiciones siguientes:

- El ciclo de prueba de su aplicación tarda mucho tiempo para cada actualización al motor de base de datos de Aurora MySQL.
- Tiene muchos clústeres de base de datos o muchas aplicaciones ejecutándose en la misma versión de Aurora MySQL. Prefiere actualizar todos sus clústeres de base de datos y aplicaciones asociadas al mismo tiempo.
- Se utilizan Aurora MySQL y RDS para MySQL. Prefiere mantener los clústeres de Autora MySQL y las instancias de base de datos de RDS àra MySQL compatibles con el mismo nivel de MySQL.

- Su aplicación de Aurora MySQL está en producción o bien es crítica para la empresa. No puede permitirse períodos de inactividad para actualizaciones fuera de los raros casos para parches críticos.
- Su aplicación de Aurora MySQL no está limitada por problemas de rendimiento o falta de características que se resuelven en versiones siguientes de Aurora MySQL.

Si los factores anteriores son aplicables a su caso, puede limitar el número de actualizaciones forzadas para un clúster de base de datos de Aurora MySQL. Lo hace eligiendo una versión de Aurora MySQL específica conocida como versión de «Soporte a largo plazo» (LTS) al crear o actualizar dicho clúster de base de datos. Al hacerlo se minimiza el número de ciclos de actualización, ciclos de prueba e interrupciones relacionadas con actualizaciones para dicho clúster de base de datos.

Podría elegir actualizar un clúster de Aurora MySQL frecuentemente si se aplican algunas o todas las condiciones siguientes:

- El ciclo de prueba de la aplicación es sencillo y breve.
- La aplicación sigue en la fase de desarrollo.
- El entorno de la base de datos usa diversas versiones de Aurora MySQL o Aurora MySQL y versiones de RDS para MySQL. Cada clúster de Aurora MySQL tiene su propio ciclo de actualización.
- Espera mejoras de características o rendimiento específico antes de aumentar el uso de Aurora MySQL.

Si los factores anteriores son aplicables a su situación, puede habilitar Aurora para aplicar actualizaciones importantes con mayor frecuencia. Para ello, actualice un clúster de base de datos de Aurora MySQL a una versión de Aurora MySQL de más reciente que la versión de LTS. Al hacerlo las últimas mejoras de rendimiento, correcciones de errores y características disponibles están disponibles para usted más rápidamente.

Ventana de mantenimiento de Amazon RDS

La ventana de mantenimiento es un intervalo de tiempo semanal durante la que se aplican los cambios del sistema. Cada clúster de base de datos tiene un periodo de mantenimiento semanal. El periodo de mantenimiento es una oportunidad para controlar cuándo ocurrirán las modificaciones y

los parches de software. Para obtener más información sobre el ajuste del periodo de mantenimiento, consulte Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos.

RDS consume algunos de los recursos de su clúster de base de datos mientras se aplica el mantenimiento. Es posible que observe un efecto mínimo en el desempeño. Para una instancia de base de datos, en raras ocasiones puede ser necesaria una conmutación por error Multi-AZ para que se complete una actualización de mantenimiento.

Si hay un evento de mantenimiento programado para una semana determinada, se iniciará durante la ventana de mantenimiento que identifique. La mayoría de los eventos de mantenimiento también se completan durante la ventana de mantenimiento de 30 minutos, aunque otros eventos de mantenimiento pueden tardar más de 30 minutos en completarse. El periodo de mantenimiento se detiene cuando se detiene el clúster de la base de datos.

La ventana de mantenimiento de 30 minutos se selecciona al azar dentro de un bloque de 8 horas por región. Si no especifica una ventana de mantenimiento al crear un clúster de base de datos, RDS asigna una ventana de mantenimiento de 30 minutos un día de la semana seleccionado al azar.

En la siguiente tabla se muestran los bloques de tiempo de cada Región de AWS desde los que se asignan las ventanas de mantenimiento predeterminadas.

Nombre de la región	Región	Bloque de tiempo
Este de EE. UU. (Norte de Virginia)	us-east-1	03:00–11:00 UTC
Este de EE. UU. (Ohio)	us-east-2	03:00 — 11:00 UTC
Oeste de EE. UU. (Norte de California)	us-west-1	06:00 — 14:00 UTC
Oeste de EE. UU. (Oregón)	us-west-2	06:00–14:00 UTC
África (Ciudad del Cabo)	af-south-1	03:00–11:00 UTC
Asia-Pacífico (Hong Kong)	ap-east-1	06:00-14:00 UTC

Nombre de la región	Región	Bloque de tiempo
Asia-Pacífico (Hyderabad)	ap-south-2	06:30 – 14:30 UTC
Asia-Pacífico (Yakarta)	ap-southeast-3	08:00 a 16:00 h UTC
Asia-Pacífico (Malasia)	ap-southeast-5	09:00–17:00 UTC
Asia-Pacífico (Melbourne)	ap-southeast-4	11:00–19:00 UTC
Asia Pacífico (Bombay)	ap-south-1	06:00-14:00 UTC
Asia-Pacífico (Nueva Zelanda)	ap-southeast-6	13:00-21:00 UTC
Asia Pacífico (Osaka)	ap-northeast-3	22:00 — 23:59 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00 — 21:00 UTC
Asia-Pacífico (Singapur)	ap-southeast-1	14:00 — 22:00 UTC
Asia Pacífico (Sídney)	ap-southeast-2	12:00 — 20:00 UTC
Asia-Pacífico (Taipéi)	ap-east-2	09:00-17:00 UTC
Asia-Pacífico (Tailandia)	ap-southeast-7	08:00-16:00 UTC
Asia-Pacífico (Tokio)	ap-northeast-1	13:00 — 21:00 UTC
Canadá (centro)	ca-central-1	03:00-11:00 UTC

Nombre de la región	Región	Bloque de tiempo
Oeste de Canadá (Calgary)	ca-west-1	18:00 — 02:00 UTC
China (Pekín)	cn-north-1	06:00–14:00 UTC
China (Ningxia)	cn-northwest-1	06:00–14:00 UTC
Europe (Fráncfort)	eu-central-1	21:00 — 05:00 UTC
Europe (Irlanda)	eu-west-1	22:00 — 06:00 UTC
Europe (Londres)	eu-west-2	22:00-06:00 UTC
Europa (Milán)	eu-south-1	02:00 — 10:00 UTC
Europa (París)	eu-west-3	23:59–07:29 UTC
Europa (España)	eu-south-2	02:00 — 10:00 UTC
Europa (Estocolmo)	eu-north-1	23:00 — 07:00 UTC
Europa (Zúrich)	eu-central-2	02:00 — 10:00 UTC
Israel (Tel Aviv)	il-central-1	03:00-11:00 UTC
México (central)	mx-central-1	19:00-03:00 UTC
Medio Oriente (Baréin)	me-south-1	06:00-14:00 UTC
Medio Oriente (EAU)	me-central-1	05:00 a 13:00 h UTC
América del Sur (São Paulo)	sa-east-1	00:00–08:00 UTC
AWS GovCloud (EE. UU. Este)	us-gov-east-1	17:00 — 01:00 UTC

Nombre de la región	Región	Bloque de tiempo
AWS GovCloud (Oeste de EE. UU.)	us-gov-west-1	06:00–14:00 UTC

Temas

- Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos

Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos

La ventana de mantenimiento del clúster de base de datos de Aurora debe corresponder al momento de mínimo uso y, por tanto, podría ser preciso modificarla cada cierto tiempo. El clúster de base de datos solo deja de estar disponible durante este tiempo si las actualizaciones que se están aplicando requieren una interrupción. La interrupción dura la cantidad de tiempo mínima requerida para realizar las actualizaciones necesarias.

> **Note**
>
> En el caso de las actualizaciones del motor de base de datos, Amazon Aurora administra la ventana de mantenimiento preferida para un clúster de base de datos y no para las instancias individuales.

Consola

Para ajustar la ventana de mantenimiento preferida del clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos cuyo periodo de mantenimiento desea cambiar.
4. Elija Modify.
5. En la sección Maintenance (Mantenimiento), actualice el periodo de mantenimiento.
6. Elija Continue.

   En la página de confirmación, revise los cambios.

7. Para aplicar los cambios a la ventana de mantenimiento inmediatamente, elija Immediately (Inmediatamente) en la sección Schedule of modifications (Programación de modificaciones).
8. Seleccione Modify cluster (Modificar clúster) para guardar los cambios.

   O bien, elija Back para editar los cambios o Cancel para cancelarlos.

AWS CLI

Para ajustar la ventana de mantenimiento preferida del clúster de base de datos, use el comando AWS CLI de la `modify-db-cluster` con los siguientes parámetros:

- `--db-cluster-identifier`
- `--preferred-maintenance-window`

Example

En el siguiente ejemplo de código, el periodo de mantenimiento se define para los martes de 4:00 a 4:30 AM UTC.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier my-cluster \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Para Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier my-cluster ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API de RDS

Para ajustar el periodo de mantenimiento preferida del clúster de base de datos, use la operación `ModifyDBCluster` de la API de Amazon RDS con los siguientes parámetros:

- `DBClusterIdentifier`
- `PreferredMaintenanceWindow`

Aplicación de actualizaciones a un clúster de base de datos

Con Amazon RDS puede elegir el momento en que desea aplicar las operaciones de mantenimiento. Puede indicar cuándo Amazon RDS debe aplicar las actualizaciones usando la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para administrar la actualización de un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el clúster de base de datos que tenga una actualización necesaria.
4. En Acciones, elija una de las siguientes opciones:

   - Aplicar parches ahora
   - Aplicar parches en el siguiente periodo

   > **Note**
   >
   > Si elige Aplicar parches en el siguiente periodo y después desea aplazar la actualización, puede seleccionar Aplazar actualización. No puede aplazar una acción de mantenimiento si ya se ha iniciado.
   > Para cancelar una acción de mantenimiento, modifique la instancia de base de datos y deshabilite la Auto minor version upgrade (Actualización automática de versiones secundarias).

AWS CLI

Para aplicar una actualización pendiente a un clúster de base de datos, use el comando apply-pending-maintenance-action de la AWS CLI.

Example

Para Linux, macOS o Unix:

```
aws rds apply-pending-maintenance-action \
```

```
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
    --apply-action system-update \
    --opt-in-type immediate
```

En Windows:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> Para aplazar una acción de mantenimiento, especifique `undo-opt-in` para `--opt-in-type`. No se puede especificar `undo-opt-in` para `--opt-in-type` si la acción de mantenimiento ya se ha iniciado.
> Para cancelar una acción de mantenimiento, ejecute el comando de la AWS CLI modify-db-instance y especifique `--no-auto-minor-version-upgrade`.

Para obtener una lista de los recursos con al menos una actualización pendiente, use el comando describe-pending-maintenance-actions de la AWS CLI.

Example

Para Linux, macOS o Unix:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Para Windows:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

También puede devolver una lista de recursos de un clúster de base de datos mediante la especificación del parámetro `--filters` del comando `describe-pending-maintenance-actions` de la AWS CLI. El formato del comando `--filters` es `Name=`*`filter-name`*`,Value=`*`resource-id`*`,...`.

Los valores aceptados para el parámetro `Name` de un filtro son los siguientes:

- `db-instance-id`: acepta una lista de identificadores o nombres de recurso de Amazon (ARN) de instancias de base de datos. La lista obtenida solo incluirá las operaciones de mantenimiento pendientes para las instancias de base de datos referidas por esos identificadores o ARN.
- `db-cluster-id`: acepta una lista de identificadores o ARN de clústeres de base de datos para Amazon Aurora. La lista obtenida solo incluirá las operaciones de mantenimiento pendientes para los clústeres de base de datos referidos por esos identificadores o ARN.

Por ejemplo, en el ejemplo siguiente se obtienen las operaciones de mantenimiento pendientes para los clústeres de base de datos `sample-cluster1` y `sample-cluster2`.

Example

Para Linux, macOS o Unix:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Para Windows:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

API de RDS

Para aplicar una actualización a un clúster de base de datos, llame a la operación `ApplyPendingMaintenanceAction` de la API de Amazon RDS.

Para obtener una lista de los recursos con al menos una actualización pendiente, llame a la operación `DescribePendingMaintenanceActions` de la API Amazon RDS.

Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora

La configuración de Actualización automática de la versión secundaria especifica si Aurora aplica automáticamente las actualizaciones al clúster de base de datos. Estas actualizaciones incluyen nuevas versiones secundarias con características adicionales y revisiones que contienen correcciones de errores.

Las actualizaciones automáticas de las versiones secundarias actualizan periódicamente la base de datos a versiones recientes del motor de base de datos. Sin embargo, es posible que la actualización no siempre incluya la versión más reciente del motor de base de datos. Si necesita mantener las bases de datos en versiones específicas en momentos determinados, le recomendamos que las actualice manualmente a las versiones de base de datos que necesite según el calendario requerido. En caso de problemas de seguridad críticos o cuando una versión alcance su fecha de fin de soporte, Amazon Aurora podría aplicar una actualización de la versión secundaria aunque no haya habilitado la opción Actualización automática de la versión secundaria. Para obtener más información, consulte la documentación de actualización para el motor de base de datos específico.

Consulte Actualización de la versión secundaria o el nivel de parche de un clúster de bases de datos Aurora MySQL y Actualización a una versión secundaria.

> **Note**
>
> La base de datos global de Aurora no admite actualizaciones de versiones secundarias automáticas.

Esta función se activa de forma predeterminada. Para cada clúster de base de datos nuevo, elija el valor adecuado para esta configuración. Este valor se basa en su importancia, duración prevista y la cantidad de pruebas de verificación que realice después de cada actualización.

Para obtener instrucciones sobre cómo activar o desactivar la configuración de Actualización automática de versiones secundarias, consulte lo siguiente:

- Activación de las actualizaciones de versiones secundarias automáticas para un clúster de base de datos de Aurora
- Activación de las actualizaciones de versiones secundarias automáticas para instancias de base de datos individuales en un clúster de base de datos de Aurora

> **Important**
>
> Recomendamos encarecidamente que, para los clústeres de bases de datos nuevos y existentes, aplique esta configuración al clúster de base de datos y no a las instancias de base de datos del clúster de forma individual. Si alguna instancia de base de datos del

clúster tiene esta configuración desactivada, el clúster de base de datos no se actualiza automáticamente.

La siguiente tabla muestra cómo funciona la configuración de Actualización automática de versiones secundarias cuando se aplica a los niveles de clúster e instancia.

Acción	Configuración del clúster	Configuraciones de la instancia	¿El clúster se ha actualizado automáticamente?
Se establece en True en el clúster de base de datos.	True	True para todas las instancias nuevas y existentes	Sí
Se establece en False en el clúster de base de datos.	False	False para todas las instancias nuevas y existentes	No
Se ha establecido previamente en True en el clúster de base de datos. Lo ha establecido en False en al menos una instancia de base de datos.	Cambia a False	False para una o varias instancias	No
Se ha establecido previamente en False en el clúster de base de datos. Lo ha establecido en True en al menos una instancia de base	False	True para una o más instancias, pero no para todas las instancias	No

Acción	Configuración del clúster	Configuraciones de la instancia	¿El clúster se ha actualizado automáticamente?
de datos, pero no en todas las instancias.			
Se ha establecido previamente en False en el clúster de base de datos. Lo ha establecido en True en todas las instancias de base de datos.	Cambia a True	True para todas las instancias	Sí

Las actualizaciones automáticas de versiones secundarias se comunican de antemano a través de un evento de clúster de base de datos de Amazon RDS con una categoría `maintenance` e ID de `RDS-EVENT-0156`. Para obtener más información, consulte Categorías y mensajes de eventos de Amazon RDS para Aurora.

La actualización automática se produce durante el período de mantenimiento. Si las instancias de base de datos individuales del clúster de base de datos tienen períodos de mantenimiento diferentes a las de la ventana de mantenimiento del clúster, la ventana de mantenimiento del clúster tiene prioridad.

Para obtener más información acerca de las actualizaciones de motor de Aurora PostgreSQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL.

Para obtener más información acerca de la configuración de Auto minor version upgrade (Actualización automática de la versión secundaria) para Aurora MySQL, consulte Activación de actualizaciones automáticas entre versiones secundarias de Aurora MySQL. Para obtener más información general acerca de las actualizaciones del motor de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

Activación de las actualizaciones de versiones secundarias automáticas para un clúster de base de datos de Aurora

Siga el procedimiento general de Modificación del clúster de base de datos con la consola, CLI y API.

Consola

En la página Modificar el clúster de base de datos, en la sección Mantenimiento, seleccione la casilla de verificación Habilitar actualización automática de versiones secundarias.

AWS CLI

Ejecute el comando de la AWS CLI modify-db-clúster. Especifique el nombre del clúster de base de datos para la opción `--db-cluster-identifier` y `true` para la opción `--auto-minor-version-upgrade`. Si lo desea, especifique la opción `--apply-immediately` para habilitar inmediatamente esta configuración para el clúster de base de datos.

API de RDS

Llame a la operación ModifyDBclúster de la API y especifique el nombre del clúster de base de datos para el parámetro `DBClusterIdentifier` y `true` para el parámetro `AutoMinorVersionUpgrade`. Como opción, defina el parámetro `ApplyImmediately` en `true` para activar inmediatamente esta configuración para el clúster de base de datos.

Activación de las actualizaciones de versiones secundarias automáticas para instancias de base de datos individuales en un clúster de base de datos de Aurora

Siga el procedimiento general de Modificación de una instancia de base de datos en un clúster de base de datos.

Consola

En la página Modificar la instancia de base de datos, en la sección Mantenimiento, seleccione la casilla de verificación Habilitar actualización automática de versiones secundarias.

AWS CLI

Ejecute el comando de la AWS CLI modify-db-instance. Especifique el nombre de la instancia de base de datos para la opción `--db-instance-identifier` y `true` para la opción `--auto-minor-version-upgrade`. Si lo desea, especifique la opción `--apply-immediately` para habilitar inmediatamente esta configuración para su instancia de base de datos. Ejecute

un comando `modify-db-instance` independiente para cada instancia de base de datos del clúster.

API de RDS

Llame a la operación ModifyDBInstance de la API y especifique el nombre del clúster de base de datos para el parámetro `DBInstanceIdentifier` y `true` para el parámetro `AutoMinorVersionUpgrade`. Como opción, defina el parámetro `ApplyImmediately` en `true` para activar inmediatamente esta configuración para la instancia de base de datos. Llame a una acción `ModifyDBInstance` independiente para cada instancia de base de datos del clúster.

Puede utilizar un comando de la CLI como el siguiente para comprobar el estado de la configuración `AutoMinorVersionUpgrade` para todas las instancias de base de datos de los clústeres de Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

El resultado de este comando debería ser similar al siguiente:

```
[
  {
      "DBInstanceIdentifier": "db-writer-instance",
      "DBClusterIdentifier": "my-db-cluster-57",
      "AutoMinorVersionUpgrade": true
  },
  {
      "DBInstanceIdentifier": "db-reader-instance1",
      "DBClusterIdentifier": "my-db-cluster-57",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "db-writer-instance2",
      "DBClusterIdentifier": "my-db-cluster-80",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

En este ejemplo, Habilitar actualización automática de versiones secundarias está desactivado para el clúster de base de datos `my-db-cluster-57`, porque está desactivado para una de las instancias de base de datos del clúster.

Actualizaciones del sistema operativo para clústeres de base de datos de Aurora

En ocasiones, las instancias de base de datos de los clústeres de base de datos de Aurora MySQL y Aurora PostgreSQL requieren actualizaciones del sistema operativo. Amazon RDS actualiza el sistema operativo a una versión más reciente para mejorar el rendimiento de la base de datos y la posición de seguridad general de los clientes. Normalmente, las actualizaciones tardan unos 10 minutos en completarse. Las actualizaciones del sistema operativo no cambian la versión del motor de la base de datos ni la clase de instancia de la base de datos.

Hay dos tipos de actualizaciones del sistema operativo, que se diferencian por la descripción que aparece en la acción de mantenimiento pendiente:

- Actualización de la distribución del sistema operativo: se utiliza para migrar a la última versión principal compatible de Amazon Linux. Su descripción es `New Operating System upgrade is available`.
- Revisión del sistema operativo: se usa para aplicar varias correcciones de seguridad y, a veces, para mejorar el rendimiento de la base de datos. Su descripción es `New Operating System patch is available`.

Las actualizaciones del sistema operativo pueden ser opcionales u obligatorias:

- Se puede aplicar una actualización opcional en cualquier momento. Aunque estas actualizaciones son opcionales, le recomendamos que las aplique periódicamente para mantener su flota de RDS al día. RDS no aplica estas actualizaciones automáticamente.

  Para recibir una notificación cuando haya un nuevo parche del sistema operativo opcional disponible, puede suscribirse al RDS-EVENT-0230 en la categoría de eventos de parches de seguridad. Para obtener información sobre cómo suscribirse a los eventos de RDS, consulte Suscripción a notificaciones de eventos de Amazon RDS.

> **Note**
>
> `RDS-EVENT-0230` no se aplica a las actualizaciones de distribución del sistema operativo.

- Una actualización obligatoria debe aplicarse, por lo que enviamos una notificación antes de la actualización obligatoria. La notificación puede incluir una fecha de vencimiento. Planifique la actualización antes de esta fecha de vencimiento. Después de la fecha de vencimiento especificada, Amazon RDS actualiza automáticamente el sistema operativo de la instancia de base de datos a la versión más reciente durante uno de los períodos de mantenimiento asignados.

  Las actualizaciones de la distribución del sistema operativo son obligatorias.

> **Note**
>
> Mantenerse al día en todas las actualizaciones opcionales y obligatorias podría ser necesario para cumplir varias obligaciones de conformidad. Le recomendamos que aplique todas las actualizaciones que RDS pone a disposición de los usuarios de forma rutinaria durante los periodos de mantenimiento.

Para los clústeres de base de datos de Aurora, puede utilizar la opción de mantenimiento del clúster para realizar actualizaciones del sistema operativo (SO). Busque la opción de realizar actualizaciones de clúster en la pestaña Mantenimiento y copias de seguridad cuando seleccione el nombre del clúster de base de datos en la consola o utilice el comando `os-upgrade` en la AWS CLI. Este método conserva la disponibilidad de lectura con actualizaciones sucesivas que aplican automáticamente las actualizaciones a unas cuantas instancias de bases de datos de lector a la vez. Para evitar múltiples conmutaciones por error y reducir los tiempos de inactividad innecesarios, Aurora actualiza la instancia de base de datos de escritor en último lugar.

Las actualizaciones del sistema operativo del clúster se producen durante el periodo de mantenimiento que especifique para el clúster. Esto garantiza las actualizaciones coordinadas en todo el clúster.

Para garantizar la compatibilidad con versiones anteriores, Aurora también mantiene la opción de mantenimiento de instancia. Sin embargo, le recomendamos que utilice actualizaciones de clúster en su lugar. Si debe usar actualizaciones de instancia, actualice primero las instancias de base de datos de lector en un clúster de base de datos y, a continuación, la instancia de base de datos de

escritor. Si actualiza las instancias de lector y escritor simultáneamente, aumenta la probabilidad de que se produzca un tiempo de inactividad relacionado con la conmutación por error. Busque la opción de realizar actualizaciones de instancia en la pestaña Mantenimiento y copias de seguridad al seleccionar el nombre de la instancia de base de datos en la consola o utilice el comando `system-update` en la AWS CLI.

Las actualizaciones del sistema operativo de instancia se producen durante el periodo de mantenimiento que especificó para cada instancia correspondiente. Por ejemplo, si un clúster y dos instancias de lector tienen periodos de mantenimiento diferentes, una actualización del sistema operativo en el nivel del clúster se alinea con el periodo de mantenimiento del clúster.

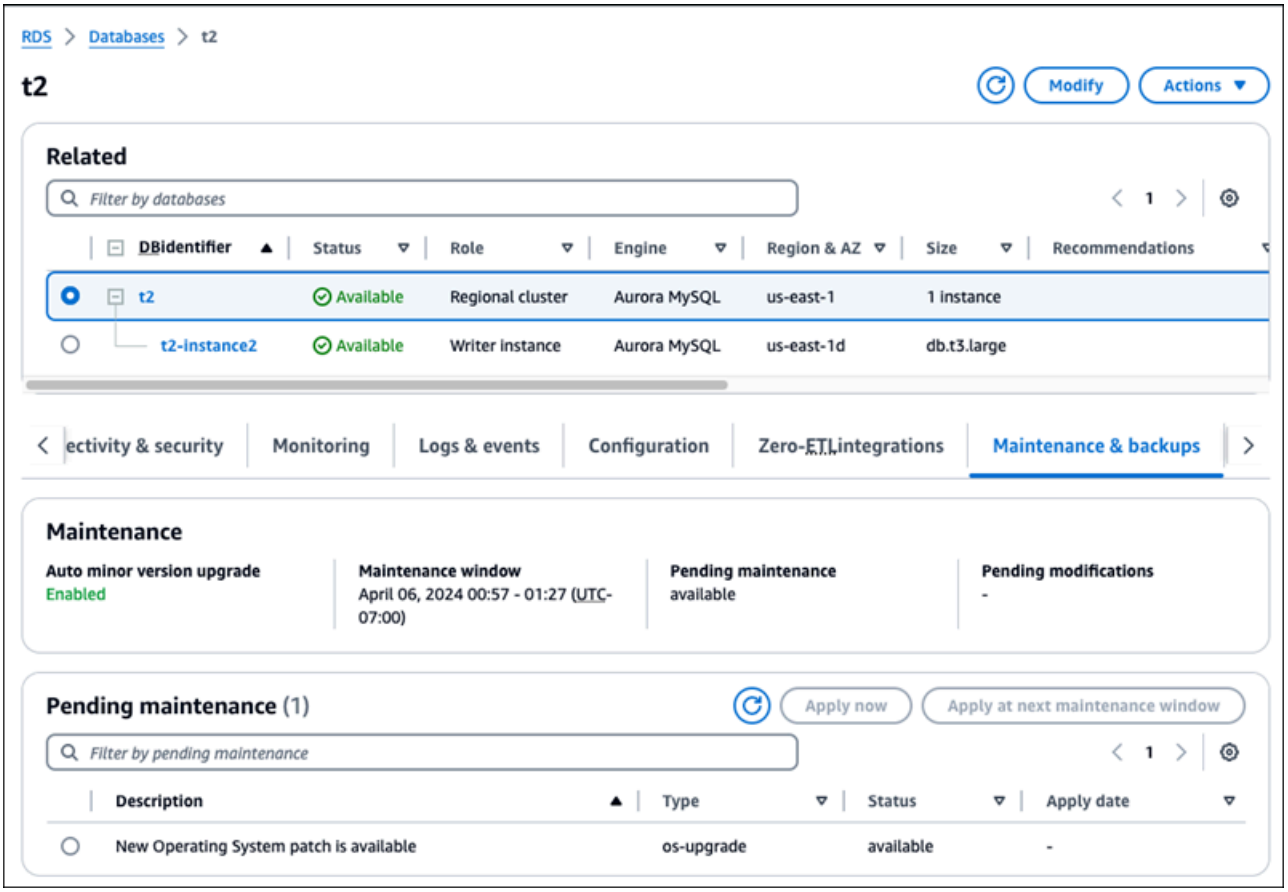
Puede utilizar la Consola de administración de AWS o la AWS CLI para obtener información sobre el tipo de actualización del sistema operativo.

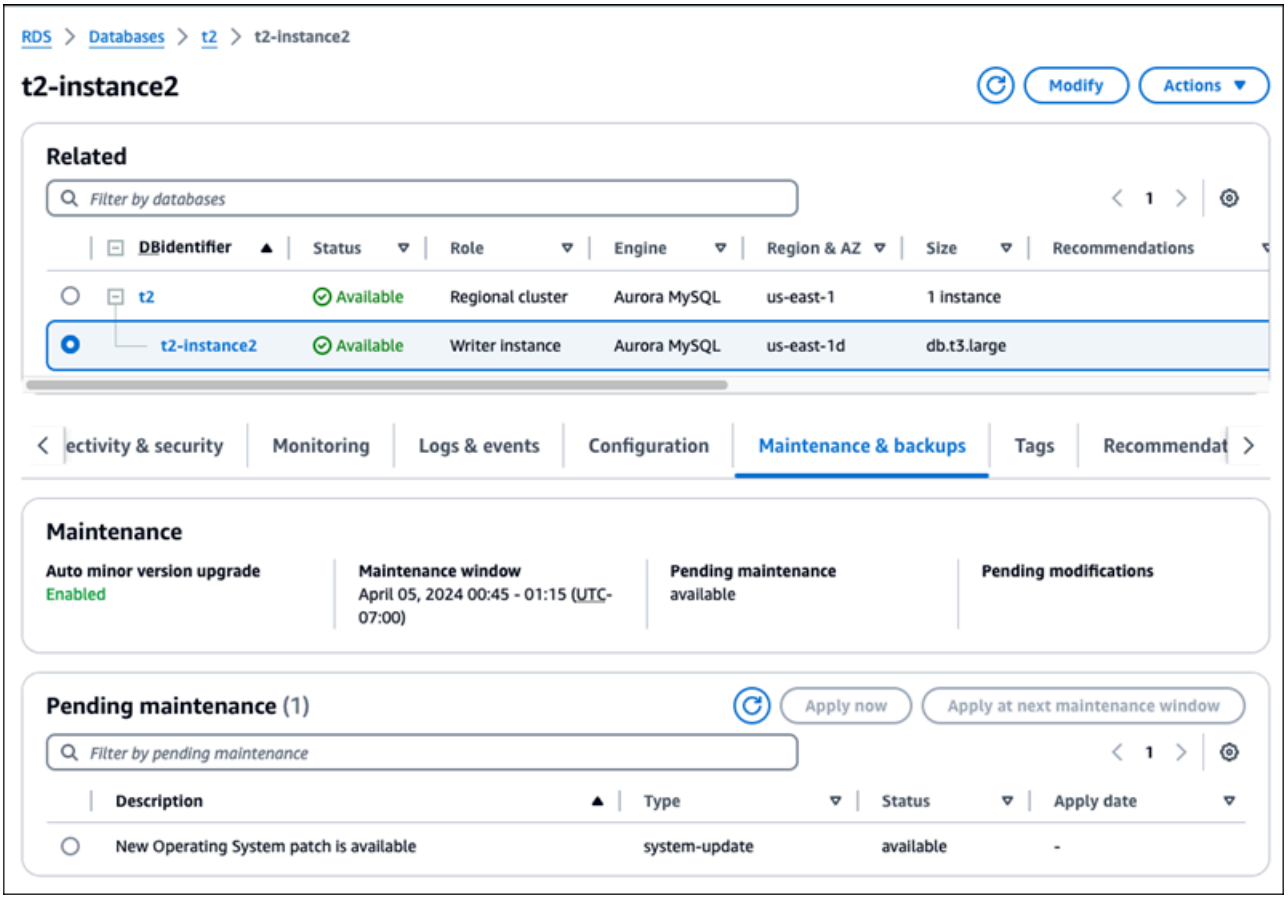
Consola

Para obtener información de actualización mediante la Consola de administración de AWS

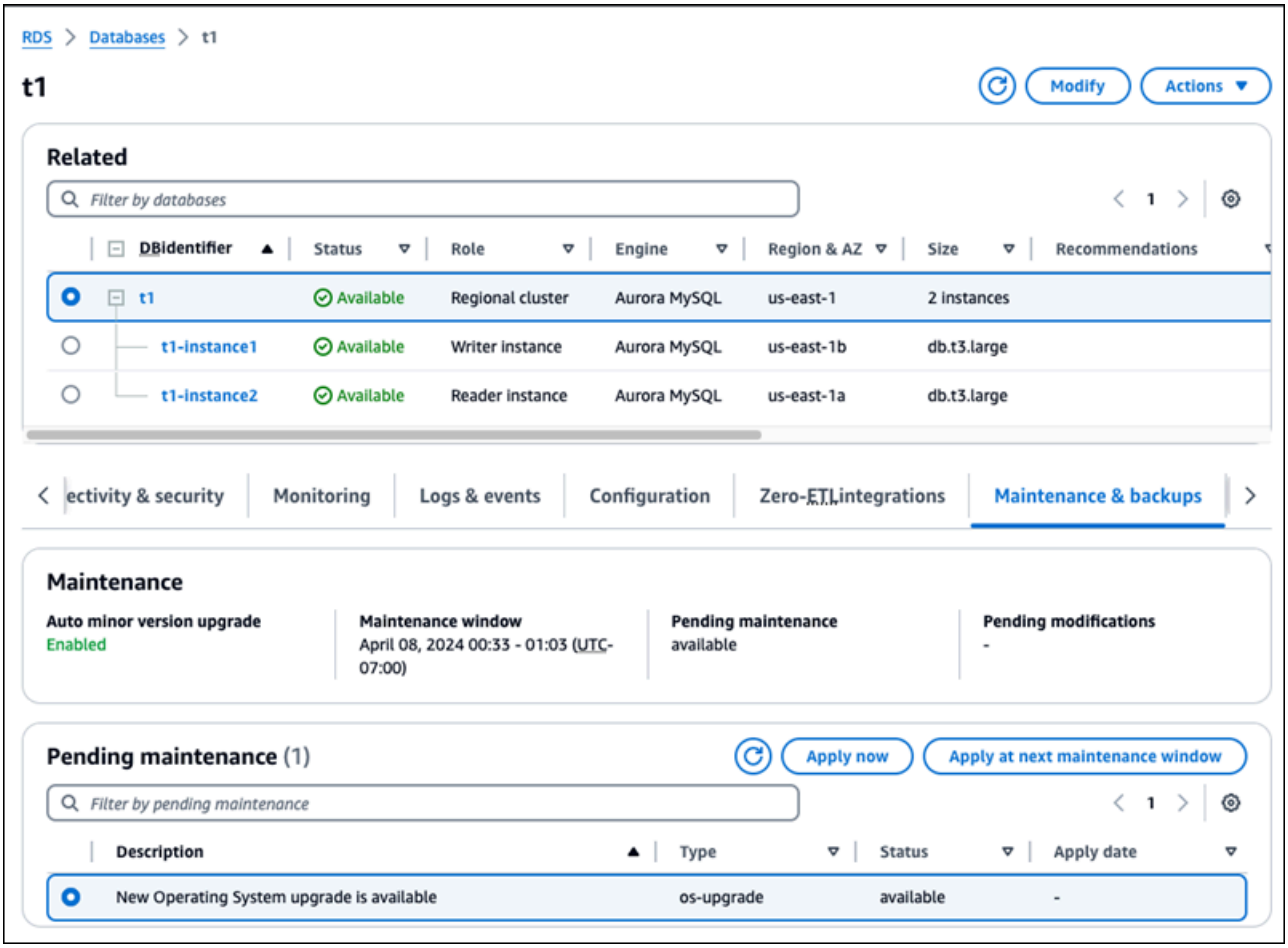
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos.
3. Seleccione Mantenimiento y copias de seguridad.
4. En la sección Mantenimiento pendiente, busque la actualización del sistema operativo y consulte el valor Descripción.

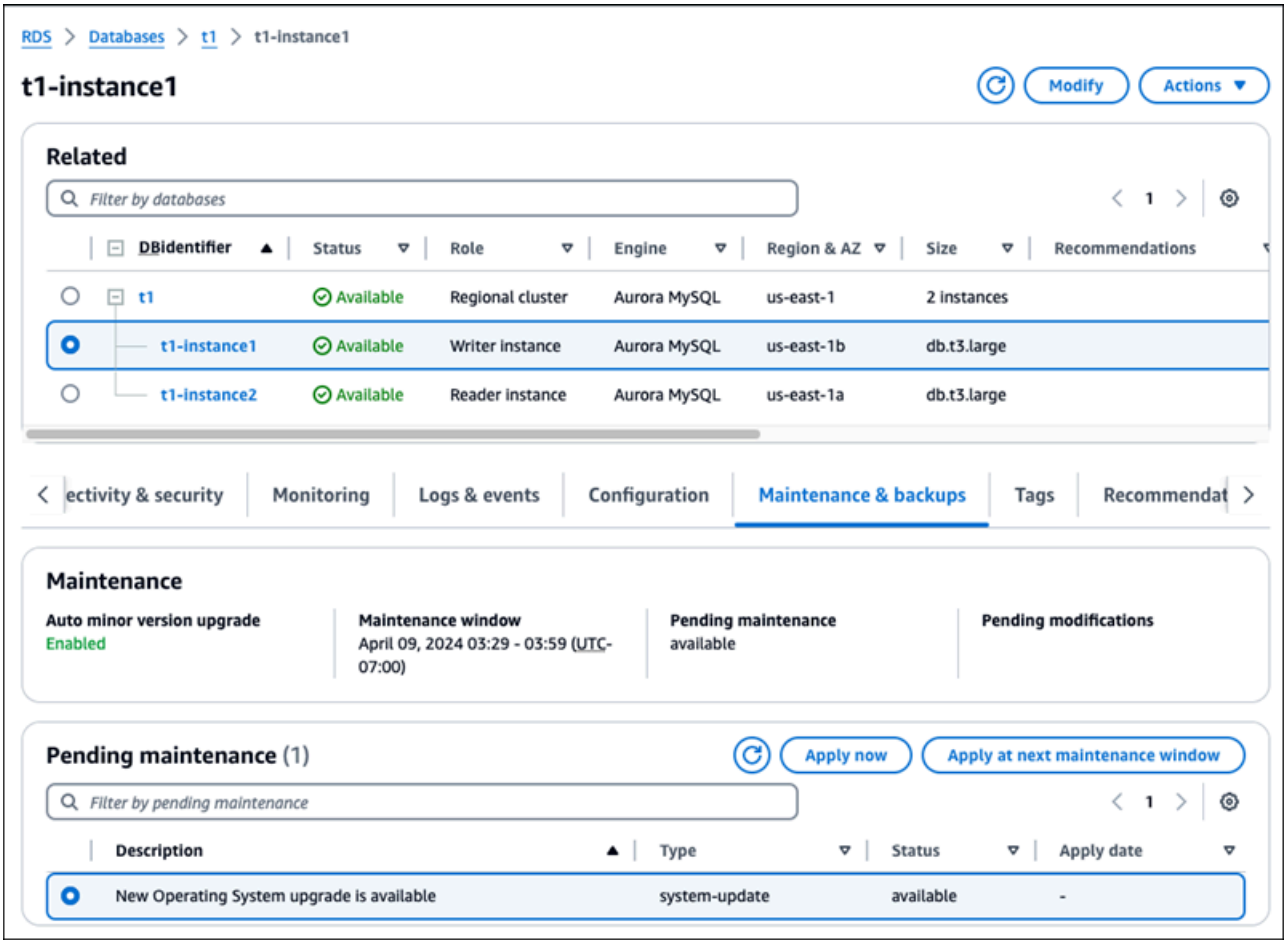
En las imágenes siguientes, se muestra un clúster de bases de datos con una instancia de base de datos de escritor que tiene disponible un parche para el sistema operativo.

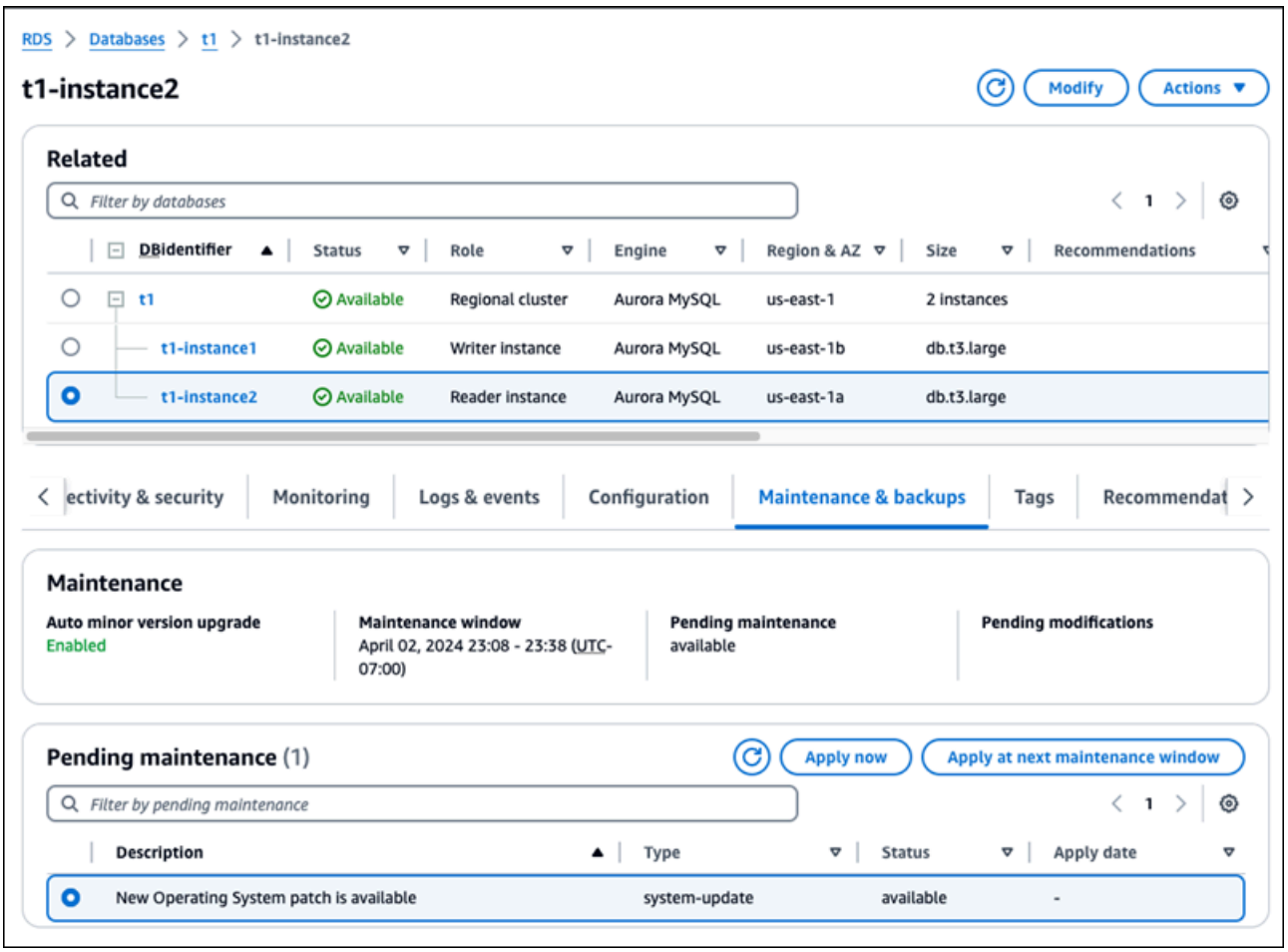

RDS > Databases > t2
t2
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t2
Available
Regional cluster
Aurora MySQL
us-east-1
1 instance
t2-instance2
Available
Writer instance
Aurora MySQL
us-east-1d
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 06, 2024 00:57 - 01:27 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System patch is available
os-upgrade
available
-



En la siguiente imagen, se muestra un clúster de bases de datos con una instancia de base de datos de escritor y una instancia de base de datos de lector. La instancia de escritor tiene disponible una actualización obligatoria del sistema operativo. La instancia de lector tiene disponible un parche del sistema operativo.


RDS > Databases > t1
t1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 08, 2024 00:33 - 01:03 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
os-upgrade
available
-


RDS > Databases > t1 > t1-instance1
t1-instance1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
Recommendat
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 09, 2024 03:29 - 03:59 (UTC-07:00)
Pending maintenance
available
Pending modifications
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
system-update
available
-



AWS CLI

Para obtener información de actualización de la AWS CLI, use el comando describe-pending-maintenance-actions.

```
aws rds describe-pending-maintenance-actions
```

La siguiente salida muestra una actualización de la distribución del sistema operativo para un clúster de base de datos y una instancia de base de datos.

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:cluster:t3",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "os-upgrade",
          "Description": "New Operating System upgrade is available"
```

```
        }
      ]
    },
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:t3-instance1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "Description": "New Operating System upgrade is available"
        }
      ]
    },
  ]
}
```

La siguiente salida muestra un parche del sistema operativo para una instancia de la base de datos.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System patch is available"
    }
  ]
}
```

Disponibilidad de las actualizaciones del sistema operativo

Las actualizaciones del sistema operativo son específicas para la versión del motor de base de datos y la clase de instancia de base de datos. Por lo tanto, las instancias de base de datos reciben o requieren actualizaciones en diferentes momentos. Cuando una actualización del sistema operativo está disponible para su instancia de base de datos en función de su versión del motor y de la clase de instancia, la actualización aparece en la consola. También puede verse ejecutando el comando describe-pending-maintenance-actions de la AWS CLI o llamando a la operación de la API de RDS DescribePendingMaintenanceActions. Si hay una actualización disponible para su instancia, puede actualizar el sistema operativo siguiendo las instrucciones de Aplicación de actualizaciones a un clúster de base de datos.

Uso de la política de implementación de actualizaciones de AWS Organizations para actualizaciones de versiones secundarias automáticas

Aurora admite la política de implementación de actualizaciones de AWS Organizations para administrar las actualizaciones automáticas de versiones secundarias en varios recursos de bases de datos y Cuentas de AWS. Esta política le ayuda a implementar una estrategia de actualización controlada para los clústeres mediante:

Cómo funciona la política de implementación de actualizaciones

Cuando una nueva versión secundaria del motor pasa a ser elegible para la actualización automática, la política controla la secuencia de actualización en función de los órdenes definidos:

- Los recursos marcados como [primeros] (normalmente los entornos de desarrollo) pasan a ser elegibles para actualizaciones durante sus periodos de mantenimiento.
- Después de un tiempo de incorporación designado, los recursos marcados como [segundos] pasan a ser elegibles.
- Tras otro tiempo de incorporación designado, los recursos marcados como [últimos] (normalmente entornos de producción) pasan a ser elegibles.
- Supervisión del progreso de las actualizaciones mediante notificaciones de AWS Health.

Puede definir los órdenes de actualización de la siguiente manera:

- Políticas por cuenta: se aplican a todos los recursos elegibles de cuentas específicas.
- Etiquetas de recursos: se aplican a recursos específicos en función de las etiquetas.

> **Note**
>
> Los recursos no configurados con una política de actualización o excluidos de la política reciben automáticamente un orden de actualización de [segundo].

Requisitos previos

- Su Cuenta de AWS debe formar parte de una organización en Organizaciones con la política de implementación de actualizaciones habilitada

- Habilitación de actualizaciones de versiones secundarias automáticas para los clústeres
- Las etiquetas no son estrictamente obligatorias para la política de implementación de actualizaciones. Si desea definir los órdenes de actualización específicos para diferentes entornos (por ejemplo, desarrollo, prueba, QA, producción), puede usar etiquetas. Si no incluye la configuración de etiquetas en la política, todos los recursos incluidos en esa política seguirán el orden de actualización predeterminado. Para los recursos de Aurora, solo se utilizan etiquetas por clúster para la política de implementación de actualizaciones, incluso si tiene etiquetas definidas por instancia.

Etiquetado de los recursos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster que desea etiquetar.
4. Elija Actions y, a continuación, elija Manage tags.
5. Seleccione Agregar etiqueta.
6. Ingrese la clave de la etiqueta (por ejemplo, “Entorno”) y el valor (por ejemplo, “Desarrollo”)
7. Elija Agregar etiqueta y, a continuación, Guardar.

También puede agregar etiquetas mediante la AWS CLI:

```
aws rds add-tags-to-resource \
    --resource-name arn:aws:rds:region:account-number:cluster:cluster-name \
    --tags Key=Environment,Value=Development
```

Orden y fases de la actualización

La política de implementación de actualizaciones admite tres órdenes de actualización:

- [primero]: normalmente se usa para entornos de desarrollo o pruebas
- [segundo]: normalmente se usa para entornos de control de calidad. Orden predeterminado de los recursos si la política no está configurada específicamente
- [último]: normalmente se reserva para entornos de producción

Cuando una nueva versión secundaria del motor pasa a ser elegible para la actualización automática:

- Los recursos con orden de actualización [primero] pasan a ser elegibles para actualizaciones durante sus periodos de mantenimiento configurados.
- Después de un tiempo de incorporación designado, los recursos con orden de actualización [primero] pasan a ser elegibles para actualizaciones durante sus periodos de mantenimiento.
- Después de un tiempo de incorporación designado, los recursos con orden de actualización [primero] pasan a ser elegibles para actualizaciones durante sus periodos de mantenimiento.
- La campaña de actualización automática de las versiones secundarias se cierra una vez que se hayan actualizado todos los recursos elegibles con órdenes de mejora [primero], [segundo] y [último] o cuando la campaña alcance la fecha de finalización programada, lo que ocurra primero.

> **Note**
>
> Todas las actualizaciones automáticas de las versiones secundarias se realizan durante el periodo de mantenimiento configurado de cada clúster para minimizar el posible impacto en las aplicaciones.

Observabilidad

AWS Health y supervisión

Recibe notificaciones de AWS Health:

- Antes de que empiece una campaña de actualización de versión secundaria automática
- Entre cada fase, realice una transición para ayudar a rastrear y supervisar el progreso de la actualización
- Actualizaciones de progreso que muestran la cantidad de recursos actualizados en la flota en la consola de AWS Health

Notificaciones de eventos de Amazon RDS:

- Notificaciones para recursos habilitados para actualizaciones de versiones secundarias automáticas, como:

  - Cuando el recurso pase a ser elegible para la actualización en función de su orden de actualización ([primero], [segundo] o [último])
  - Plazo de actualización programada durante el periodo de mantenimiento
  - Estado de inicio y finalización de la actualización de la base de datos individual
- Suscribirse a estos eventos a través de Amazon EventBridge0 para una supervisión automatizada

Consideraciones

Algunas cuestiones a tener en cuenta:

- La política se aplica a todas las futuras campañas de actualización automática de versiones secundarias, incluidos los cambios de política realizados durante las campañas activas.
- Si se une a una campaña de actualizaciones en curso, los recursos seguirán el orden de actualización actual y no esperarán a que se configure una política.
- Los recursos no configurados con una política de actualización o excluidos de la política reciben automáticamente un orden de actualización de [segundo].
- La política proporciona periodos de validación entre las fases de actualización antes de pasar a la siguiente fase.
- Los cambios en la política o en las etiquetas de recursos requieren tiempo para propagarse antes de que se aplique el nuevo orden de actualización.
- La política se aplica solo a los recursos de Aurora con las actualizaciones automáticas de versiones secundarias habilitadas.
- Si detecta un problema en un entorno, puede desactivar las actualizaciones automáticas de las versiones secundarias para los entornos posteriores o utilizar el periodo de validación para resolver los problemas antes de que las actualizaciones pasen al siguiente orden de actualización.

Para obtener más información acerca del etiquetado de recursos de RDS, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS. Para obtener instrucciones detalladas sobre la configuración y el uso de la política de implementación de actualizaciones, consulte Primeros pasos de AWS Organizations en la Guía del usuario de AWS Organizations.

Reinicio de un clúster de base de datos de Amazon Aurora o de una instancia de base de datos de Amazon Aurora

Es posible que tenga que reiniciar el clúster de base de datos o algunas instancias dentro del clúster, normalmente por razones de mantenimiento. Por ejemplo, supongamos que modifica los parámetros de un grupo de parámetros o asocia otro grupo de parámetros al clúster. En estos casos, debe reiniciar el clúster para que los cambios tengan efecto. Del mismo modo, puede reiniciar una o más instancias de base de datos del lector dentro del clúster. Puede organizar las operaciones de reinicio de instancias individuales para minimizar el tiempo de inactividad de todo el clúster.

El tiempo necesario para reiniciar cada instancia de base de datos del clúster depende de la actividad de la base de datos en el momento del reinicio. También depende del proceso de recuperación del motor de base de datos específico. Si resulta práctico, reduzca la actividad de la base de datos para esa instancia en particular antes de iniciar el proceso de reinicio. Al hacerlo, se puede reducir el tiempo necesario para reiniciar la base de datos.

Solo puede reiniciar cada instancia de base de datos del clúster cuando esté disponible. Una instancia de base de datos puede no estar disponible por varios motivos. Estos incluyen el estado de detención del clúster, una modificación que se aplica a la instancia y una acción de tiempo de mantenimiento, como, por ejemplo, una actualización de versión.

Cuando se reinicia una instancia de base de datos, se reinicia el proceso del motor de la base de datos. Al reiniciar una instancia de base de datos, se produce una interrupción momentánea, durante la cual su estado se establece en rebooting.

> **Note**
>
> Por ejemplo, si la instancia de base de datos no está utilizando los cambios más recientes del grupo de parámetros de base de datos asociado, la Consola de administración de AWS muestra el grupo de parámetros de base de datos con el estado pending-reboot. El estado de los grupos de parámetros pending-reboot no genera un reinicio automático durante la siguiente ventana de mantenimiento. Para aplicar los cambios de parámetros más recientes en esa instancia de base de datos, reinicie manualmente la instancia de base de datos. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.

Temas

- Reinicio de una instancia de base de datos dentro de un clúster de Aurora
- Reinicio de un clúster de Aurora con disponibilidad de lectura
- Reinicio de un clúster de Aurora con disponibilidad de lectura
- Verificación del tiempo de actividad de clústeres e instancias de Aurora
- Ejemplos de operaciones de reinicio de Aurora

Reinicio de una instancia de base de datos dentro de un clúster de Aurora

Este procedimiento es la operación más importante que se realiza al ejecutar los reinicios con Aurora. Muchos de los procedimientos de mantenimiento implican el reinicio de una o más instancias de base de datos de Aurora en un orden determinado.

Consola

Para reiniciar una instancia de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos que desee reiniciar.
3. Para Actions (Acciones), elija Reboot (Reiniciar).

   Aparece la página Reboot DB Instance.
4. Elija Reboot para reiniciar su instancia de base de datos.

   O elija Cancel (Cancelar).

AWS CLI

Para reiniciar una instancia de base de datos mediante la AWS CLI, llame al comando `reboot-db-instance`.

Example

Para Linux, macOS o:Unix

```
aws rds reboot-db-instance \
```

```
    --db-instance-identifier mydbinstance
```

En:Windows

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance
```

API de RDS

Para reiniciar una instancia de base de datos mediante la Amazon RDS API, llame a la `RebootDBInstance` operación.

Reinicio de un clúster de Aurora con disponibilidad de lectura

Con la característica de disponibilidad de lectura, puede reiniciar la instancia de escritura de su clúster de Aurora sin reiniciar las instancias del lector en el clúster de base de datos principal o secundario. Al hacerlo, puede ayudar a mantener la alta disponibilidad del clúster para las operaciones del lector mientras reinicia la instancia del escritor. Puede reiniciar las instancias del lector más tarde, según la programación que le resulte conveniente. Por ejemplo, en un clúster de producción, puede reiniciar las instancias del lector de una en una y comenzar solo después de que finalice el reinicio de la instancia principal. Para cada instancia de base de datos que reinicie, siga el procedimiento descrito en Reinicio de una instancia de base de datos dentro de un clúster de Aurora.

La característica de disponibilidad de lectura de los clústeres de base de datos principales está disponible en Aurora MySQL 2.10 y versiones posteriores. La característica de lectura de los clústeres de base de datos secundario está disponible en Aurora MySQL 3.06 y versiones posteriores.

Esta característica está disponible de forma predeterminada para las siguientes versiones de Aurora PostgreSQL:

- Versión 15.2 y versiones posteriores a la 15
- Versión 14.7 y versiones posteriores a la 14
- Versión 13.10 y versiones posteriores a la 13
- Versión 12.14 y versiones posteriores a la 12

Para obtener más información sobre la característica de disponibilidad de lectura en Aurora PostgreSQL, consulte Mejora de la disponibilidad de lectura de las réplicas de Aurora.

Antes de esta característica, reiniciar la instancia principal provocaba también el reinicio de cada instancia del lector. Si el clúster de Aurora ejecuta una versión anterior, utilice el procedimiento de reinicio descrito en Reinicio de un clúster de Aurora con disponibilidad de lectura en su lugar.

> **Note**
>
> El cambio en el comportamiento de reinicio en los clústeres de base de datos de Aurora con disponibilidad de lectura es diferente para las bases de datos globales de Aurora en versiones de Aurora MySQL anteriores a la 3.06. Si reinicia la instancia del escritor del clúster principal de una base de datos de Aurora global, las instancias del lector del clúster principal permanecen disponibles. Sin embargo, las instancias de base de datos de cualquier clúster secundario se reinician al mismo tiempo.
>
> Hay una versión limitada de la característica de disponibilidad de lectura mejorada que es compatible con las bases de datos globales de Aurora para las versiones 12.16, 13.12, 14.9, 15.4 y posteriores de Aurora PostgreSQL.

Con frecuencia, reinicia el clúster después de realizar cambios en los grupos de parámetros del clúster. Puede realizar cambios en los parámetros siguiendo los procedimientos descritos en Grupos de parámetros para Amazon Aurora. Supongamos que reinicia la instancia de base de datos del escritor en un clúster de Aurora para aplicar cambios a los parámetros del clúster. Es posible que algunas o todas las instancias de base de datos del lector sigan utilizando la configuración de parámetros anterior. Sin embargo, las distintas configuraciones de parámetros no afectan la integridad de los datos del clúster. Los parámetros de clúster que afectan a la organización de los archivos de datos solo se utilizan por la instancia de base de datos del escritor.

Por ejemplo, en un clúster de Aurora MySQL puede actualizar los parámetros del clúster, como `binlog_format` y `innodb_purge_threads`, en la instancia del escritor antes de las instancias del lector. Solo la instancia del escritor escribe registros binarios y purga registros de deshacer. Para los parámetros que cambian la forma en que las consultas interpretan las instrucciones SQL o la salida de las consultas, es posible que deba reiniciar inmediatamente las instancias del lector. Esto se hace para evitar un comportamiento inesperado de la aplicación durante las consultas. Por ejemplo, supongamos que cambia el parámetro `lower_case_table_names` y reinicia la instancia del escritor. En este caso, es posible que las instancias del lector no puedan acceder a una tabla recién creada hasta que se reinicien todas.

Para obtener una lista de todos los parámetros de clúster de Aurora MySQL, consulte Parámetros de nivel de clúster.

Para obtener una lista de todos los parámetros de clúster de Aurora PostgreSQL, consulte Parámetros de nivel de clúster de Aurora PostgreSQL.

> **Tip**
>
> Aurora MySQL podría reiniciar algunas de las instancias del lector junto con la instancia del escritor si el clúster tiene una carga de trabajo en proceso con un alto rendimiento. La reducción en el número de reinicios se aplica también durante las operaciones de conmutación por error. Aurora reinicia solo la instancia de base de datos del escritor y el destino de conmutación por error durante una conmutación por error. Otras instancias de base de datos del lector en el clúster siguen disponibles para continuar el procesamiento de consultas mediante conexiones con el punto de enlace del lector. Por lo tanto, puede mejorar la disponibilidad durante una conmutación por error si tiene más de una instancia de base de datos del lector en un clúster.

Reinicio de un clúster de Aurora con disponibilidad de lectura

Con la característica de disponibilidad de lectura, puede reiniciar un clúster de base de datos de Aurora entero reiniciando la instancia de base de datos de escritor de dicho clúster. Para ello, siga el procedimiento en Reinicio de una instancia de base de datos dentro de un clúster de Aurora.

Al reiniciar la instancia de base de datos del escritor, también se reinicia cada instancia de base de datos del lector del clúster. De esta forma, los cambios de parámetros de todo el clúster se aplican a todas las instancias de base de datos al mismo tiempo. Sin embargo, el reinicio de todas las instancias de base de datos provoca una breve interrupción del clúster. Las instancias de base de datos del lector siguen sin estar disponibles hasta que la instancia de base de datos del escritor termine de reiniciarse y esté disponible.

Este comportamiento de reinicio se aplica a todos los clústeres de bases de datos creados en la versión 2.09 y versiones anteriores de Aurora MySQL.

Para Aurora PostgreSQL, este comportamiento se aplica a las siguientes versiones:

- Versión 14.6 y anteriores a la 14
- Versión 13.9 y anteriores a la 13
- Versión 12.13 y anteriores a la 12
- Todas las versiones 11 de PostgreSQL

En la consola de RDS, la instancia de base de datos del escritor tiene el valorWriter (Escritor) en la columna Role (Rol) de la página Databases (Bases de datos). En la CLI de RDS, el resultado del comando `describe-db-clusters` incluye una sección `DBClusterMembers`. El elemento `DBClusterMembers` que representa la instancia de base de datos del escritor tiene un valor `true` para el campo `IsClusterWriter`.

> **Important**
>
> Con la característica de disponibilidad de lectura, el comportamiento de reinicio es diferente en Aurora MySQL y Aurora PostgreSQL: las instancias de base de datos del lector suelen permanecer disponibles mientras reinicia la instancia del escritor. A continuación, puede reiniciar las instancias del lector en el momento conveniente. Puede reiniciar las instancias del lector en una programación escalonada si desea que algunas instancias del lector estén siempre disponibles. Para obtener más información, consulte Reinicio de un clúster de Aurora con disponibilidad de lectura.

Verificación del tiempo de actividad de clústeres e instancias de Aurora

Puede verificar y monitorear el tiempo transcurrido desde el último reinicio de cada instancia de base de datos del clúster de Aurora. La métrica `EngineUptime` de Amazon CloudWatch informa el número de segundos transcurridos desde el último inicio de una instancia de base de datos. Puede examinar esta métrica en un momento dado para averiguar el tiempo de actividad de la instancia de base de datos. También puede monitorear esta métrica a lo largo del tiempo para detectar cuándo se reinicia la instancia.

También puede examinar la métrica `EngineUptime` a nivel de clúster. Las dimensiones `Minimum` y `Maximum` indican los valores de tiempo de actividad máximos y mínimos de todas las instancias de base de datos del clúster. Para verificar la última vez en que se reinició cualquier instancia del lector de un clúster normalmente o por otro motivo, monitoree la métrica de nivel de clúster mediante la dimensión `Minimum`. Para verificar qué instancia del clúster duró más tiempo sin reiniciarse, monitoree la métrica de nivel de clúster mediante la dimensión `Maximum`. Por ejemplo, puede que desee confirmar que todas las instancias de base de datos del clúster se reiniciaron tras un cambio de configuración.

> **Tip**
>
> Para el monitoreo a largo plazo, recomendamos controlar la métrica `EngineUptime` de instancias individuales, en lugar de hacerlo a nivel del clúster. La métrica a nivel del clúster `EngineUptime` se establece en cero cuando se agrega una nueva instancia de base de datos al clúster. Estos cambios del clúster pueden ser parte de las operaciones de mantenimiento y escalado, como las realizadas por Amazon EC2 Auto Scaling.

En los siguientes ejemplos de la CLI, se muestra cómo examinar la métrica `EngineUptime` de las instancias del escritor y del lector de un clúster. En los ejemplos, se utiliza un clúster denominado `tpch100g`. Este clúster tiene una instancia de base de datos del escritor `instance-1234`. También tiene dos instancias de base de datos del lector: `instance-7448` y `instance-6305`.

En primer lugar, el comando `reboot-db-instance` reinicia una de las instancias del lector. El comando `wait` espera que la instancia termine de reiniciarse.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-6305
```

El comando `get-metric-statistics` de CloudWatch examina la métrica `EngineUptime` durante los últimos cinco minutos en intervalos de un minuto. El tiempo de actividad de la instancia `instance-6305` se restablece a cero y vuelve a contar hacia arriba. En este ejemplo de la AWS CLI de Linux, se utiliza la sustitución de variables `$()` para insertar las marcas de tiempo apropiadas en los comandos de la CLI. También se utiliza el comando `sort` de Linux para ordenar la salida en el momento en que se recopila la métrica. Ese valor de marca de tiempo es el tercer campo de cada línea de salida.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 231.0 2021-03-16T18:19:00+00:00 Seconds
```

```
DATAPOINTS 291.0 2021-03-16T18:20:00+00:00 Seconds
DATAPOINTS 351.0 2021-03-16T18:21:00+00:00 Seconds
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
```

El tiempo de actividad mínimo del clúster se restablece a cero porque se reinició una de las instancias del clúster. El tiempo de actividad máximo del clúster no se restablece porque al menos una de las instancias de base de datos del clúster permaneció disponible.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Minimum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63099.0 2021-03-16T18:12:00+00:00 Seconds
DATAPOINTS 63159.0 2021-03-16T18:13:00+00:00 Seconds
DATAPOINTS 63219.0 2021-03-16T18:14:00+00:00 Seconds
DATAPOINTS 63279.0 2021-03-16T18:15:00+00:00 Seconds
DATAPOINTS 51.0 2021-03-16T18:16:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63389.0 2021-03-16T18:16:00+00:00 Seconds
DATAPOINTS 63449.0 2021-03-16T18:17:00+00:00 Seconds
DATAPOINTS 63509.0 2021-03-16T18:18:00+00:00 Seconds
DATAPOINTS 63569.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 63629.0 2021-03-16T18:20:00+00:00 Seconds
```

A continuación, otro comando `reboot-db-instance` reinicia la instancia del escritor del clúster. Otro comando `wait` realiza una pausa hasta que la instancia del escritor termine de reiniciarse.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
  "DBInstanceIdentifier": "instance-1234",
  "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-1234
```

Ahora, la métrica `EngineUptime` de la instancia del escritor muestra que la instancia `instance-1234` se reinició recientemente. La instancia del lector `instance-6305` también se reinició automáticamente junto con la instancia del escritor. Este clúster ejecuta la versión 2.09 de Aurora MySQL, lo que no mantiene las instancias del lector en ejecución mientras se reinicia la instancia del escritor.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-1234 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63749.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 63809.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 63869.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 41.0 2021-03-16T18:25:00+00:00 Seconds
DATAPOINTS 101.0 2021-03-16T18:26:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 531.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 49.0 2021-03-16T18:26:00+00:00 Seconds
```

Ejemplos de operaciones de reinicio de Aurora

En los siguientes ejemplos de Aurora MySQL, se muestran diferentes combinaciones de operaciones de reinicio de instancias de base de datos del lector y del escritor en un clúster de base de datos de Aurora. Después de cada reinicio, las consultas SQL demuestran el tiempo de actividad de las instancias del clúster.

Temas

- Búsqueda de instancias del escritor y del lector de un clúster de Aurora
- Reinicio de una única instancia del lector
- Reinicio de la instancia del escritor

- Reiniciar las instancias del lector y del escritor de forma independiente
- Aplicación de un cambio de parámetro de clúster a un clúster de Aurora MySQL versión 2.10

Búsqueda de instancias del escritor y del lector de un clúster de Aurora

En un clúster de Aurora MySQL con varias instancias de base de datos, es importante saber cuál es el escritor y cuáles son los lectores. Las instancias del escritor y lector también pueden cambiar de rol cuando se produce una operación de conmutación por error. Por lo tanto, es mejor realizar una verificación como la siguiente antes de realizar cualquier operación que requiera una instancia del escritor o del lector. En este caso, los valores `False` para `IsClusterWriter` identifican las instancias del lector, `instance-6305` y `instance-7448`. El valor `True` identifica la instancia del escritor, `instance-1234`.

```
$ aws rds describe-db-clusters --db-cluster-id tpch100g \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:     tpch100g
Instance:    instance-6305    False
Instance:    instance-7448    False
Instance:    instance-1234    True
```

Antes de comenzar con los ejemplos de reinicio, la instancia del escritor tiene un tiempo de actividad de aproximadamente una semana. La consulta SQL de este ejemplo muestra una forma específica de MySQL de verificar el tiempo de actividad. Puede utilizar esta técnica en una aplicación de base de datos. Para obtener información sobre otra técnica que utiliza la AWS CLI y funciona para ambos motores de Aurora, consulte Verificación del tiempo de actividad de clústeres e instancias de Aurora.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-08 17:49:06.000000 | 174h 42m|
+----------------------------+---------+
```

Reinicio de una única instancia del lector

En este ejemplo, se reinicia una de las instancias de base de datos del lector. Tal vez esta instancia se sobrecargó debido a una consulta enorme o a varias conexiones simultáneas. O tal vez se quedó atrás de la instancia del escritor debido a un problema de red. Después de comenzar con la operación de reinicio, el ejemplo utiliza un comando `wait` para realizar una pausa hasta que la instancia esté disponible. Para entonces, la instancia tiene un tiempo de actividad de unos minutos.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-6305
$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:35:02.000000 | 00h 03m |
+----------------------------+---------+
```

El reinicio de la instancia del lector no afectó al tiempo de actividad de la instancia del escritor. Aún tiene un tiempo de actividad de aproximadamente una semana.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+----------+
| Last Startup               | Uptime   |
+----------------------------+----------+
| 2021-03-08 17:49:06.000000 | 174h 49m |
+----------------------------+----------+
```

Reinicio de la instancia del escritor

En este ejemplo, se reinicia la instancia del escritor. Este clúster ejecuta la versión 2.09 de Aurora MySQL. Dado que la versión de Aurora MySQL es anterior a 2.10, al reiniciar la instancia del escritor también se reinicia cualquier instancia del lector del clúster.

Un comando `wait` realiza una pausa hasta que finaliza el reinicio. Ahora, el tiempo de actividad de esa instancia se restablece a cero. Es posible que una operación de reinicio demore tiempos significativamente diferentes para las instancias de base de datos del escritor y del lector. Las instancias de base de datos del escritor y del lector realizan diferentes tipos de operaciones de limpieza en función de sus roles.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-1234",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-1234
$ mysql -h instance-1234.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:27.000000 | 00h 00m |
+----------------------------+---------+
```

Tras el reinicio de la instancia de base de datos del escritor, también se restablece el tiempo de actividad de ambas instancias de base de datos del lector. Reiniciar la instancia del escritor también reinicia las instancias del lector. Este comportamiento se aplica a los clústeres de Aurora PostgreSQL y a los clústeres de Aurora MySQL anteriores a la versión 2.10.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
```

```
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:35.000000 | 00h 00m |
+----------------------------+---------+

$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:33.000000 | 00h 01m |
+----------------------------+---------+
```

Reiniciar las instancias del lector y del escritor de forma independiente

En los siguientes ejemplos, se muestra un clúster que ejecuta la versión 2.10 de Aurora MySQL. En esta versión de Aurora MySQL y en versiones posteriores, puede reiniciar la instancia del escritor sin provocar el reinicio de todas las instancias del lector. De esta forma, las aplicaciones que requieren un uso intensivo de consultas no experimentan interrupciones cuando reinicia la instancia del escritor. Puede reiniciar las instancias del lector más tarde. Puede ejecutar estos reinicios en un momento de bajo tráfico de consultas. También puede reiniciar las instancias del lector de una a la vez. De esta forma, al menos una instancia del lector siempre estará disponible para el tráfico de consultas de la aplicación.

En el ejemplo siguiente, se utiliza un clúster denominado `cluster-2393` que ejecuta la versión `5.7.mysql_aurora.2.10.0` de Aurora MySQL. Este clúster tiene una instancia del escritor denominada `instance-9404` y tres instancias del lector denominadas `instance-6772`, `instance-2470` y `instance-5138`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:        cluster-2393
Instance:       instance-5138        False
Instance:       instance-2470        False
```

```
Instance:       instance-6772        False
Instance:       instance-9404        True
```

La verificación del valor del `uptime` de cada instancia de base de datos mediante el comando `mysql` muestra que cada una tiene aproximadamente el mismo tiempo de actividad. Por ejemplo, aquí está el tiempo de actividad de `instance-5138`.

```
mysql> SHOW GLOBAL STATUS LIKE 'uptime';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Uptime        | 3866  |
+---------------+-------+
```

Mediante CloudWatch, podemos obtener la información de tiempo de actividad correspondiente sin realmente iniciar sesión en las instancias. De esta forma, un administrador puede monitorear la base de datos, pero no puede ver ni cambiar ningún dato de tabla. En este caso, especificamos un periodo que abarca cinco minutos y verificamos el valor del tiempo de actividad cada minuto. Los crecientes valores de tiempo de actividad demuestran que las instancias no se reiniciaron durante ese periodo.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4648.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4708.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4768.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4828.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4888.0 2021-03-17T23:46:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4315.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4375.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4435.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4495.0 2021-03-17T23:45:00+00:00 Seconds
```

```
DATAPOINTS 4555.0 2021-03-17T23:46:00+00:00 Seconds
```

Ahora, reiniciamos una de las instancias del lecto, `instance-5138`. Esperamos que la instancia vuelva a estar disponible tras el reinicio. Ahora, el monitoreo del tiempo de actividad durante un periodo de cinco minutos muestra que el tiempo de actividad se restableció a cero durante ese tiempo. El valor de tiempo de actividad más reciente se midió cinco segundos después de que finalizó el reinicio.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4500.0 2021-03-17T23:46:00+00:00 Seconds
DATAPOINTS 4560.0 2021-03-17T23:47:00+00:00 Seconds
DATAPOINTS 4620.0 2021-03-17T23:48:00+00:00 Seconds
DATAPOINTS 4680.0 2021-03-17T23:49:00+00:00 Seconds
DATAPOINTS  5.0 2021-03-17T23:50:00+00:00 Seconds
```

A continuación, ejecutamos un reinicio para la instancia del escrito, `instance-9404`. Comparamos los valores de tiempo de actividad de la instancia del escritor y de una de las instancias del lector. Al hacerlo, podemos ver que reiniciar el escritor no provocó un reinicio de los lectores. En versiones anteriores a la 2.10 de Aurora MySQL, los valores de tiempo de actividad de todos los lectores se restablecían al mismo tiempo que el escritor.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
{
  "DBInstanceIdentifier": "instance-9404",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-9404

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
```

```
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 371.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 431.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 491.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 551.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS  37.0  2021-03-18T00:01:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5215.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 5275.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 5335.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 5395.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS 5455.0 2021-03-18T00:01:00+00:00 Seconds
```

Para asegurarse de que todas las instancias del lector tengan los mismos cambios en los parámetros de configuración que la instancia del escritor, reinicie todas las instancias del lector después del escritor. En este ejemplo, se reinician todos los lectores y luego se espera que todos estén disponibles antes de continuar.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6772
{
  "DBInstanceIdentifier": "instance-6772",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
```

```
  "DBInstanceStatus": "rebooting"
}

$ aws rds wait db-instance-available --db-instance-id instance-6772
$ aws rds wait db-instance-available --db-instance-id instance-2470
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

Ahora podemos ver que la instancia de base de datos del escritor tiene el mayor tiempo de actividad. El valor de tiempo de actividad de esta instancia aumentó constantemente durante todo el periodo de monitoreo. Las instancias de base de datos del lector se reiniciaron después del lector. Podemos ver el punto dentro del periodo de monitoreo en el que se reinició cada lector y su tiempo de actividad se restableció a cero.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 457.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 517.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 577.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 637.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS  697.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-2470 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5819.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS  35.0  2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 95.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 155.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 215.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
```

```
DATAPOINTS 1085.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 1145.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 1205.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS  49.0  2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 109.0 2021-03-18T00:12:00+00:00 Seconds
```

Aplicación de un cambio de parámetro de clúster a un clúster de Aurora MySQL versión 2.10

En el siguiente ejemplo, se muestra cómo aplicar un cambio de parámetro a todas las instancias de base de datos del clúster de Aurora MySQL 2.10. Con esta versión de Aurora MySQL, reinicia la instancia del escritor y todas las instancias del lector de forma independiente.

En el ejemplo se utiliza el parámetro de configuración de MySQL `lower_case_table_names` como ilustración. Cuando esta configuración de parámetro es diferente entre las instancias de base de datos del escritor y del lector, es posible que una consulta no logre acceder a una tabla declarada con un nombre en mayúsculas o minúsculas mixtas. O si dos nombres de tabla difieren solo en las mayúsculas y minúsculas, una consulta podría acceder a la tabla incorrecta.

En este ejemplo, se muestra cómo determinar las instancias del escritor y del lector del clúster al examinar el atributo `IsClusterWriter` de cada instancia. El clúster se denomina `cluster-2393`. El clúster tiene una instancia del escritor denominada `instance-9404`. Las instancias del lector del clúster se denominan `instance-5138` y `instance-2470`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
cluster-2393
instance-5138        False
instance-2470        False
instance-9404        True
```

Para demostrar los efectos de cambiar el parámetro `lower_case_table_names`, definimos dos grupos de parámetros de clúster de base de datos. El grupo de parámetros `lower-case-table-names-0` tiene este parámetro establecido en 0. El grupo de parámetros `lower-case-table-names-1` tiene este grupo de parámetros establecido en 1.

```
$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-0' \
  --db-parameter-group-family aurora-mysql5.7 \
```

```
  --db-cluster-parameter-group-name lower-case-table-names-0
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-0",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-0"
    }
}

$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-1' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-1",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-1"
    }
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-0 \
  --parameters
 ParameterName=lower_case_table_names,ParameterValue=0,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-0"
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-1 \
    --parameters
 ParameterName=lower_case_table_names,ParameterValue=1,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-1"
}
```

El valor predeterminado de `lower_case_table_names` es 0. Con esta configuración de parámetros, la tabla `foo` es distinta de la tabla `F00`. En este ejemplo, se verifica que el parámetro mantiene su configuración predeterminada. A continuación, en el ejemplo se crean tres tablas que solo difieren en las mayúsculas y minúsculas en sus nombres.

```
mysql> create database lctn;
Query OK, 1 row affected (0.07 sec)
```

```
mysql> use lctn;
Database changed
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> create table foo (s varchar(128));
mysql> insert into foo values ('Lowercase table name foo');

mysql> create table Foo (s varchar(128));
mysql> insert into Foo values ('Mixed-case table name Foo');

mysql> create table FOO (s varchar(128));
mysql> insert into FOO values ('Uppercase table name FOO');

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                         |
+--------------------------+
| Mixed-case table name Foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
+--------------------------+
```

A continuación, asociamos el grupo de parámetros de base de datos con el clúster para establecer el parámetro `lower_case_table_names` en 1. Este cambio solo tiene efecto después de reiniciar cada instancia de base de datos.

```
$ aws rds modify-db-cluster --db-cluster-identifier cluster-2393 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
  "DBClusterIdentifier": "cluster-2393",
  "DBClusterParameterGroup": "lower-case-table-names-1",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.0"
}
```

El primer reinicio que ejecutamos es para la instancia de base de datos del escritor. Luego, esperamos que la instancia vuelva a estar disponible. En ese momento, nos conectamos al punto de enlace del escritor y verificamos que la instancia del escritor tenga el valor del parámetro modificado. El comando `SHOW TABLES` confirma que la base de datos contiene las tres tablas diferentes. Sin embargo, todas las consultas que hacen referencia a tablas denominadas `foo`, `Foo` o `F00` acceden a la tabla cuyo nombre está en minúsculas, `foo`.

```
# Rebooting the writer instance
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
$ aws rds wait db-instance-available --db-instance-id instance-9404
```

Ahora, las consultas que utilizan el punto de enlace del clúster muestran los efectos del cambio de parámetros. Ya sea que el nombre de la tabla de la consulta está en mayúsculas, minúsculas o ambas, la instrucción SQL accederá a la tabla cuyo nombre está en minúsculas.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+

mysql> use lctn;
mysql> show tables;
+----------------+
| Tables_in_lctn |
+----------------+
| FOO            |
| Foo            |
| foo            |
+----------------+
```

```
mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from F00;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

En el siguiente ejemplo, se muestran las mismas consultas que la anterior. En este caso, las consultas utilizan el punto de enlace del lector y se ejecutan en una de las instancias de base de datos del lector. Todavía no se reiniciaron esas instancias. Por lo tanto, aún tienen la configuración original para el parámetro `lower_case_table_names`. Esto significa que las consultas pueden acceder a cada una de las tablas `foo`, `Foo` y `F00`.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

```
mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Mixed-case table name Foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
+--------------------------+
```

A continuación, reiniciamos una de las instancias del lector y esperamos que vuelva a estar disponible.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-2470
```

Mientras está conectado al punto de enlace de la instancia para `instance-2470`, una consulta muestra que el nuevo parámetro está activo.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+
```

En este punto, las dos instancias del lector del clúster se ejecutan con diferentes configuraciones `lower_case_table_names`. Por lo tanto, cualquier conexión con el punto de enlace del lector del clúster utiliza un valor para esta configuración que es impredecible. Es importante reiniciar inmediatamente la otra instancia del lector para que ambas tengan una configuración coherente.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
```

```
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

En el siguiente ejemplo se confirma que todas las instancias del lector tienen la misma configuración para el parámetro `lower_case_table_names`. Los comandos verifican el valor de configuración de `lower_case_table_names` para cada instancia del lector. A continuación, el mismo comando que utiliza el punto de enlace del lector demuestra que cada conexión al punto de enlace del lector utiliza una de las instancias del lector, pero no se puede predecir cuál de ellas.

```
# Check lower_case_table_names setting on each reader instance.

$ mysql -h instance-5138.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

$ mysql -h instance-2470.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-2470            |                        1 |
+--------------------------+--------------------------+

# Check lower_case_table_names setting on the reader endpoint of the cluster.

$ mysql -h cluster-2393.cluster-ro-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

# Run query on writer instance

$ mysql -h cluster-2393.cluster-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
```

```
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-9404            |                        1 |
+--------------------------+--------------------------+
```

Con el cambio de parámetros aplicado en todas partes, podemos ver el efecto de la configuración `lower_case_table_names=1`. Si la tabla se denomina `foo`, `Foo` o `F00`, la consulta convierte el nombre a `foo` y accede a la misma tabla en cada caso.

```
mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

Conmutación por error de un clúster de base de datos de Amazon Aurora

Puede realizar una conmutación por error manual de un clúster de base de datos Aurora, por ejemplo, cuando desee sustituir una instancia de base de datos de escritura aprovisionada por una instancia de escritor Aurora Serverless v2.

Aurora conmuta por error a una nueva instancia de base de datos principal de una de las dos formas siguientes:

- Promocionando una instancia de base de datos de lectura existente como nueva instancia principal
- Creando una nueva instancia principal

Si el clúster de base de datos tiene una o varias instancias de lectura, se promociona una lectura a instancia principal durante un evento de error. Para aumentar la disponibilidad de su clúster de base de datos, es recomendable que cree al menos una o varias instancias de lectura en dos o más zonas de disponibilidad diferentes. Para obtener información sobre el mecanismo de conmutación por error, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

Puede usar la Consola de administración de AWS, la AWS CLI o la API de RDS para realizar una conmutación por error manual.

Consola

Conmutación por error de un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Bases de datos y, a continuación, seleccione la instancia de base de datos en el clúster de base de datos donde desea realizar la conmutación por error.
3. En Actions (Acciones), elija Failover (conmutación por error).

   Aparece una página de confirmación.
4. Elija Failover (Conmutación por error).

En la página Bases de datos, se puede ver que el estado del clúster de base de datos es Conmutación por error. El estado vuelve a Disponible cuando se completa la conmutación por error y se muestran los roles de las instancias de base de datos principales nuevas y anteriores.

AWS CLI

Para llevar a cabo una conmutación por error de un clúster de base de datos con la AWS CLI, llame al comando failover-db-clúster. Especifique los siguientes parámetros:

- `--db-cluster-identifier`: el clúster de base de datos que desea conmutar por error.
- `--target-db-instance-identifier`: el nombre de la instancia de base de datos que se va a promocionar a instancia de base de datos principal.

Example

Para Linux, macOS o:Unix

```
aws rds failover-db-cluster \
    --db-cluster-identifier mydbcluster \
    --target-db-instance-identifier mydbcluster-instance-2
```

En:Windows

```
aws rds failover-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --target-db-instance-identifier mydbcluster-instance-2
```

API de RDS

Para modificar un clúster de base de datos mediante la API de Amazon RDS, llame a la operación FailoverDBclúster. Especifique los siguientes parámetros:

- DBclústerIdentifier
- TargetDBInstanceIdentifier

Eliminación de clústeres e instancias de base de datos de Aurora

Puede eliminar un clúster de base de datos de Aurora cuando ya no lo necesite. Al eliminar el clúster, se elimina el volumen del clúster que contiene todos los datos. Antes de eliminar el clúster, puede guardar una instantánea de los datos. Puede restaurar la instantánea más adelante para crear un nuevo clúster que contenga los mismos datos.

También puede eliminar las instancias de base de datos de un clúster y, al mismo tiempo, conservar el clúster en sí y los datos que contiene. Eliminar las instancias de base de datos puede ayudar a reducir los cargos si el clúster no está ocupado o si no necesita la capacidad de computación de varias instancias de base de datos.

Temas

- Eliminación de un clúster de base de datos de Aurora
- Protección contra eliminación para clústeres de Aurora
- Eliminación de un clúster detenido de Aurora
- Eliminación de clústeres de Aurora MySQL que son réplicas de lectura
- La instantánea final al eliminar un clúster
- Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora

Eliminación de un clúster de base de datos de Aurora

Aurora no proporciona un método de un solo paso para eliminar un clúster de base de datos. Esta opción de diseño está pensada para evitar que accidentalmente pierda datos o quite la conexión de la aplicación. Las aplicaciones de Aurora suelen ser de misión crítica y requieren alta disponibilidad. Por lo tanto, Aurora facilita el escalado ascendente y descendente de la capacidad del clúster mediante la incorporación y la eliminación de instancias de base de datos. Eliminar el clúster de base de datos en sí requiere que realice una eliminación por separado.

Utilice el siguiente procedimiento general para eliminar todas las instancias de base de datos de un clúster y, a continuación, elimine el clúster en sí.

1. Elimine todas las instancias de lector del clúster. Utilice el procedimiento en Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora.

   Si el clúster tiene alguna una instancia del lector, eliminar una de las instancias solo reduce la capacidad de computación del clúster. Si primero se eliminan las instancias de lector, se garantiza

que el clúster permanezca disponible durante todo el procedimiento y no realice operaciones de conmutación por error innecesarias.

2. Elimine la instancia de escritor del clúster. Utilice nuevamente el procedimiento en Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora.

   Si elimina las instancias de base de datos, el clúster y su volumen del clúster asociado permanecerán incluso después de eliminar todas las instancias de base de datos.

3. Elimine el clúster de base de datos.

   - Consola de administración de AWS: elija el clúster y, a continuación, elija Eliminar del menú Acciones. Puede elegir las siguientes opciones para conservar los datos del clúster en caso de que los necesite más adelante:
     - Crear una instantánea final del volumen del clúster. La configuración predeterminada es crear una instantánea final.
     - Conservar copias de seguridad automatizadas La configuración predeterminada es no retener copias de seguridad automatizadas.

     > **Note**
     >
     > No se conservan las copias de seguridad automatizadas de los clústeres de bases de datos de Aurora Serverless v1.

     Aurora también requiere que confirme que tiene intención de eliminar el clúster.
   - CLI y API: llame al comando de la CLI `delete-db-cluster` o a la operación de API `DeleteDBCluster`. Puede elegir las siguientes opciones para conservar los datos del clúster en caso de que los necesite más adelante:
     - Crear una instantánea final del volumen del clúster.
     - Conservar copias de seguridad automatizadas

     > **Note**
     >
     > No se conservan las copias de seguridad automatizadas de los clústeres de bases de datos de Aurora Serverless v1.

Temas

- Eliminación de un clúster de Aurora vacío
- Eliminación de un clúster de Aurora con una única instancia de base de datos
- Eliminación de un clúster Aurora con varias instancias de base de datos

Eliminación de un clúster de Aurora vacío

Puede eliminar un clúster de base de datos vacío con al Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

> **Tip**
>
> Puede mantener un clúster sin instancias de base de datos para conservar los datos sin incurrir en cargos por CPU por el clúster. Puede volver a utilizar rápidamente el clúster si crea una o más instancias de base de datos nuevas para el clúster. Sin embargo, solo puede agregar nuevas instancias de base de datos mediante la AWS CLI o la API de RDS. No puede agregar nuevas instancias de base de datos mediante la consola de Amazon RDS. Puede realizar operaciones administrativas específicas de Aurora en el clúster siempre y cuando no tenga ninguna instancia de base de datos asociada. No puede acceder a los datos ni realizar ninguna operación que requiera conectarse a una instancia de base de datos.

Consola

Para eliminar un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Bases de datos y, a continuación, elija el clúster de base de datos que desea eliminar.
3. En Actions (Acciones), seleccione Delete (Eliminar).
4. Para crear una instantánea de base de datos final del clúster de base de datos, elija ¿Crear la instantánea final?. Este es el valor predeterminado.
5. Si elige crear una instantánea final, introduzca el nombre de instantánea final.
6. Para conservar las copias de seguridad automatizadas, seleccione Retain automated backups (Conservar copias de seguridad automatizadas). Este no es el valor predeterminado.

7. En el cuadro, escriba **`delete me`**.
8. Elija Eliminar.

CLI

Para eliminar un clúster de base de datos de Aurora vacío mediante la AWS CLI, llame al comando delete-db-clúster.

Supongamos que el clúster vacío `deleteme-zero-instances` solo se utilizó para el desarrollo y la prueba, y que no contiene ningún dato importante. En ese caso, no es necesario conservar una instantánea del volumen del clúster cuando elimine el clúster. En el siguiente ejemplo se muestra que un clúster no contiene ninguna instancia de base de datos y, a continuación, se elimina el clúster vacío sin crear una instantánea final ni copias de seguridad automatizadas.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-zero-instances --output
 text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-zero-instances

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-zero-instances \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-zero-instances",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

API de RDS

Para eliminar un clúster de base de datos de Aurora vacío mediante la API de Amazon RDS, llame a la operación DeleteDBclúster.

Eliminación de un clúster de Aurora con una única instancia de base de datos

Puede eliminar la instancia de base de datos, incluso si el clúster de base de datos tiene la protección de eliminación habilitada. En ese caso, el propio clúster de base de datos sigue existiendo y se conservan los datos. Puede acceder a los datos de nuevo si asocia una nueva instancia de base de datos al clúster.

En el siguiente ejemplo se muestra de qué manera no funciona el comando `delete-db-cluster` cuando el clúster aún tiene instancias de base de datos asociadas. Este clúster tiene una única instancia de base de datos de escritor. Cuando examinamos las instancias de base de datos en el clúster, comprobamos el atributo `IsClusterWriter` de cada una. El clúster podría no tener instancias o tener una instancia de base de datos de escritor. Un valor de `true` implica una instancia de base de datos de escritor. Un valor de `false` implica una instancia de base de datos de lector. El clúster podría no tener, tener una o muchas instancias de base de datos de escritor. En este caso, eliminamos la instancia de base de datos de escritor usando el comando `delete-db-instance`. Tan pronto como la instancia de base de datos tenga el estado `deleting`, también podemos eliminar el clúster. En este ejemplo, también suponemos que el clúster no contiene ningún dato que valga la pena conservar. Por lo tanto, no creamos una instantánea del volumen del clúster ni retenemos copias de seguridad automatizadas.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only --skip-final-
snapshot
An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-only \
  --query '*[].[DBClusterIdentifier,Status,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]'
[
    [
        "deleteme-writer-only",
        "available",
        [
            [
                "instance-2130",
                true
            ]
        ]
    ]
]

$ aws rds delete-db-instance --db-instance-identifier instance-2130
{
  "DBInstanceIdentifier": "instance-2130",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}
```

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-only",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Eliminación de un clúster Aurora con varias instancias de base de datos

Si el clúster contiene varias instancias de base de datos, normalmente hay una única instancia de escritor y una o más instancias de lector. Las instancias de lector ayudan con la alta disponibilidad, ya que se encuentran en espera para hacerse cargo si la instancia de escritor encuentra algún problema. También puede utilizar instancias de lector para escalar el clúster de manera ascendente para gestionar una carga de trabajo de lectura intensiva sin agregar sobrecarga a la instancia de escritor.

Para eliminar un clúster con varias instancias de base de datos de lector, primero debe eliminar las instancias de lector y, luego, la instancia de escritor. La eliminación de la instancia de escritor deja el clúster y sus datos en su lugar. El clúster se elimina mediante una acción independiente.

- Para ver el procedimiento para eliminar una instancia de base de datos de Aurora, consulte Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora.
- Para ver el procedimiento para eliminar la instancia de base de datos de escritor en un clúster de Aurora, consulte Eliminación de un clúster de Aurora con una única instancia de base de datos.
- Para ver el procedimiento para eliminar un clúster vacío de Aurora, consulte Eliminación de un clúster de Aurora vacío.

En este ejemplo de la CLI se muestra cómo eliminar un clúster que contiene una instancia de base de datos de escritor y una única instancia de base de datos de lector. El resultado `describe-db-clusters` muestra que `instance-7384` es la instancia de escritor y `instance-1039` es la instancia de lector. En el ejemplo se elimina primero la instancia de lector, ya que eliminar la instancia de escritor mientras todavía existe una instancia de lector provocaría una operación de conmutación por error. No tiene sentido promover la instancia de lector a una de escritor si planea eliminar esa instancia también. Supongamos nuevamente que estas instancias `db.t2.small` solo

se utilizan para el desarrollo y la prueba, por lo que la operación de eliminación omite la instantánea final y no retiene copias de seguridad automatizadas.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader --skip-
final-snapshot

An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-and-reader --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-writer-and-reader
Instance:       instance-1039  False
Instance:       instance-7384   True

$ aws rds delete-db-instance --db-instance-identifier instance-1039
{
  "DBInstanceIdentifier": "instance-1039",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-7384
{
  "DBInstanceIdentifier": "instance-7384",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-and-reader",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

En el siguiente ejemplo se muestra cómo eliminar un clúster de base de datos que contiene una instancia de base de datos de escritor y varias instancias de base de datos de lector. Utiliza un

resultado conciso del comando `describe-db-clusters` para obtener un informe de las instancias de escritor y lector. Nuevamente, eliminamos todas las instancias de base de datos de lector antes de eliminar la instancia de base de datos de escritor. No importa el orden en que se eliminan las instancias de base de datos de lector.

Supongamos que este clúster con varias instancias de base de datos contiene datos que vale la pena conservar. Por lo tanto, el comando `delete-db-cluster` de este ejemplo incluye los parámetros `--no-skip-final-snapshot` y `--final-db-snapshot-identifier` para especificar los detalles de la instantánea que se creará. También incluye el parámetro `--no-delete-automated-backups` para retener las copias de seguridad automatizadas.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-multiple-readers --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-multiple-readers
Instance:       instance-1010   False
Instance:       instance-5410   False
Instance:       instance-9948   False
Instance:       instance-8451   True

$ aws rds delete-db-instance --db-instance-identifier instance-1010
{
  "DBInstanceIdentifier": "instance-1010",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-5410
{
  "DBInstanceIdentifier": "instance-5410",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-9948
{
  "DBInstanceIdentifier": "instance-9948",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}
```

```
$ aws rds delete-db-instance --db-instance-identifier instance-8451
{
  "DBInstanceIdentifier": "instance-8451",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-multiple-readers \
  --no-delete-automated-backups \
  --no-skip-final-snapshot \
  --final-db-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
  "DBClusterIdentifier": "deleteme-multiple-readers",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

En el siguiente ejemplo se muestra cómo confirmar que Aurora creó la instantánea solicitada. Puede solicitar detalles de la instantánea específica especificando su identificador `deleteme-multiple-readers-final-snapshot`. También puede obtener un informe de todas las instantáneas del clúster que se eliminó especificando el identificador del clúster `deleteme-multiple-readers`. Ambos comandos devuelven información sobre la misma instantánea.

```
$ aws rds describe-db-cluster-snapshots \
  --db-cluster-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...

$ aws rds describe-db-cluster-snapshots --db-cluster-identifier deleteme-multiple-
readers
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
```

```
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...
```

Protección contra eliminación para clústeres de Aurora

No puede eliminar clústeres que tengan habilitada la protección contra eliminación. Puede eliminar instancias de base de datos dentro del clúster, pero no el clúster en sí. De esta forma, el volumen del clúster que contiene todos los datos está protegido de la eliminación accidental. Aurora aplica la protección de eliminación para un clúster si intenta eliminar el clúster usando la consola, la AWS CLI o la API de RDS.

La protección contra eliminación se habilita de forma predeterminada cuando crea un clúster de base de datos de producción con la Consola de administración de AWS. Sin embargo, la protección contra eliminación está deshabilitada de forma predeterminada si crea un clúster con la AWS CLI o la API. Habilitar o deshabilitar la protección contra eliminación no provoca una interrupción. Para poder eliminar el clúster, modifique el clúster y deshabilite la protección contra eliminación. Para obtener más información sobre la activación y desactivación de la protección contra contraseña, consulte Modificación del clúster de base de datos con la consola, CLI y API.

> **Tip**
>
> Incluso si se eliminan todas las instancias de base de datos, puede acceder a los datos creando una nueva instancia de base de datos en el clúster.

Eliminación de un clúster detenido de Aurora

No puede eliminar un clúster si se encuentra en el estado `stopped`. En este caso, inicie el clúster antes de eliminarlo. Para obtener más información, consulte Inicio de un clúster de bases de datos de Aurora.

Eliminación de clústeres de Aurora MySQL que son réplicas de lectura

Para Aurora MySQL, no puede eliminar una instancia de la base de datos en un clúster de base de datos si se cumplen las dos condiciones siguientes:

- El clúster de base de datos es una réplica de lectura de otro clúster de base de datos de Aurora.
- La instancia de base de datos es la única instancia en el clúster de base de datos.

Para eliminar una instancia de base de datos en este caso, primero promocione el clúster de bases de datos para que deje de ser una réplica de lectura. Una vez finalizada la promoción, puede eliminar la instancia final de la base de datos en el clúster de bases de datos. Para obtener más información, consulte Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS.

La instantánea final al eliminar un clúster

A lo largo de esta sección, los ejemplos muestran cómo puede elegir si desea tomar una instantánea final al eliminar un clúster de Aurora. Si elige tomar una instantánea final pero el nombre especificado coincide con una instantánea existente, la operación se detiene con un error. En este caso, examine los detalles de la instantánea para confirmar si representa el detalle actual o si es una instantánea anterior. Si la instantánea existente no tiene los datos más recientes que desea conservar, cambie el nombre de la instantánea e inténtelo de nuevo, o especifique un nombre diferente para el parámetro de la instantánea final.

Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora

Puede eliminar una instancia de base de datos de un clúster de base de datos de Aurora como parte del proceso de eliminación de todo el clúster. Si el clúster contiene una cierta cantidad de instancias de base de datos, la eliminación del clúster requiere eliminar cada una de esas instancias de base de datos. También puede eliminar una o más instancias de lector de un clúster mientras el clúster se sigue ejecutando. Puede hacerlo para reducir la capacidad de cómputo y los cargos asociados si el clúster no está ocupado.

Para eliminar una instancia de base de datos, debe especificar el nombre de la instancia.

Puede eliminar una instancia de base de datos mediante la consola de Consola de administración de AWS, la AWS CLI o la API de RDS.

> **Note**
>
> Cuando se elimina una réplica de Aurora su punto de enlace de instancia se elimina inmediatamente y la réplica de Aurora se elimina del punto de enlace del lector. Si hay instrucciones que se ejecutan en la réplica de Aurora que se van a eliminar, hay un periodo de gracia de tres minutos. Las instrucciones existentes pueden finalizar durante el periodo de gracia. Cuando termina dicho periodo, se cierra la réplica de Aurora y se elimina.

Para los clústeres de base de datos de Aurora, eliminar una instancia de base de datos no necesariamente elimina todo el clúster. Puede eliminar una instancia de base de datos de Aurora de un clúster para reducir la capacidad de cómputo y los cargos asociados cuando el clúster no está ocupado. Para obtener información acerca de las circunstancias especiales de los clústeres de Aurora que tienen una instancia de base de datos o no tienen instancias de base de datos, consulte Eliminación de un clúster de Aurora con una única instancia de base de datos y Eliminación de un clúster de Aurora vacío.

> **Note**
>
> No se puede eliminar un clúster de base de datos cuando tiene habilitada la protección contra eliminación. Para obtener más información, consulte Protección contra eliminación para clústeres de Aurora.
> Puede deshabilitar la protección contra eliminación modificando el clúster de base de datos. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Consola

Para eliminar una instancia de base de datos en un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos que desee eliminar.
3. En Actions (Acciones), elija Delete (Eliminar).
4. En el cuadro, escriba **`delete me`**.
5. Elija Eliminar.

AWS CLI

Para eliminar una instancia de base de datos mediante la AWS CLI, llame al comando delete-db-instance y especifique el valor `--db-instance-identifier`.

Example

Para Linux, macOS o Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance
```

Para Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance
```

API de RDS

Para eliminar una instancia de base de datos con la API de Amazon RDS, llame a la operación DeleteDBInstance y especifique el parámetro `DBInstanceIdentifier`.

> **Note**
>
> Cuando el estado de una instancia de base de datos es `deleting`, su valor de certificado de entidad de certificación no aparece en la consola de RDS ni en la salida de comandos de la AWS CLI ni en las operaciones de la API de RDS. Para obtener más información acerca de los certificados de entidad de certificación, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

Etiquetado de los recursos de Amazon Aurora y Amazon RDS

Una etiqueta de Amazon RDS es un par nombre-valor que define y asocia a un recurso de Amazon RDS, como una instancia de base de datos o una instantánea de base de datos. El nombre es la clave. Opcionalmente, puede proporcionar un valor para la clave.

Puede utilizar la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS para agregar, enumerar y eliminar etiquetas de recursos de Amazon RDS. Si utiliza la CLI de o la API, asegúrese de proporcionar el nombre de recurso de Amazon (ARN) correspondiente al recurso de RDS con el que desee trabajar. Para obtener más información sobre cómo crear un ARN, consulte Creación de un nombre ARN para Amazon RDS.

Puede utilizar etiquetas para agregar metadatos a sus recursos de Aurora y Amazon RDS. Puede utilizar las etiquetas para agregar sus propias notaciones sobre instancias de base de datos, instantáneas, Aurora clústeres, etc. Si lo hace, puede ayudarlo a documentar sus recursos de Aurora y Amazon RDS. También puede utilizar las etiquetas con procedimientos de mantenimiento automatizados.

En concreto, puede utilizar estas etiquetas con las políticas de IAM. Puede utilizarlas para administrar el acceso a los recursos de Aurora y Amazon RDS y controlar qué acciones se pueden aplicar a estos recursos. También puede utilizar estas etiquetas para realizar un seguimiento de los costos al agrupar los gastos de recursos etiquetados de forma similar.

Puede etiquetar los siguientes recursos de Aurora y Amazon RDS:

- Instancias de base de datos
- Clústeres de base de datos
- Clústeres globales de Aurora
- Puntos de conexión de clústeres de base de datos
- Réplicas de lectura
- Instantáneas de base de datos
- Instantáneas de clúster de base de datos
- Instancias de base de datos reservadas
- Suscripciones de eventos
- Grupos de opciones de base de datos
- Grupos de parámetros de base de datos

- Grupos de parámetros de clúster de bases de datos
- Grupos de subred de base de datos
- Proxies de RDS Proxy
- Puntos de enlace de RDS Proxy
- Implementaciones azul/verde
- Integraciones sin ETL

> **Note**
>
> Al etiquetar una instancia de base de datos, Aurora aplica automáticamente esas etiquetas a los recursos de información de rendimiento asociados. Actualmente, no puede etiquetar los proxies de RDS ni los puntos de conexión de proxies de RDS mediante la Consola de administración de AWS.

Temas

- ¿Por qué usar etiquetas de recursos de Amazon RDS?
- Funcionamiento de las etiquetas de recursos de Amazon RDS
- Prácticas recomendadas para el etiquetado de los recursos de Amazon RDS
- Copia de etiquetas a instantáneas de clúster de base de datos
- Añadido y eliminación de etiquetas en Amazon RDS
- Tutorial: Uso de etiquetas para especificar qué clústeres de base de datos de Aurora se deben detener

¿Por qué usar etiquetas de recursos de Amazon RDS?

Puede usar etiquetas para hacer lo siguiente:

- Clasifique sus recursos de RDS por aplicación, proyecto, departamento, entorno, etc. Por ejemplo, puede usar una clave de etiqueta para definir una categoría en la que el valor de la etiqueta sea un elemento dentro de esa categoría. Podría crear la etiqueta `environment=prod`. También podría definir una clave de etiqueta de `project` y un valor de etiqueta de `Salix` para indicar que se ha asignado un recurso de Amazon RDS al proyecto Salix.

- Automatice las tareas de administración de recursos. Por ejemplo, podría crear una ventana de mantenimiento para las instancias etiquetadas con `environment=prod` que sea diferente de la ventana para las instancias etiquetadas con `environment=test`. También puede configurar instantáneas de bases de datos automáticas para las instancias etiquetadas con `environment=prod`.
- Controle el acceso a los recursos de RDS dentro de una política de IAM. Para ello, utilice la clave de condición `aws:ResourceTag/`*`tag-key`* global. Por ejemplo, una política podría permitir que solo los usuarios del grupo `DBAdmin` modifiquen las instancias de base de datos etiquetadas con `environment=prod`. Para obtener más información sobre la administración del acceso a los recursos etiquetados con políticas de IAM, consulte Administración de la identidad y el acceso en Amazon Aurora y Control del acceso a los recursos de AWS en la Guía del usuario de AWS Identity and Access Management.
- Supervise los recursos en función de una etiqueta. Por ejemplo, puede crear un panel de Amazon CloudWatch para instancias de base de datos etiquetadas con `environment=prod`.
- Realice un seguimiento de los costos agrupando los gastos por recursos con etiquetas similares. Por ejemplo, si etiqueta los recursos de RDS asociados al proyecto de Salix con `project=Salix`, puede generar informes de costos y asignar los gastos a este proyecto. Para obtener más información, consulte Funcionamiento de la facturación de AWS con etiquetas en Amazon RDS.

Funcionamiento de las etiquetas de recursos de Amazon RDS

AWS no aplica ningún significado semántico a las etiquetas. Las etiquetas se interpretan estrictamente como cadenas de caracteres.

Temas

- Conjuntos de etiquetas en Amazon RDS
- Estructura de etiquetas en Amazon RDS
- Recursos de Amazon RDS aptos para el etiquetado
- Funcionamiento de la facturación de AWS con etiquetas en Amazon RDS

Conjuntos de etiquetas en Amazon RDS

Cada recurso de Amazon RDS tiene un contenedor denominado conjunto de etiquetas. El contenedor incluye todas las etiquetas asignadas al recurso. Un recurso tiene exactamente un conjunto de etiquetas.

Un conjunto de etiquetas contiene de 0 a 50 etiquetas. Si agrega una etiqueta a un recurso de RDS con la misma clave que una etiqueta existente, el nuevo valor sobrescribirá al antiguo.

Estructura de etiquetas en Amazon RDS

La estructura de una etiqueta de RDS es la siguiente:

Clave de etiqueta

La clave de la etiqueta es el nombre obligatorio de la etiqueta. El valor de la cadena debe tener una longitud de entre 1 y 128 caracteres Unicode y no puede llevar el prefijo `aws:` ni `rds:`. La cadena puede contener únicamente el conjunto de letras Unicode, dígitos, espacio en blanco, `_`, `.`, `:`, `/`, `=`, `+`, `-` y `@`. La expresión regular de Java es `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. Las claves de etiqueta distinguen entre mayúsculas y minúsculas. Por lo tanto, las claves `project` y `Project` son distintas.

Una clave es única en un conjunto de etiquetas. Por ejemplo, en un conjunto de etiquetas no puede haber claves iguales pero con valores diferentes, como `project=Trinity` y `project=Xanadu`.

Valor de etiqueta

El valor es un valor de cadena opcional en la etiqueta. El valor de la cadena debe tener una longitud de entre 1 y 256 caracteres Unicode. La cadena puede contener únicamente el conjunto de letras Unicode, dígitos, espacio en blanco, `_`, `.`, `:`, `/`, `=`, `+`, `-` y `@`. La expresión regular de Java es `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. Los valores distinguen entre mayúsculas y minúsculas. Por lo tanto, los valores `prod` y `Prod` son distintos.

Los valores no tienen que ser únicos dentro de un conjunto de etiquetas y también pueden ser nulos. Por ejemplo, es posible tener en un conjunto de etiquetas los pares clave-valor `project=Trinity` y `cost-center=Trinity`.

Recursos de Amazon RDS aptos para el etiquetado

Puede etiquetar los siguientes recursos de Amazon RDS:

- Instancias de base de datos
- Clústeres de base de datos
- Puntos de conexión de clústeres de base de datos
- Réplicas de lectura

- Instantáneas de base de datos
- Instantáneas de clúster de base de datos
- Instancias de base de datos reservadas
- Suscripciones de eventos
- Grupos de opciones de base de datos
- Grupos de parámetros de base de datos
- Grupos de parámetros de clúster de bases de datos
- Grupos de subred de base de datos
- Proxies de RDS Proxy
- Puntos de enlace de RDS Proxy

> **Note**
>
> Actualmente, no puede etiquetar los proxies de RDS ni los puntos de conexión de proxies de RDS mediante la Consola de administración de AWS.

- Implementaciones azul/verde
- Integraciones sin ETL (versión preliminar)

Funcionamiento de la facturación de AWS con etiquetas en Amazon RDS

Puede usar etiquetas para organizar la factura de AWS de modo que refleje su propia estructura de costos. Para ello, inscríbase para obtener una factura de Cuenta de AWS que incluya valores de clave de etiquetas. A continuación, para ver los costos de los recursos combinados, organice la información de facturación de acuerdo con los recursos con los mismos valores de clave de etiquetas. Por ejemplo, puede etiquetar varios recursos con un nombre de aplicación específico y luego organizar su información de facturación para ver el costo total de la aplicación en distintos servicios. Para obtener más información, consulte Uso de etiquetas de asignación de costos en la Guía del usuario de AWS Billing.

Funcionamiento de las etiquetas de asignación de costos con las instantáneas de clústeres de bases de datos

Puede añadir una etiqueta a una instantánea de clúster de base de datos. Sin embargo, la factura no reflejará esta agrupación. Para que las etiquetas de asignación de costos se apliquen a las instantáneas de clúster de base de datos, se deben cumplir las siguientes condiciones:

- Las etiquetas se deben asociar a la instancia de base de datos principal.
- La instancia de base de datos principal debe existir en la misma Cuenta de AWS que la instantánea del clúster de base de datos.
- La instancia de base de datos principal debe existir en la misma Región de AWS que la instantánea del clúster de base de datos.

Las instantáneas del clúster de base de datos se consideran huérfanas si no existen en la misma región que la instancia de base de datos principal, o bien si la instancia de base de datos principal se elimina. Las instantáneas de bases de datos huérfanas no admiten etiquetas de asignación de costos. Los costos de las instantáneas huérfanas se agregan en un único elemento de línea sin etiquetar. Las instantáneas de clústeres de bases de datos entre cuentas no se consideran huérfanas cuando se cumplen las siguientes condiciones:

- Existen en la misma región que la instancia de base de datos principal.
- La instancia de base de datos principal es propiedad de la cuenta de origen.

> **Note**
>
> Si la instancia de base de datos principal pertenece a una cuenta diferente, las etiquetas de asignación de costos no se aplican a las instantáneas entre cuentas de la cuenta de destino.

Prácticas recomendadas para el etiquetado de los recursos de Amazon RDS

Al utilizar etiquetas, le sugerimos que siga las siguientes prácticas recomendadas:

- Documente las convenciones sobre el uso de etiquetas que siguen todos los equipos de su organización. En concreto, asegúrese de que los nombres sean descriptivos y coherentes. Por ejemplo, estandarice el formato `environment:prod` en lugar de etiquetar algunos recursos con `env:production`.

> **Important**
>
> No almacene información de identificación personal (PII) ni otra información confidencial en las etiquetas.

- Automatice el etiquetado para garantizar la coherencia. Por ejemplo, puede utilizar las siguientes técnicas:
  - Incluya etiquetas en una plantilla de CloudFormation. Al crear recursos con la plantilla, los recursos se etiquetan automáticamente.
  - Defina y aplique etiquetas mediante funciones de AWS Lambda.
  - Cree un documento SSM que incluya los pasos para añadir etiquetas a sus recursos de RDS.
- Use las etiquetas solo cuando sea necesario. Puede añadir hasta 50 etiquetas para un único recurso de RDS, pero se recomienda evitar la complejidad y la proliferación innecesarias de etiquetas.
- Revise las etiquetas periódicamente para comprobar su relevancia y precisión. Elimine o modifique las etiquetas obsoletas según sea necesario.
- Plantéese la posibilidad de crear etiquetas con el editor de etiquetas de AWS en la Consola de administración de AWS. Puede utilizar el editor de etiquetas para añadir etiquetas a varios recursos de AWS compatibles, incluidos los recursos de RDS, al mismo tiempo. Para obtener más información, consulte el Editor de etiquetas en la Guía del usuario de grupos de recursos de AWS.

Copia de etiquetas a instantáneas de clúster de base de datos

Al crear o restaurar un clúster de base de datos, puede especificar que las etiquetas del clúster se copien en instantáneas del clúster de base de datos. La copia de las etiquetas garantiza que los metadatos para las instantáneas coincidan con los del clúster de base de datos de origen. Además, garantiza que cualquier política de acceso para las instantáneas de base de datos también coincida con las del clúster de base de datos de origen. Las etiquetas no se copian de forma predeterminada.

Puede especificar que las etiquetas se copien en las instantáneas de base de datos para las siguientes acciones:

- Creación de un clúster de base de datos
- Restauración de un clúster de base de datos
- Creación de una réplica de lectura

- Copia de una instantánea de clúster de base de datos

> **Note**
>
> En algunos casos, puede incluir un valor para el parámetro `--tags` del comando create-db-snapshot de la AWS CLI. O puede proporcionar al menos una etiqueta a la operación de la API CreateDBSnapshot. En estos casos, RDS no copia las etiquetas de la instancia de base de datos de origen a la nueva instantánea de base de datos. Esta funcionalidad se aplica incluso si la instancia de base de datos de origen tiene la opción `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) activada.
> Si adopta este enfoque, puede crear una copia de una instancia de base de datos a partir de una instantánea de base de datos. Este enfoque evita añadir etiquetas que no se apliquen a la nueva instancia de base de datos. La instantánea de base de datos se crea mediante el comando `create-db-snapshot` de AWS CLI (o la operación de la API `CreateDBSnapshot` RDS). Después de crear la instantánea de la base de datos, puede añadir etiquetas como se describe más adelante en este tema.

Añadido y eliminación de etiquetas en Amazon RDS

Puede hacer lo siguiente:

- Cree etiquetas al crear un recurso, por ejemplo, al ejecutar el comando `create-db-instance` de la AWS CLI.
- Añada etiquetas a un recurso existente con el comando `add-tags-to-resource`.
- Enumere las etiquetas asociadas a un recurso específico con el comando `list-tags-for-resource`.
- Actualice las etiquetas con el comando `add-tags-to-resource`.
- Elimine las etiquetas de un recurso con el comando `remove-tags-from-resource`.

Los procedimientos siguientes muestran cómo realizar operaciones de etiquetado típicas en recursos relacionados con instancias de base de datos y clústeres de base de datos de Aurora. Tenga en cuenta que las etiquetas se almacenan en caché con fines de autorización. Por este motivo, al añadir o actualizar etiquetas en los recursos de Amazon RDS, pueden pasar varios minutos hasta que las modificaciones estén disponibles.

Consola

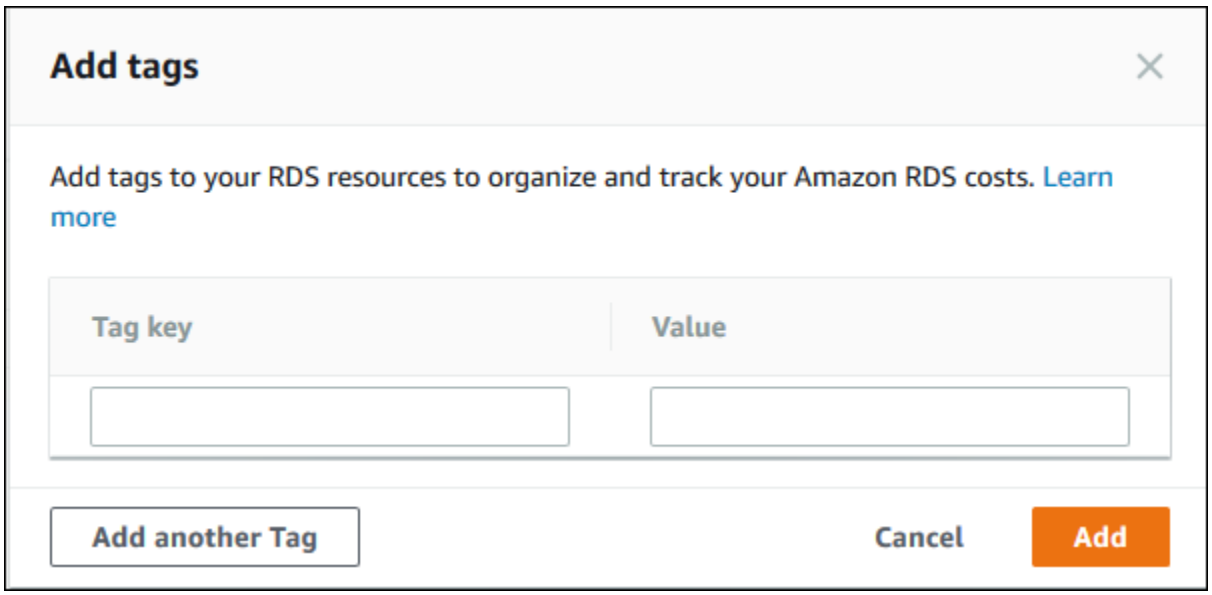
El proceso para etiquetar un recurso de Amazon RDS es similar para todos los recursos. El siguiente procedimiento muestra cómo etiquetar una instancia de base de datos de Amazon RDS.

Para agregar una etiqueta a una instancia de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).

> **Note**
>
> Para filtrar la lista de instancias de base de datos en el panel Databases (Bases de datos), escriba una cadena de texto para Filter databases (Filtrar bases de datos). Solo aparecen instancias de base de datos que contienen la cadena.

3. Seleccione el nombre de la instancia de base de datos que desea etiquetar para mostrar sus detalles.
4. En la sección de detalles, desplácese hasta la sección Tags (Etiquetas).
5. Elija Add (Añadir). Aparece la ventana Add tags (Añadir etiquetas).



6. Escriba un valor para Tag key (Clave de etiqueta) y Value (Valor).
7. Para añadir otra etiqueta, puede elegir Add another Tag (Añadir otra etiqueta) y escribir un valor para Tag key (Clave de etiqueta) y Value (Valor).

Repita este paso tantas veces como sea necesario.

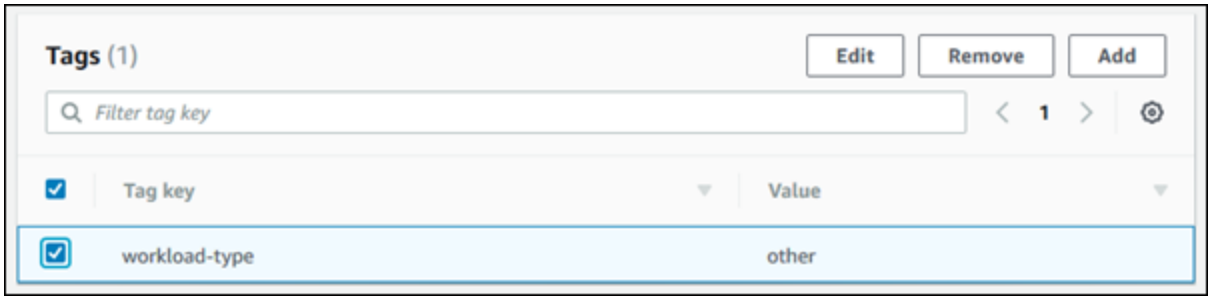
8. Elija Add (Añadir).

Para eliminar una etiqueta de una instancia de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).

> **Note**
>
> Para filtrar la lista de instancias de base de datos en el panel Databases (Bases de datos), escriba una cadena de texto en el cuadro Filter databases (Filtrar bases de datos). Solo aparecen instancias de base de datos que contienen la cadena.

3. Seleccione el nombre de la instancia de base de datos para mostrar sus detalles.
4. En la sección de detalles, desplácese hasta la sección Tags (Etiquetas).
5. Elija la etiqueta desea eliminar.



6. Elija Delete (Eliminar) y después elija (Eliminar) en la ventana Delete tags (Eliminar etiquetas).

AWS CLI

Puede utilizar la para agregar, listar o eliminar etiquetas de una instancia de base de dato AWS CLI.

- Para agregar una o más etiquetas a un recurso de Amazon RDS, utilice el comando `add-tags-to-resource` de la AWS CLI.
- Para ver una lista de las etiquetas de un recurso de Amazon RDS, utilice el comando `list-tags-for-resource` de la AWS CLI.

- Para eliminar una o más etiquetas de un recurso de Amazon RDS, utilice el comando `remove-tags-from-resource` de la AWS CLI.

Para obtener más información acerca de cómo crear el ARN requerido, consulte Creación de un nombre ARN para Amazon RDS.

API de RDS

Puede utilizar la API de Amazon RDS para agregar, listar o eliminar etiquetas de una instancia de base de datos.

- Para añadir una etiqueta a un recurso de Amazon RDS, utilice la operación `AddTagsToResource`.
- Para ver una lista de las etiquetas asignadas a un recurso de Amazon RDS, utilice `ListTagsForResource`.
- Para eliminar etiquetas de un recurso de Amazon RDS, utilice la operación `RemoveTagsFromResource`.

Para obtener más información acerca de cómo crear el ARN requerido, consulte Creación de un nombre ARN para Amazon RDS.

Cuando se trabaja con XML mediante la API de Amazon RDS, las etiquetas utilizan el esquema siguiente:

```
<Tagging>
      <TagSet>
          <Tag>
              <Key>Project</Key>
              <Value>Trinity</Value>
          </Tag>
          <Tag>
              <Key>User</Key>
              <Value>Jones</Value>
          </Tag>
      </TagSet>
  </Tagging>
```

La tabla siguiente proporciona una lista de las etiquetas XML permitidas y sus características. Los valores de `Key` y `Value` distinguen entre mayúsculas y minúsculas. Por ejemplo, `project=Trinity` y `PROJECT=Trinity` son dos etiquetas diferentes.

Elemento de etiquetado	Descripción
TagSet	Los conjuntos de etiquetas contienen todas las etiquetas asignadas a un recurso de Amazon RDS. Solo puede haber un conjunto de etiquetas por recurso. Solo puede trabajar con conjuntos de etiquetas a través de la API de Amazon RDS.
Tag	Las etiquetas son pares clave-valor que define el usuario. En un conjunto de etiquetas puede haber entre 1 y 50 etiquetas.
Key	La clave es el nombre obligatorio de la etiqueta. Para conocer las restricciones, consulte Estructura de etiquetas en Amazon RDS. El valor de la cadena puede tener una longitud de entre 1 y 128 caracteres Unicode y no puede llevar el prefijo `aws:` ni `rds:`. La cadena solo puede contener el conjunto de letras, dígitos y espacio en blanco Unicode, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Las claves deben ser únicas dentro de un conjunto de etiquetas. Por ejemplo, en un conjunto de etiquetas no puede haber claves iguales pero con valores diferentes, como proyecto/Trinity y proyecto/Xanadu.
Valor	El valor es la parte opcional de la etiqueta. Para conocer las restricciones, consulte Estructura de etiquetas en Amazon RDS. El valor de la cadena puede tener una longitud de entre 1 y 256 caracteres Unicode y no puede llevar el prefijo `aws:` ni `rds:`. La cadena solo puede contener el conjunto de letras, dígitos y espacio en blanco Unicode, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Los valores no deben ser únicos dentro de un conjunto de etiquetas y también pueden ser nulos. Por ejemplo, puede tener un par clave-valor en un conjunto de etiquetas en proyecto/Trinity y centro-de-costos/Trinity.

Tutorial: Uso de etiquetas para especificar qué clústeres de base de datos de Aurora se deben detener

Supongamos que crea una serie de clústeres de Aurora base de datos en un entorno de desarrollo o prueba. Necesita conservar todos estos clústeres durante varios días. Algunos de los clústeres ejecutan pruebas durante la noche. Otros clústeres se pueden detener durante la noche y comenzar de nuevo al día siguiente. En el ejemplo siguiente se muestra cómo asignar una etiqueta a los clústeres adecuados para detenerse durante la noche. A continuación, el ejemplo muestra cómo un script puede detectar qué clústeres tienen esa etiqueta y, entonces, detenerlos. En este ejemplo, la parte del valor del par clave-valor no importa. La presencia de la `stoppable` etiqueta significa que el clúster tiene esta propiedad definida por el usuario.

Para especificar qué clústeres de base de datos de Aurora se deben detener

1. Determine el ARN de un clúster que queremos designar como detenible.

   Los comandos y las API para etiquetar funcionan con los ARN. De esta forma, pueden funcionar sin problemas en regiones de AWS, cuentas de AWS y diferentes tipos de recursos que pueden tener nombres cortos idénticos. Puede especificar el ARN en lugar del ID de clúster en los comandos CLI que operan en clústeres. Sustituya el nombre de su propio clúster por *`dev-test-clúster`*. En comandos posteriores que utilicen parámetros ARN, sustituya el ARN de su propio clúster. El ARN incluye el propio ID de cuenta de AWS y el nombre de la región de AWS donde se encuentra el clúster.

   ```
   $ aws rds describe-db-clusters --db-cluster-identifier dev-test-cluster \
     --query "*[].{DBClusterArn:DBClusterArn}" --output text
   arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
   ```

2. Agregue la etiqueta `stoppable` a este clúster.

   Usted elige el nombre de esta etiqueta. Este enfoque significa que puede evitar diseñar una convención de nomenclatura que codifique toda la información relevante en los nombres. En una convención de este tipo, puede codificar la información en el nombre de la instancia de base de datos o en los nombres de otros recursos. Dado que en este ejemplo se trata la etiqueta como un atributo presente o ausente, se omite la `Value=` parte del `--tags` parámetro.

   ```
   $ aws rds add-tags-to-resource \
     --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster \
   ```

```
  --tags Key=stoppable
```

3. Confirme que la etiqueta está presente en el clúster.

   Estos comandos recuperan la información de etiqueta para el clúster en formato JSON y en texto plano separado por tabulaciones.

```
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
{
    "TagList": [
        {
            "Key": "stoppable",
            "Value": ""

        }
    ]
}
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster --output
 text
TAGLIST stoppable
```

4. Para detener todos los clústeres designados como `stoppable`, prepare una lista de todos los clústeres. Recorra la lista y compruebe si cada clúster está etiquetado con el atributo correspondiente.

   Este ejemplo de Linux utiliza scripts shell a fin de guardar la lista de las ARN de clúster en un archivo temporal y, a continuación, ejecutar comandos de CLI para cada clúster.

```
$ aws rds describe-db-clusters --query "*[].[DBClusterArn]" --output text >/tmp/
cluster_arns.lst
$ for arn in $(cat /tmp/cluster_arns.lst)
do
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 'TAGLIST\tstoppable')"
  if [[ ! -z "$match" ]]
  then
      echo "Cluster $arn is tagged as stoppable. Stopping it now."
# Note that you can specify the full ARN value as the parameter instead of the
 short ID 'dev-test-cluster'.
      aws rds stop-db-cluster --db-cluster-identifier $arn
  fi
```

```
done

Cluster arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster is tagged as
 stoppable. Stopping it now.
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "us-east-1e",
            "us-east-1c",
            "us-east-1d"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "dev-test-cluster",
        ...
```

Puede ejecutar un script como este al final de cada día para asegurarse de que los clústeres no esenciales se detienen. También puede programar un trabajo al usar una utilidad como `cron` para realizar dicha comprobación cada noche. Por ejemplo, puede hacer esto en caso de que algunos clústeres de base de datos siguieran ejecutándose por error. Aquí, puede ajustar el comando que prepara la lista de clústeres para comprobar.

El siguiente comando produce una lista de los clústeres, pero sólo los que están en `available` estado. El script puede ignorar los clústeres que ya están detenidos, porque tendrán valores de estado diferentes, como `stopped` o `stopping`.

```
$ aws rds describe-db-clusters \
  --query '*[].{DBClusterArn:DBClusterArn,Status:Status}|[?Status == `available`]|[].
{DBClusterArn:DBClusterArn}' \
  --output text
arn:aws:rds:us-east-1:123456789:cluster:cluster-2447
arn:aws:rds:us-east-1:123456789:cluster:cluster-3395
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
arn:aws:rds:us-east-1:123456789:cluster:pg2-cluster
```

> **Tip**
>
> Puede utilizar la asignación de etiquetas y la búsqueda de clústeres con esas etiquetas para reducir costos de otras maneras. Por ejemplo, veamos este escenario con clústeres de base de datos de Aurora utilizadas para el desarrollo y las pruebas. En este caso puede

designar algunos clústeres para que se eliminen al final de cada día o que solo se eliminen las instancias de base de datos de lector. O puede designar algunos para que sus instancias de base de datos se cambien a clases de instancias de base de datos pequeñas durante los períodos de uso escaso previsto.

Nombres de recursos de Amazon (ARN) en Amazon RDS

Cada recurso que se crea en Amazon Web Services se identifica de forma inequívoca mediante un nombre de recurso de Amazon (ARN). Para determinadas operaciones de Amazon RDS, debe identificar de forma inequívoca un recurso de Amazon RDS mediante su ARN. Por ejemplo, al crear una réplica de lectura de una instancia de base de datos de RDS, debe proporcionar el ARN de la instancia de base de datos de origen.

Para obtener información sobre la creación de un ARN y la obtención de un ARN existente, consulte los siguientes temas.

Temas

- Creación de un nombre ARN para Amazon RDS
- Obtención de un ARN existente para Amazon RDS

Creación de un nombre ARN para Amazon RDS

Cada recurso que se crea en Amazon Web Services se identifica de forma inequívoca mediante un nombre de recurso de Amazon (ARN). Puede crear un ARN para un recurso de Amazon RDS utilizando la siguiente sintaxis.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Para los recursos de clústeres globales, el ARN no incluye una Región de AWS: `arn:aws:rds::<account number>:global-cluster:<name>`. Los ARN de los clústeres globales no aparecen en la Consola de administración de AWS.

Nombre de la región	Región	Punto de conexión	Protocolo	
Este de EE. UU. (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	

Nombre de la región	Región	Punto de conexión	Protocolo
Este de EE. UU. (Norte de Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
		rds-fips.us-east-1.api.aws	HTTPS
		rds-fips.us-east-1.amazonaws.com	HTTPS
		rds.us-east-1.api.aws	HTTPS
Oeste de EE. UU. (Norte de California)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
Oeste de EE. UU. (Oregón)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
África (Ciudad del Cabo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
Asia-Pacífico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
Asia-Pacífico (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com rds.ap-south-2.api.aws	HTTPS HTTPS
Asia-Pacífico (Yakarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com rds.ap-southeast-3.api.aws	HTTPS HTTPS
Asia-Pacífico (Malasia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Asia-Pacífico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS
Asia-Pacífico (Bombay)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS
Asia-Pacífico (Nueva Zelanda)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS
Asia-Pacífico (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS
Asia-Pacífico (Seúl)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
Asia-Pacífico (Singapur)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS
		rds.ap-southeast-1.api.aws	HTTPS
Asia-Pacífico (Sídney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
		rds.ap-southeast-2.api.aws	HTTPS
Asia-Pacífico (Taipéi)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
Asia-Pacífico (Tailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
Asia-Pacífico (Tokio)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
Canadá (centro)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
Oeste de Canadá (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
Europa (Fráncfort)	eu-central-1	rds.eu-central-1.amazonaws.com rds.eu-central-1.api.aws	HTTPS HTTPS
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com rds.eu-west-1.api.aws	HTTPS HTTPS
Europa (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com rds.eu-west-2.api.aws	HTTPS HTTPS
Europa (Milán)	eu-south-1	rds.eu-south-1.amazonaws.com rds.eu-south-1.api.aws	HTTPS HTTPS
Europa (París)	eu-west-3	rds.eu-west-3.amazonaws.com rds.eu-west-3.api.aws	HTTPS HTTPS
Europa (España)	eu-south-2	rds.eu-south-2.amazonaws.com rds.eu-south-2.api.aws	HTTPS HTTPS
Europa (Estocolmo)	eu-north-1	rds.eu-north-1.amazonaws.com rds.eu-north-1.api.aws	HTTPS HTTPS
Europa (Zúrich)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS
Israel (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS

Nombre de la región	Región	Punto de conexión	Protocolo
México (central)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Medio Oriente (Baréin)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS
Medio Oriente (EAU)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS
América del Sur (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (Este de EE. UU.)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (Oeste de EE.UU.)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

En la siguiente tabla se muestra el formato que debe utilizar al crear un ARN para un tipo de recurso concreto de Amazon RDS.

Tipo de recurso	Formato de ARN
Instancia de base de datos	arn:aws:rds:*<region>*:*<account>* :db:*<name>*

Tipo de recurso	Formato de ARN
	Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
Clúster de base de datos	arn:aws:rds:*<region>*:*<account>* :clúster:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
Suscripción a eventos	arn:aws:rds:*<region>*:*<account>* :es:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB Parameter Group (Grupo de parámetros de base de datos)	arn:aws:rds:*<region>*:*<account>* :pg:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Grupo de parámetros de clúster de base de datos	arn:aws:rds:*<region>*:*<account>* :clúster-pg:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`

Tipo de recurso	Formato de ARN
Instancia de base de datos reservada	arn:aws:rds:*<region>*:*<account>* :ri:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :ri:my-reserved-postgresql`
Grupo de seguridad de base de datos	arn:aws:rds:*<region>*:*<account>* :secgrp:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :secgrp:my-public`
Instantánea de base de datos automatizada	arn:aws:rds:*<region>*:*<account>* :snapshot:rds:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :snapshot:rds: my-mysql-db-2019-07-22-07-23`
Instantánea de clúster de base de datos automatizada	arn:aws:rds:*<region>*:*<account>* :clúster-snapshot:rds:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot:rds:  my-aurora-cluster-2019-07-22-16-16`
Instantánea de base de datos manual	arn:aws:rds:*<region>*:*<account>* :snapshot:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`

Tipo de recurso	Formato de ARN
Instantánea de un clúster de base de datos manual	arn:aws:rds:*<region>*:*<account>* :clúster-snapshot:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`
Grupo de subred de base de datos	arn:aws:rds:*<region>*:*<account>* :subgrp:*<name>* Por ejemplo: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`
Clúster global	arn:aws:rds::*<account>* :global-cluster:*<name>* Por ejemplo: `arn:aws:rds:: 123456789012 :global-cluster: my-aurora-global-cluster-1`

Obtención de un ARN existente para Amazon RDS

Puede obtener el ARN de un recurso de RDS mediante la Consola de administración de AWS, la AWS Command Line Interface (AWS CLI) o la API de RDS.

Consola

Para obtener un ARN desde la Consola de administración de AWS, vaya al recurso cuyo ARN desea obtener y consulte los detalles de ese recurso.

Por ejemplo, puede obtener el ARN de un clúster de base de datos desde la pestaña Configuración de los detalles del clúster de base de datos.
ARN del clúster de base de datos.

AWS CLI

Para obtener el ARN de un recurso concreto de RDS desde la AWS CLI, se utiliza el comando `describe` con dicho recurso. En la siguiente tabla se muestran los distintos comandos de AWS CLI, junto con la propiedad ARN que se utiliza con el comando para obtener un ARN.

AWS CLI command	Propiedad ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-clúster-parameter-groups	DBclústerParameterGroupArn
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	ReservedDBInstanceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clústers	DBclústerArn
describe-db-clúster-snapshots	DBclústerSnapshotArn

Por ejemplo, el siguiente comando de la AWS CLI obtiene el ARN de una instancia de base de datos.

Example

Para Linux, macOS o:Unix

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
```

```
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

En:Windows

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

El resultado de ese comando es como el siguiente:

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

API de RDS

Para obtener el ARN de un recurso concreto de RDS, puede llamar a las siguientes operaciones de la API de RDS y utilizar las propiedades ARN que se muestran a continuación.

Operación de la API de RDS	Propiedad ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBclústerParameterGroups	DBclústerParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeDBSecurityGroups	DBSecurityGroupArn
DescribeDBSnapshots	DBSnapshotArn

Operación de la API de RDS	Propiedad ARN
DescribeEvents	SourceArn
DescribeReservedDBInstances	ReservedDBInstanceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeDBclústers	DBclústerArn
DescribeDBclústerSnapshots	DBclústerSnapshotArn

Actualizaciones de Amazon Aurora

Amazon Aurora publica de forma periódica actualizaciones. Las actualizaciones se aplican a clústeres de base de datos de Amazon Aurora durante los períodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del periodo de mantenimiento para el clúster de la base de datos, así como del tipo de actualización. Las actualizaciones exigen el reinicio de la base de datos, por lo que se producen habitualmente entre 20 y 30 segundos de inactividad. Tras esta inactividad, podrá volver a utilizar los clústeres de la base de datos. Puede ver o cambiar la configuración del periodo de mantenimiento desde la Consola de administración de AWS.

> **Note**
>
> El tiempo necesario para reiniciar la instancia de base de datos depende del proceso de recuperación de fallos, la actividad de la base de datos en el momento del reinicio y el comportamiento del motor de base de datos específico. Para mejorar el tiempo de reinicio, recomendamos reducir la actividad de la base de datos tanto como sea posible durante el proceso de reinicio. Al reducirse la actividad de la base de datos, se reduce la actividad de restauración para las transacciones en tránsito.

Para obtener información sobre las actualizaciones del sistema operativo de Amazon Aurora, consulte Actualizaciones del sistema operativo para clústeres de base de datos de Aurora.

Algunas de las actualizaciones son específicas de un motor de base de datos admitido por Aurora. Para obtener más información acerca de las actualizaciones del motor de base de datos, consulte la siguiente tabla.

Motor de base de datos	Actualizaciones
MySQL de Amazon Aurora	Consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL
PostgreSQL de Amazon Aurora	Consulte Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL

Identificación de su versión de Amazon Aurora

Amazon Aurora incluye algunas características que son generales para Aurora y están disponibles para todos los clústeres de base de datos Aurora. Aurora incluye otras características específicas de un motor de base de datos en particular compatible con Aurora. Estas características están disponibles solo para aquellos clústeres de base de datos Aurora que usan ese motor de base de datos, como Aurora PostgreSQL.

Una instancia de base de datos de Aurora tiene dos números de versión: el número de versión de Aurora y el número de versión del motor de la base de datos de Aurora: Los números de versión de Aurora usan el siguiente formato.

```
<major version>.<minor version>.<patch version>
```

Para obtener el número de versión de Aurora a partir de una instancia de base de datos de Aurora mediante un motor de base de datos determinado, use una de las siguientes consultas.

Motor de base de datos	Consultas
MySQL de Amazon Aurora	`SELECT AURORA_VERSION();` `SHOW @@aurora_version;`
PostgreSQL de Amazon Aurora	`SELECT AURORA_VERSION();`

Soporte extendido de Amazon RDS con Amazon Aurora

El soporte extendido de RDS le permite seguir ejecutando una base de datos en una versión principal del motor después de la fecha de finalización del soporte estándar de Aurora por un costo adicional.

Solo puede inscribir una base de datos en el soporte extendido de RDS si habilita el soporte extendido de RDS al crear o restaurar una instancia de base de datos por primera vez. No puede actualizar el estado de inscripción de soporte extendido de RDS en las instancias de base de datos existentes a menos que las restaure.

Si ha habilitado el soporte extendido de RDS durante la creación o la restauración de una instancia de base de datos, después del final de Aurora de la fecha de soporte estándar, Amazon Aurora inscribirán automáticamente la instancia de base de datos en el soporte extendido de RDS. La inscripción automática en el soporte extendido de RDS no cambia el motor de la base de datos ni afecta al tiempo de actividad ni al rendimiento de la instancia de base de datos.

Soporte extendido de RDS ofrece las siguientes actualizaciones y soporte técnico:

- Actualizaciones de seguridad para las CVE críticas y altas para la instancia de base de datos o clúster de base de datos, incluido el motor de base de datos
- Correcciones de errores y parches para problemas críticos
- La posibilidad de abrir casos de soporte y recibir ayuda para la solución de problemas según el Acuerdo de nivel de servicio de Amazon RDS

Esta oferta de pago le da más tiempo para llevar a cabo la actualización a una versión principal del motor compatible. Por ejemplo, la fecha de finalización del soporte estándar de Aurora para la versión 2 de Aurora MySQL es el 31 de octubre de 2024. Sin embargo, no está preparado para llevar a cabo la actualización manual a la versión 3 de Aurora MySQL antes de esa fecha. En este caso, Amazon Aurora inscribe automáticamente el clúster en el Soporte extendido de RDS el 31 de octubre de 2024 y podrá seguir ejecutando la versión 2 de Aurora MySQL. A partir del 1 de diciembre de 2024, Amazon Aurora le cobrará automáticamente por el Soporte extendido de RDS.

El soporte extendido de RDS está disponible durante un máximo de 3 años después de la fecha de fin de vida útil de la comunidad para una versión principal del motor (3 años y 4 meses para Aurora MySQL versión 2). Transcurrido este tiempo, si no ha actualizado la versión principal del motor a una versión compatible, Amazon Aurora actualizará automáticamente la versión principal del motor. Se

recomienda que lleva a cabo la actualización a una versión principal del motor compatible lo antes posible.

Para obtener más información sobre las fechas de fin del soporte estándar de Aurora y las fechas de finalización del soporte extendido de RDS, consulte Release calendar for Aurora MySQL major versions y Release calendar for Aurora PostgreSQL major versions.

Temas

- Información general del Soporte extendido de Amazon RDS
- Precios del Soporte extendido de Amazon RDS
- Versiones con el Soporte extendido de Amazon RDS
- Amazon Aurora y las responsabilidades del cliente con el Soporte extendido de Amazon RDS
- Creación de un clúster de base de datos de Aurora con Soporte extendido de Amazon RDS
- Visualización de la inscripción de sus clústeres de base de datos Aurora o clústeres globales en el Soporte extendido de Amazon RDS
- Visualización de las fechas de soporte de las versiones del motor en soporte extendido de Amazon RDS
- Restauración de un clúster de base de datos de Aurora con Soporte extendido de Amazon RDS

Información general del Soporte extendido de Amazon RDS

Tras la fecha de finalización del soporte estándar de Aurora, si no ha deshabilitado el soporte extendido de RDS durante la creación o restauración de las instancias de base de datos, Amazon Aurora las inscribirá automáticamente en el soporte extendido de RDS. Aurora actualiza automáticamente su instancia de base de datos a la versión secundaria más reciente publicada antes de la fecha de fin del soporte estándar de Aurora, si aún no está ejecutando esa versión. Amazon Aurora no actualizará la versión secundaria hasta después de la fecha de finalización del soporte estándar de Aurora para su versión principal del motor.

Puede crear nuevas bases de datos con las versiones principales del motor que hayan alcanzado la fecha de finalización del soporte estándar de Aurora. Aurora inscribe automáticamente estas nuevas bases de datos en el Soporte extendido de RDS y le cobra por esta oferta.

Si realiza una actualización a un motor para el que sigue vigente el soporte estándar de Aurora antes de la fecha de finalización del soporte estándar de Aurora, Amazon Aurora no inscribirá el motor en el Soporte extendido de RDS.

Si intenta restaurar una instantánea de una base de datos compatible con un motor cuya fecha de finalización del soporte estándar de Aurora haya pasado, pero que no está inscrito en el Soporte extendido de RDS, Amazon Aurora intentará actualizar la instantánea para que sea compatible con la última versión del motor cuyo soporte estándar de Aurora sigue vigente. Si se produce un error en la restauración, Amazon Aurora inscribirá automáticamente el motor en el Soporte extendido de RDS con una versión que sea compatible con la instantánea.

Puede finalizar la inscripción en el Soporte extendido de RDS en cualquier momento. Para finalizar la inscripción, actualice cada motor inscrito a una versión más reciente cuyo soporte estándar de Aurora siga vigente. La finalización de la inscripción en el Soporte extendido de RDS entrará en vigor el día en que complete una actualización a una versión más reciente del motor cuyo soporte estándar de Aurora siga vigente.

Para obtener más información sobre las fechas de fin del soporte estándar de Aurora y las fechas de finalización del soporte extendido de RDS, consulte Release calendar for Aurora MySQL major versions y Release calendar for Aurora PostgreSQL major versions.

Precios del Soporte extendido de Amazon RDS

Se incurrirá en cargos por todos los motores inscritos en el Soporte extendido de RDS a partir del día siguiente de la fecha de finalización del soporte estándar de Aurora. Para conocer la fecha de finalización del soporte estándar de Aurora, consulte Versiones principales de Amazon Aurora.

El cargo adicional del Soporte extendido de RDS se detiene automáticamente cuando realiza una de las siguientes acciones:

- Actualizar a una versión de motor incluida en el soporte estándar.
- Eliminar la base de datos en la que se ejecuta una versión principal pasada la fecha de finalización del soporte estándar de Aurora.

Los cargos se reiniciarán si la versión del motor de destino se incluye en el Soporte extendido de RDS en el futuro.

Por ejemplo, Aurora PostgreSQL 11 entra en el Soporte extendido el 1 de marzo de 2024, pero los cargos no empiezan a cobrarse hasta el 1 de abril de 2024. Actualiza la base de datos Aurora PostgreSQL 11 a Aurora PostgreSQL 12 el 30 de abril de 2024. Solo se le cobrarán 30 días de soporte extendido en Aurora PostgreSQL 11. Seguirá ejecutando Aurora PostgreSQL 12 en esta

instancia de base de datos después de la fecha de finalización del soporte estándar de RDS, el 28 de febrero de 2025. Su base de datos volverá a incurrir en cargos de Soporte extendido de RDS a partir del 1 de marzo de 2025.

Para obtener más información, consulte Precios de Amazon Aurora.

Prevención de cargos del Soporte extendido de Amazon RDS

Puede evitar que se le cobre por el Soporte extendido de RDS impidiendo que Aurora cree o restaure un clúster de base de datos de Aurora o un clúster global después de la fecha de finalización del soporte estándar de Aurora. Para ello, utilice la AWS CLI o la API de RDS.

En la AWS CLI, especifique `open-source-rds-extended-support-disabled` para la opción `--engine-lifecycle-support`. En la API de RDS, especifique `open-source-rds-extended-support-disabled` para el parámetro `LifeCycleSupport`. Para obtener más información, consulte Creación de un clúster de base de datos de Aurora o un clúster global o Restauración de un clúster de base de datos de Aurora o un clúster global.

Versiones con el Soporte extendido de Amazon RDS

El soporte extendido de RDS está disponible Aurora MySQL y para Aurora PostgreSQL. Para obtener más información, consulte Versiones principales de Amazon Aurora.

El Soporte extendido de RDS solo está disponible en determinadas versiones secundarias. Las versiones secundarias solo se indican como compatibles con la compatibilidad extendida con RDS después de que las versiones principales alcancen las fechas de fin de vida de la comunidad. Para obtener más información, consulte Calendario de lanzamiento de Aurora MySQL en las Notas de la versión de Aurora MySQL y Calendario de lanzamiento de Aurora PostgreSQL en las Notas de la versión de Aurora PostgreSQL.

El Soporte extendido de RDS solo está disponible en Aurora Serverless v2. No está disponible en Aurora Serverless v1.

También puede ver información sobre las fechas de soporte de las versiones del motor mediante la AWS CLI o la API de RDS. Para obtener más información, consulte Visualización de las fechas de soporte de las versiones del motor en soporte extendido de Amazon RDS.

Amazon Aurora y las responsabilidades del cliente con el Soporte extendido de Amazon RDS

El siguiente contenido describe las responsabilidades de Amazon Aurora y sus responsabilidades con el Soporte extendido de RDS.

Temas

- Responsabilidades de Amazon Aurora
- Sus responsabilidades

Responsabilidades de Amazon Aurora

Tras la fecha de finalización del soporte estándar de Aurora, Amazon Aurora proporcionará parches, correcciones de errores y actualizaciones para los motores inscritos en el Soporte extendido de RDS. Esto ocurrirá durante un máximo de 3 años o hasta que deje de utilizar los motores, lo que ocurra primero.

Los parches serán para las CVE críticas y altas, según se definen en las puntuaciones de gravedad de CVSS de la National Vulnerability Database (NVD). Para obtener más información, consulte las métricas de vulnerabilidad.

Sus responsabilidades

Usted es responsable de aplicar los parches, las correcciones de errores y las actualizaciones proporcionadas para los clústeres de bases de datos de Aurora o clústeres globales inscritos en el Soporte extendido de RDS. Amazon Aurora se reserva el derecho de cambiar, sustituir o retirar dichos parches, correcciones de errores y actualizaciones en cualquier momento. Si se necesita un parche para solucionar problemas de seguridad o estabilidad críticos, Amazon Aurora se reserva el derecho de actualizar sus clústeres de bases de datos de Aurora o clústeres globales con el parche, o bien de solicitarle que instale el parche.

También es responsable de actualizar el motor a una versión más reciente antes de la fecha de finalización del Soporte extendido de RDS. La fecha de finalización del Soporte extendido de RDS suele ser 3 años después de la fecha de finalización del fin de la vida útil de la comunidad. . Para conocer la fecha de fin del Soporte extendido de RDS para la versión principal del motor de bases de datos, consulte Versiones principales de Amazon Aurora.

Si no actualiza el motor, después de la fecha de finalización del soporte extendido de RDS, Amazon Aurora intentará actualizar el motor a una versión más reciente del motor que admita el soporte estándar Aurora. Si se produce un fallo de actualización, Amazon Aurora se reserva el derecho a eliminar la clúster de base de datos de Aurora o clúster global que ejecuta el motor una vez pasada la fecha de finalización del soporte estándar de Aurora. Sin embargo, antes de hacerlo, Amazon Aurora conservará los datos de ese motor.

Creación de un clúster de base de datos de Aurora con Soporte extendido de Amazon RDS

Al crear un clúster de base de datos de Aurora o un clúster global, seleccione Habilitar el Soporte extendido de RDS en la consola, o utilice la opción Soporte extendido en la AWS CLI o el parámetro de API de RDS. Al inscribir un clúster de base de datos de Aurora o un clúster global en el Soporte extendido de Amazon RDS, se inscribe permanentemente en el Soporte extendido de RDS durante toda la vida de el clúster de base de datos de Aurora o el clúster global.

Si usa la consola, debe seleccionar Activar el soporte extendido de RDS. La opción no está seleccionada de forma predeterminada.

Si utiliza la AWS CLI o la API de RDS y no especifica la opción del Soporte extendido de RDS, Amazon RDS usará de manera predeterminada el Soporte extendido de RDS. Al automatizar usando CloudFormation u otros servicios, este comportamiento predeterminado mantiene la disponibilidad de la base de datos después de la fecha del fin del soporte estándar de Aurora.

Puede evitar la inscripción en el Soporte extendido de RDS mediante la AWS CLI o la API de RDS para crear un clúster de base de datos Aurora o un clúster global.

Temas

- Comportamiento del Soporte extendido de RDS
- Observaciones sobre el Soporte extendido de RDS
- Creación de un clúster de base de datos de Aurora o un clúster global con Soporte extendido de RDS

Comportamiento del Soporte extendido de RDS

En la siguiente tabla se resume lo que ocurre cuando una versión principal del motor llega al final del soporte estándar de Aurora.

Estado del Soporte extendido de RDS*	Comportamiento
Habilitado	Amazon RDS le cobra por el Soporte extendido de RDS.
Deshabilitado	Amazon RDS actualiza su clúster de base de datos de Aurora o clúster global a una versión del motor compatible. Esta actualización se lleva a cabo en la fecha de fin del soporte estándar de Aurora o poco después.

* En la consola de RDS, el estado del Soporte extendido de RDS aparece como Sí o No. En la AWS CLI o en la API de RDS, el estado del Soporte extendido de RDS aparece como `open-source-rds-extended-support` o `open-source-rds-extended-support-disabled`.

Observaciones sobre el Soporte extendido de RDS

Antes de crear un clúster de base de datos de Aurora o un clúster global, tenga en cuenta lo siguiente:

- Una vez pasada la fecha del fin del soporte estándar de Aurora, puede impedir que se cree un nuevo clúster de base de datos de Aurora o un nuevo clúster global y evitar los cargos del Soporte extendido de RDS. Para ello, utilice la AWS CLI o la API de RDS. En la AWS CLI, especifique `open-source-rds-extended-support-disabled` para la opción `--engine-lifecycle-support`. En la API de RDS, especifique `open-source-rds-extended-support-disabled` para el parámetro `LifeCycleSupport`. Si especifica `open-source-rds-extended-support-disabled` y ha pasado la fecha del fin del soporte estándar de Aurora, siempre se producirá un error al crear un clúster de base de datos de Aurora o un clúster global.
- El Soporte extendido de RDS se establece en el nivel de clúster. Los miembros de un clúster siempre tendrán la misma configuración para el Soporte extendido de RDS en la consola de RDS, `--engine-lifecycle-support` en la AWS CLI y `EngineLifecycleSupport` en la API de RDS.

Para obtener más información, consulte Versiones de Amazon Aurora.

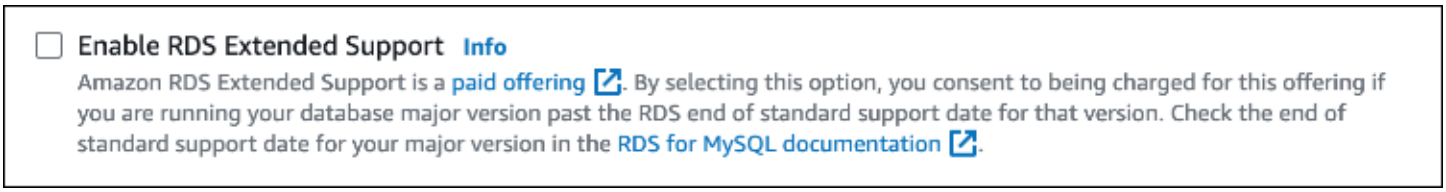
Creación de un clúster de base de datos de Aurora o un clúster global con Soporte extendido de RDS

Puede crear un clúster de base de datos de Aurora o un clúster global con una versión del Soporte extendido de RDS que utilice la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Al crear un clúster de base de datos de Aurora o un clúster global, seleccione Habilitar el Soporte extendido de RDS en la sección Opciones del motor. Esta configuración no está seleccionada de forma predeterminada.

La siguiente imagen muestra la configuración Habilitar el Soporte extendido de RDS:



AWS CLI

Cuando ejecute el comando create-db-cluster o create-global-cluster de la AWS CLI, seleccione el Soporte extendido de RDS especificando `open-source-rds-extended-support` para la opción `--engine-lifecycle-support`. Esta opción está configurada en `open-source-rds-extended-support` de forma predeterminada.

Para evitar la creación de un nuevo clúster de base de datos de Aurora o un clúster global después de la fecha del fin del soporte estándar de Aurora, especifique `open-source-rds-extended-support-disabled` para la opción `--engine-lifecycle-support`. De este modo, evitará los cargos del Soporte extendido de RDS asociados.

API de RDS

Cuando utilice la operación API CreateDBCluster o CreateGlobalCluster de Amazon RDS, seleccione el Soporte extendido de RDS poniendo el parámetro `EngineLifecycleSupport` en `open-source-rds-extended-support`. Este parámetro está establecido en `open-source-rds-extended-support` de forma predeterminada.

Para evitar la creación de un nuevo clúster de base de datos de Aurora o un clúster global después de la fecha del fin del soporte estándar de Aurora, especifique `open-source-rds-extended-`

`support-disabled` para el parámetro `EngineLifecycleSupport`. De este modo, evitará los cargos del Soporte extendido de RDS asociados.

Para obtener más información, consulte los temas siguientes:

- Para crear un clúster de base de datos de Aurora, siga las instrucciones para su motor de base de datos en Creación de un clúster de base de datos de Amazon Aurora.
- Para crear un clúster global, siga las instrucciones indicadas en Creación de una base de datos global de Amazon Aurora para su motor de base de datos.

Visualización de la inscripción de sus clústeres de base de datos Aurora o clústeres globales en el Soporte extendido de Amazon RDS

Puede ver la inscripción de sus clústeres de base de datos Aurora o clústeres globales en el Soporte extendido de RDS usando la Consola de administración de AWS, la AWS CLI o la API de RDS.

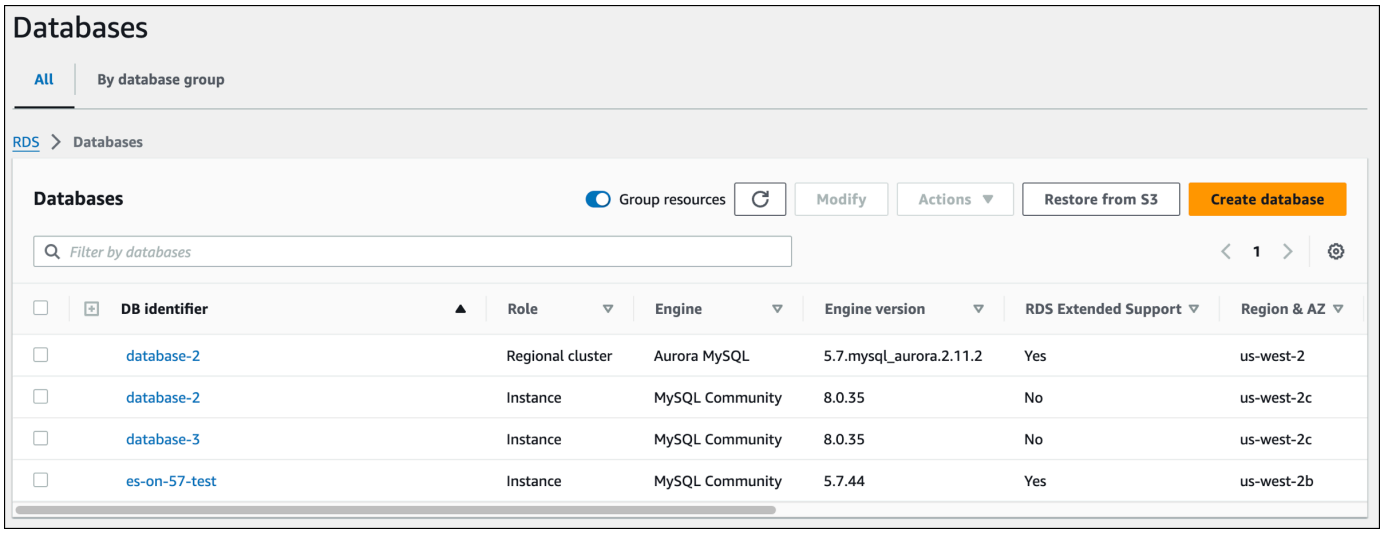
> **Note**
>
> La columna Soporte extendido de RDS de la consola, la opción `-engine-lifecycle-support` en la AWS CLI y el parámetro `EngineLifecycleSupport` de la API de RDS solo indican la inscripción en el Soporte extendido de RDS. Los cargos por el Soporte extendido de RDS solo comienzan cuando la versión de su motor de base de datos ha alcanzado el final del soporte estándar de Aurora. Para obtener más información, consulte Versiones principales de Amazon Aurora.
>
> Por ejemplo, tiene una base de datos de Aurora PostgreSQL 11 que está inscrita en el Soporte extendido de RDS. El 1 de abril de 2024, Amazon RDS comenzó a cobrarle por el Soporte extendido de RDS para esta base de datos. El 31 de julio de 2024, actualizó esta base de datos a Aurora PostgreSQL 12. El estado del Soporte extendido de RDS para esta base de datos permanece habilitado. Sin embargo, los cargos del Soporte extendido de RDS para esta base de datos se suspendieron porque Aurora PostgreSQL 12 aún no había alcanzado el final del soporte estándar de Aurora. Amazon RDS no le cobrará por el Soporte extendido de RDS para esta base de datos hasta el 1 de marzo de 2025, fecha en la que finaliza el soporte estándar de Aurora PostgreSQL 12.

Consola

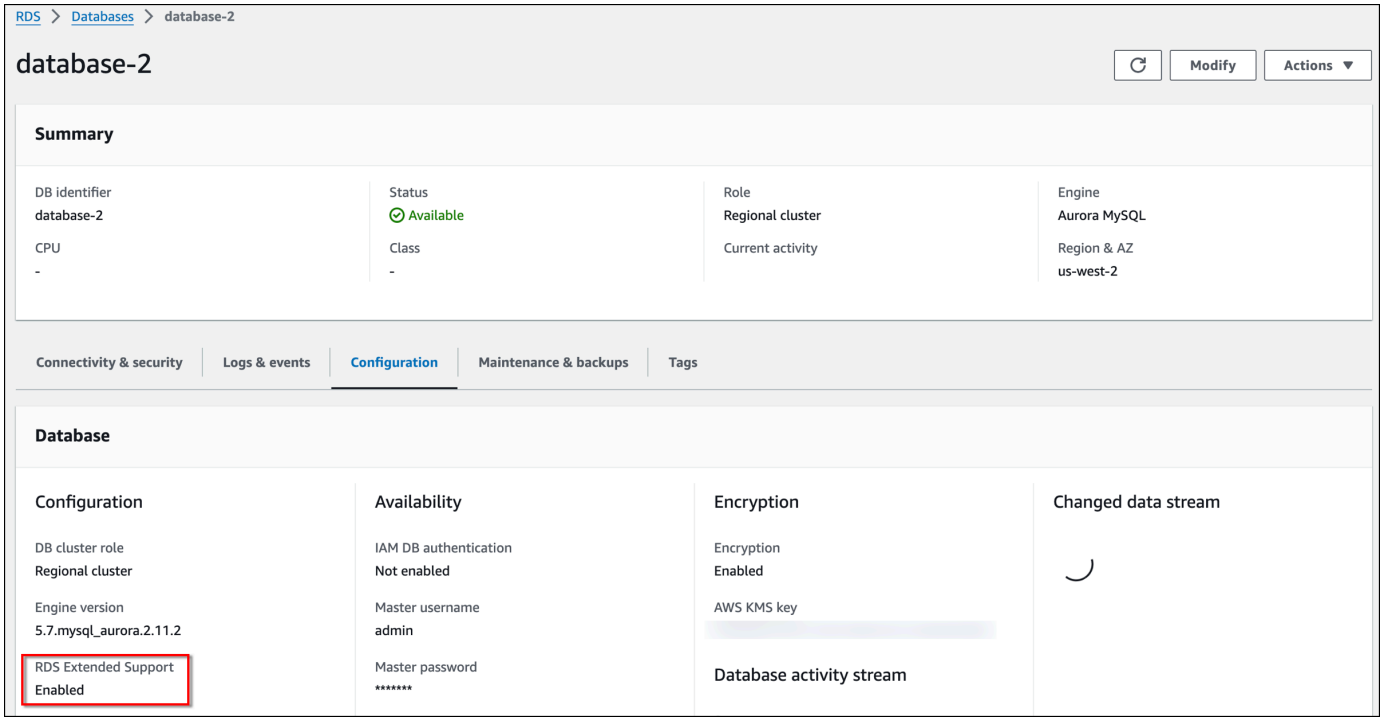
Para ver la inscripción de sus clústeres de base de datos Aurora o clústeres globales en el Soporte extendido de RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos). El valor en Soporte extendido de RDS indica si un clúster de base de datos Aurora o un clúster global están inscritos en el Soporte extendido de RDS. Si no aparece ningún valor, eso significa que el Soporte extendido de RDS no está disponible para su base de datos.

> **Tip**
>
> Si la columna Soporte extendido de RDS no aparece, seleccione el icono Preferencias y, a continuación, active Soporte extendido de RDS.



3. También puede ver la inscripción en la pestaña Configuración de de cada base de datos. Elija una base de datos en el Identificador de la base de datos. En la pestaña Configuración, consulte Soporte extendido para ver si la base de datos está inscrita o no.



AWS CLI

Para ver la inscripción de sus bases de datos en el Soporte extendido de RDS mediante la AWS CLI, ejecute el comando describe-db-clusters o describe-global-clusters.

Si el Soporte extendido de RDS está disponible para una base de datos, la respuesta incluye el parámetro `EngineLifecycleSupport`. El valor `open-source-rds-extended-support` indica que un clúster de base de datos Aurora o un clúster global están inscritos en el Soporte extendido de RDS. El valor `open-source-rds-extended-support-disabled` indica que se ha deshabilitado la inscripción de la clúster de base de datos Aurora o clúster global en el Soporte extendido de RDS.

Ejemplo

El siguiente comando devuelve la información de todos los clústeres de bases de datos Aurora:

```
aws rds describe-db-clusters
```

La siguiente respuesta muestra que un motor Aurora PostgreSQL que se ejecuta en el clúster de base de datos Aurora `database-1` está inscrito en el Soporte extendido de RDS:

```
{
    "DBClusterIdentifier": "database-1",
```

```
    ...
    "Engine": "aurora-postgresql",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

API de RDS

Para ver la inscripción de sus bases de datos en el Soporte extendido de RDS mediante la API de Amazon RDS, utilice la operación DescribeDBClusters o DescribeGlobalClusters.

Si el Soporte extendido de RDS está disponible para una base de datos, la respuesta incluye el parámetro `EngineLifecycleSupport`. El valor `open-source-rds-extended-support` indica que un clúster de base de datos Aurora o un clúster global están inscritos en el Soporte extendido de RDS. El valor `open-source-rds-extended-support-disabled` indica que se ha deshabilitado la inscripción de la clúster de base de datos Aurora o clúster global en el Soporte extendido de RDS.

Visualización de las fechas de soporte de las versiones del motor en soporte extendido de Amazon RDS

Puede ver la información sobre las fechas de soporte de las versiones de motor para los clústeres de base de datos de Aurora o los clústeres globales en el soporte extendido de Amazon RDS mediante la AWS CLI o la API de RDS. Esta información puede ayudarle a planificar las actualizaciones.

Los comandos de la AWS CLI y las operaciones de la API de RDS devuelven las fechas de inicio y finalización del soporte estándar de Aurora y del soporte extendido de RDS. Estas fechas también se pueden encontrar en las tablas de versiones principales del motor. Para obtener más información, consulte Versiones principales de Amazon Aurora.

AWS CLI

Para ver las fechas de inicio y finalización del soporte estándar de Aurora y del soporte extendido de RDS para las versiones principales del motor mediante la AWS CLI, ejecute el comando describe-db-major-engine-versions.

Este comando devuelve los siguientes parámetros relevantes:

- `SupportedEngineLifecycles`: este parámetro es una matriz de objetos que incluye `LifecycleSupportName`, `LifecycleSupportStartDate` y `LifecycleSupportEndDate`.

- `LifecycleSupportName`: este parámetro indica el tipo de soporte en el que se encuentra la versión del motor: soporte estándar para Aurora (`open-source-rds-standard-support`) o soporte extendido de RDS (`open-source-rds-extended-support`).
- `LifecycleSupportStartDate`: este parámetro muestra la fecha de inicio del soporte estándar de Aurora o del soporte extendido de RDS para la versión principal del motor, según el valor de `LifecycleSupportName`.
- `LifecycleSupportEndDate`: este parámetro muestra la fecha de finalización del soporte estándar de Aurora o del soporte extendido de RDS para la versión principal del motor, según el valor de `LifecycleSupportName`.

Ejemplo

El ejemplo de respuesta muestra las fechas de inicio y finalización de los ciclos de vida del motor compatibles `open-source-rds-standard-support` y `open-source-rds-extended-support` para Aurora MySQL versión 2 (MySQL 5.7). Se encuentra disponible el soporte extendido de RDS para la versión 2 de Aurora MySQL (MySQL 5.7).

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "aurora-mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2018-02-06T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-10-31T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-11-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        ...
    ]
}
```

API de RDS

Para ver las fechas de inicio y finalización del soporte estándar de Aurora y del soporte extendido de RDS para las versiones principales del motor mediante la API de RDS, utilice la operación DescribeDBMajorEngineVersions.

Si el soporte extendido de RDS está disponible para una versión del motor, la respuesta incluye el parámetro `SupportedEngineLifeCycles` como una matriz con dos objetos. Un objeto incluye las fechas de inicio y finalización del soporte estándar de Aurora. El segundo objeto incluye las fechas de inicio y finalización del soporte extendido de RDS.

Si el soporte extendido de RDS no está disponible para una versión de motor, la respuesta solo incluye el parámetro `SupportedEngineLifeCycles` como una matriz con un único objeto. Este objeto incluye las fechas de inicio y finalización del soporte estándar de Aurora.

Restauración de un clúster de base de datos de Aurora con Soporte extendido de Amazon RDS

Al restaurar un clúster de base de datos de Aurora o un clúster global, seleccione Habilitar el Soporte extendido de RDS en la consola, o utilice la opción Soporte extendido en la AWS CLI o el parámetro de API de RDS. Al inscribir un clúster de base de datos de Aurora o un clúster global en el Soporte extendido de Amazon RDS, se inscribe permanentemente en el Soporte extendido de RDS durante toda la vida de el clúster de base de datos de Aurora o el clúster global.

El valor predeterminado de la configuración del Soporte extendido de RDS depende de si utiliza la consola, la AWS CLI o la API de RDS para restaurar la base de datos. Si utiliza la consola, no selecciona Habilitar el Soporte extendido de RDS y la versión principal del motor que va a restaurar ha superado la fecha de finalización del soporte estándar para Aurora, Amazon Aurora actualiza automáticamente la instancia de base de datos a una versión de motor más reciente. Si utiliza la AWS CLI o la API de RDS y no especifica la configuración del Soporte extendido de RDS, entonces Amazon RDS usará de manera predeterminada el Soporte extendido de RDS. Al automatizar usando CloudFormation u otros servicios, este comportamiento predeterminado mantiene la disponibilidad de la base de datos después de la fecha del fin del soporte estándar de Aurora. Puede deshabilitar el Soporte extendido de RDS mediante la AWS CLI o la API de RDS.

Temas

- Comportamiento del Soporte extendido de RDS

- Observaciones sobre el Soporte extendido de RDS
- Restauración de un clúster de base de datos de Aurora o un clúster global con Soporte extendido de RDS

Comportamiento del Soporte extendido de RDS

En la siguiente tabla se resume lo que ocurre cuando una versión del motor principal de un clúster de base de datos de Aurora o un clúster global que está restaurando ha alcanzado el final del soporte estándar de Aurora.

Estado del Soporte extendido de RDS*	Comportamiento
Habilitado	Amazon RDS le cobra por el Soporte extendido de RDS.
Deshabilitado	Una vez finalizada la restauración, Amazon RDS actualiza automáticamente el clúster de base de datos de Aurora o el clúster global a una versión de motor más reciente (en un futuro periodo de mantenimiento).

* En la consola de RDS, el estado del Soporte extendido de RDS aparece como Sí o No. En la AWS CLI o en la API de RDS, el estado del Soporte extendido de RDS aparece como `open-source-rds-extended-support` o `open-source-rds-extended-support-disabled`.

Observaciones sobre el Soporte extendido de RDS

Antes de restaurar un clúster de base de datos de Aurora o un clúster global, tenga en cuenta lo siguiente:

- Tras la fecha de finalización del soporte estándar de Aurora, si desea restaurar un clúster de base de datos de Aurora o un clúster global desde Amazon S3, solo podrá hacerlo con la AWS CLI o la API de RDS. Utilice la opción `--engine-lifecycle-support` en el comando restore-db-cluster-from-s3 de la AWS CLI o el parámetro `EngineLifecycleSupport` en la operación API RestoreDBClusterFromS3 de RDS.
- Si desea evitar que Aurora restaure sus bases de datos a las versiones del Soporte extendido de RDS, especifique `open-source-rds-extended-support-disabled` en la AWS CLI o en la API de RDS. De este modo, evitará los cargos del Soporte extendido de RDS asociados.

Si especifica esta configuración, Amazon Aurora actualizará automáticamente la base de datos restaurada a una versión principal compatible más reciente. Si la actualización no pasa las comprobaciones previas, Amazon Aurora volverá de forma segura a la versión del motor del Soporte extendido de RDS. Esta base de datos permanecerá en el modo de Soporte extendido de RDS y Amazon Aurora le cobrará por el Soporte extendido de RDS hasta que actualice manualmente la base de datos.

- El Soporte extendido de RDS se establece en el nivel de clúster. Los miembros de un clúster siempre tendrán la misma configuración para el Soporte extendido de RDS en la consola de RDS, `--engine-lifecycle-support` en la AWS CLI y `EngineLifecycleSupport` en la API de RDS.

Para obtener más información, consulte Versiones de Amazon Aurora.

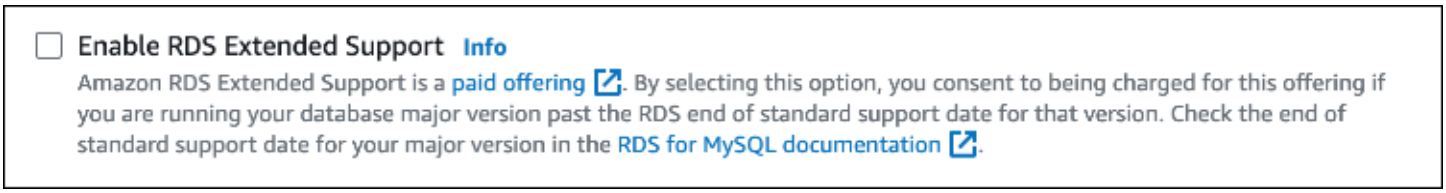
Restauración de un clúster de base de datos de Aurora o un clúster global con Soporte extendido de RDS

Puede restaurar un clúster de base de datos de Aurora o un clúster global con una versión del Soporte extendido de RDS que utilice la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Al restaurar un clúster de base de datos de Aurora o un clúster global, seleccione Habilitar el Soporte extendido de RDS en la sección Opciones del motor. Si no selecciona esta configuración y la versión principal del motor que va a restaurar ha superado la fecha de finalización del soporte estándar de Aurora, Amazon Aurora actualiza automáticamente su clúster de base de datos de Aurora o clúster global a una versión cuyo soporte estándar de Aurora esté vigente.

La siguiente imagen muestra la configuración Habilitar el Soporte extendido de RDS:



AWS CLI

Cuando ejecute el comando restore-db-cluster-from-snapshot de la AWS CLI, seleccione el Soporte extendido de RDS especificando `open-source-rds-extended-support` para la opción `--engine-lifecycle-support`.

Si quiere evitar los cargos asociados con el Soporte extendido de RDS, defina la opción `--engine-lifecycle-support` en `open-source-rds-extended-support-disabled`. Esta opción está configurada en `open-source-rds-extended-support` de forma predeterminada.

También puede especificar este valor con los siguientes comandos de la AWS CLI:

- restore-db-cluster-from-s3
- restore-db-clúster-to-point-in-time

API de RDS

Cuando utilice la operación API RestoreDBClusterFromSnapshot de Amazon RDS, seleccione el Soporte extendido de RDS ajustando el parámetro `EngineLifecycleSupport` en `open-source-rds-extended-support`.

Si quiere evitar los cargos asociados con el Soporte extendido de RDS, defina el parámetro `EngineLifecycleSupport` en `open-source-rds-extended-support-disabled`. Este parámetro está establecido en `open-source-rds-extended-support` de forma predeterminada.

También puede especificar este valor utilizando las siguientes operaciones API de RDS:

- RestoreDBClusterFromS3
- RestoreDBclústerToPointInTime

Para obtener más información sobre cómo restaurar un clúster de base de datos Aurora, siga las instrucciones en Copias de seguridad y restauración de un clúster de base de datos de Amazon Aurora para su motor de base de datos.

Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos

Una implementación azul/verde copia un entorno de base de datos de producción en un entorno de almacenamiento provisional sincronizado e independiente. Con las implementaciones azul/verde de Amazon Aurora, puede realizar cambios en la base de datos en el entorno de almacenamiento provisional sin que eso afecte al entorno de producción. Por ejemplo, puede actualizar la versión principal o secundaria del motor de base de datos o cambiar los parámetros de la base de datos en el entorno de ensayo. Cuando esté listo, puede promocionar el entorno de almacenamiento provisional para que sea el nuevo entorno de la base de datos de producción, con un tiempo de inactividad normalmente inferior a un minuto.

Amazon Aurora crea el entorno de ensayo clonando el volumen de almacenamiento Aurora subyacente en el entorno de producción. El volumen del clúster del entorno de ensayo solo almacena los cambios incrementales que se realizan en ese entorno.

> **Note**
>
> Actualmente, las implementaciones azul/verde solo son compatibles en Aurora MySQL, Aurora PostgreSQL y base de datos global de Aurora. Para conocer la disponibilidad del motor de Amazon RDS, consulte Uso de las implementaciones azules/verdes de Amazon RDS para actualizar las bases de datos en la Guía del usuario de Amazon RDS.

Temas

- Descripción general de las implementaciones azul/verde de Amazon Aurora
- Creación de una implementación azul/verde en Amazon Aurora
- Visualización de una implementación azul/verde en Amazon Aurora
- Cambio de una implementación azul/verde en Amazon Aurora
- Eliminación de una implementación azul/verde en Amazon Aurora

Descripción general de las implementaciones azul/verde de Amazon Aurora

Con las implementaciones azul/verde de Amazon Aurora, puede realizar y probar cambios en las bases de datos antes de implementarlas en un entorno de producción. Una implementación azul/verde crea un área de almacenamiento provisional que copia el entorno de producción. En una implementación azul/verde, el entorno azul es el entorno de producción actual. El entorno verde es el entorno provisional y está sincronizado con el entorno de producción actual.

Puede realizar cambios en el clúster de base de datos de Aurora en un entorno verde sin que eso afecte a las cargas de trabajo de producción. Por ejemplo, puede actualizar la versión principal o secundaria del motor de base de datos o cambiar los parámetros de la base de datos en el entorno de almacenamiento provisional. Puede probar exhaustivamente los cambios en el entorno verde. Cuando esté todo listo, puede realizar una transición a los entornos para hacer que el entorno verde sea el nuevo entorno de producción. La conmutación suele tardar menos de un minuto sin que se produzca una pérdida de datos y sin la necesidad de realizar cambios en la aplicación.

Como el entorno verde es una copia de la topología del entorno de producción, el clúster de base de datos y todas sus instancias de base de datos se copian en la implementación. El entorno verde también incluye las características que utiliza el clúster de base de datos, como las instantáneas del clúster de base de datos, Información sobre rendimiento, monitorización mejorada yAurora Serverless v2.

> **Note**
>
> Las implementaciones azul/verde son compatibles con Aurora MySQL, Aurora PostgreSQL y la base de datos global de Aurora. Para conocer la disponibilidad de Amazon RDS, consulte Descripción general de las implementaciones azul/verde de Amazon RDS para Aurora en la Guía del usuario de Amazon RDS.

Temas

- Disponibilidad en regiones y versiones
- Ventajas de utilizar las implementaciones azul/verde de Amazon RDS
- Flujo de trabajo de una implementación azul/verde
- Autorización del acceso a las operaciones de la implementación azul/verde de Amazon Aurora

- Limitaciones y consideraciones para implementaciones azul/verde de Amazon Aurora
- Prácticas recomendadas para las implementaciones azul/verde de Amazon Aurora

Disponibilidad en regiones y versiones

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos y entre Regiones de AWS. Para obtener más información, consulte the section called "Implementaciones azul/verde".

Ventajas de utilizar las implementaciones azul/verde de Amazon RDS

Al utilizar las implementaciones azul/verde de Amazon RDS, puede mantenerse al día con los parches de seguridad, mejorar el rendimiento de las bases de datos y adoptar nuevas características de bases de datos con un tiempo de inactividad breve y predecible. Las implementaciones azules y verdes reducen los riesgos y el tiempo de inactividad de las actualizaciones de las bases de datos, como las actualizaciones principales o secundarias de las versiones del motor.

Las implementaciones azul/verde ofrecen los siguientes beneficios:

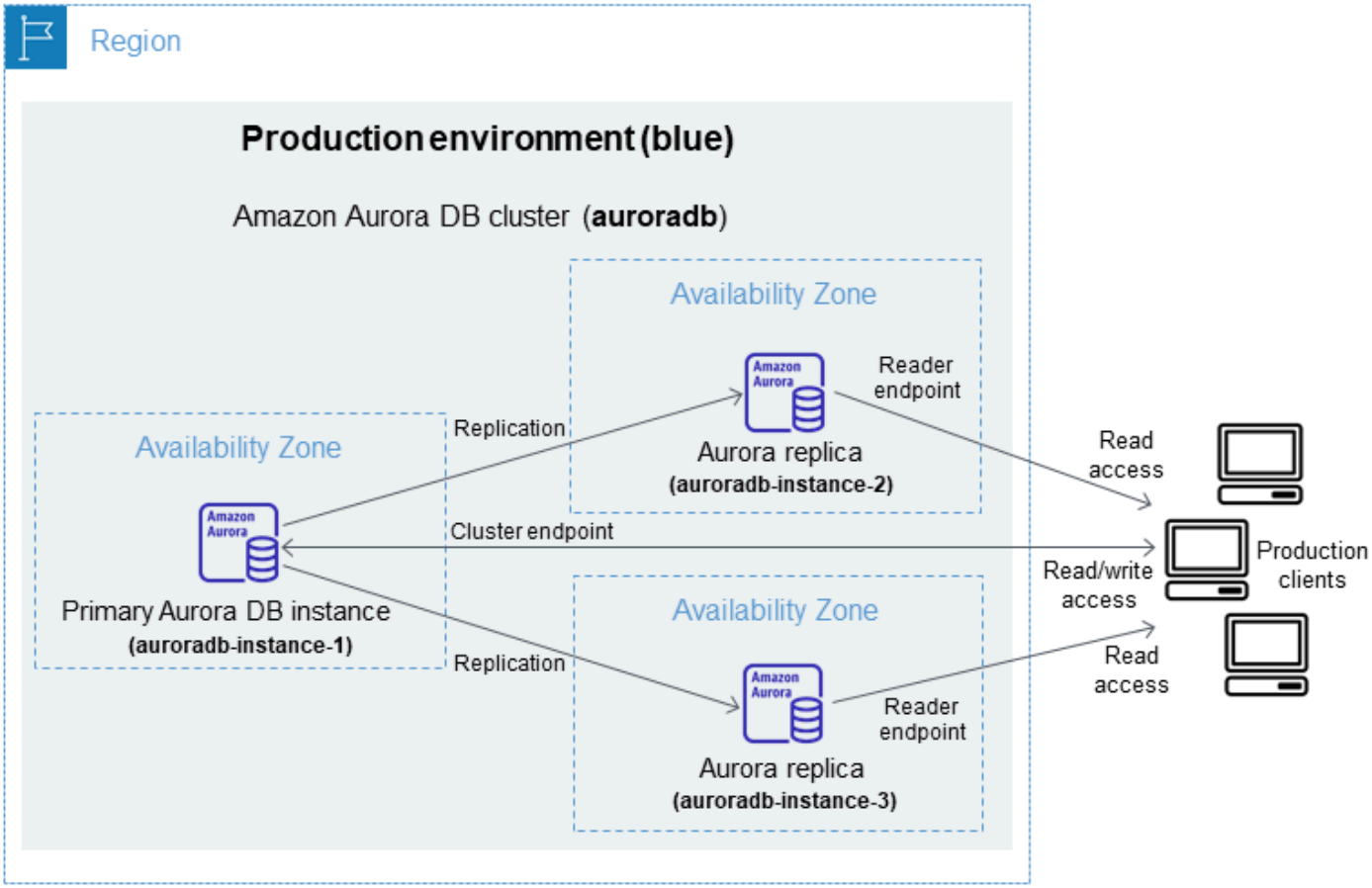
- Cree fácilmente un entorno de almacenamiento provisional listo para la producción.
- Replique automáticamente los cambios de la base de datos del entorno de producción al entorno de almacenamiento provisional.
- Pruebe los cambios en la base de datos en un entorno de almacenamiento provisional seguro sin que eso afecte al entorno de producción.
- Manténgase al día con los parches de las bases de datos y las actualizaciones del sistema.
- Implemente y pruebe las características más recientes de las bases de datos.
- Conmute su entorno de almacenamiento provisional para convertirlo en el nuevo entorno de producción sin cambios en la aplicación.
- Cambie de forma segura mediante el uso de barreras de protección de conmutaciones integradas.
- Elimine la pérdida de datos durante la conmutación.
- Conmutar rápidamente, normalmente en menos de un minuto, según su carga de trabajo.

Flujo de trabajo de una implementación azul/verde

Realice los siguientes pasos principales cuando utilice una implementación azul/verde para las actualizaciones del clúster de base de datos de Aurora.

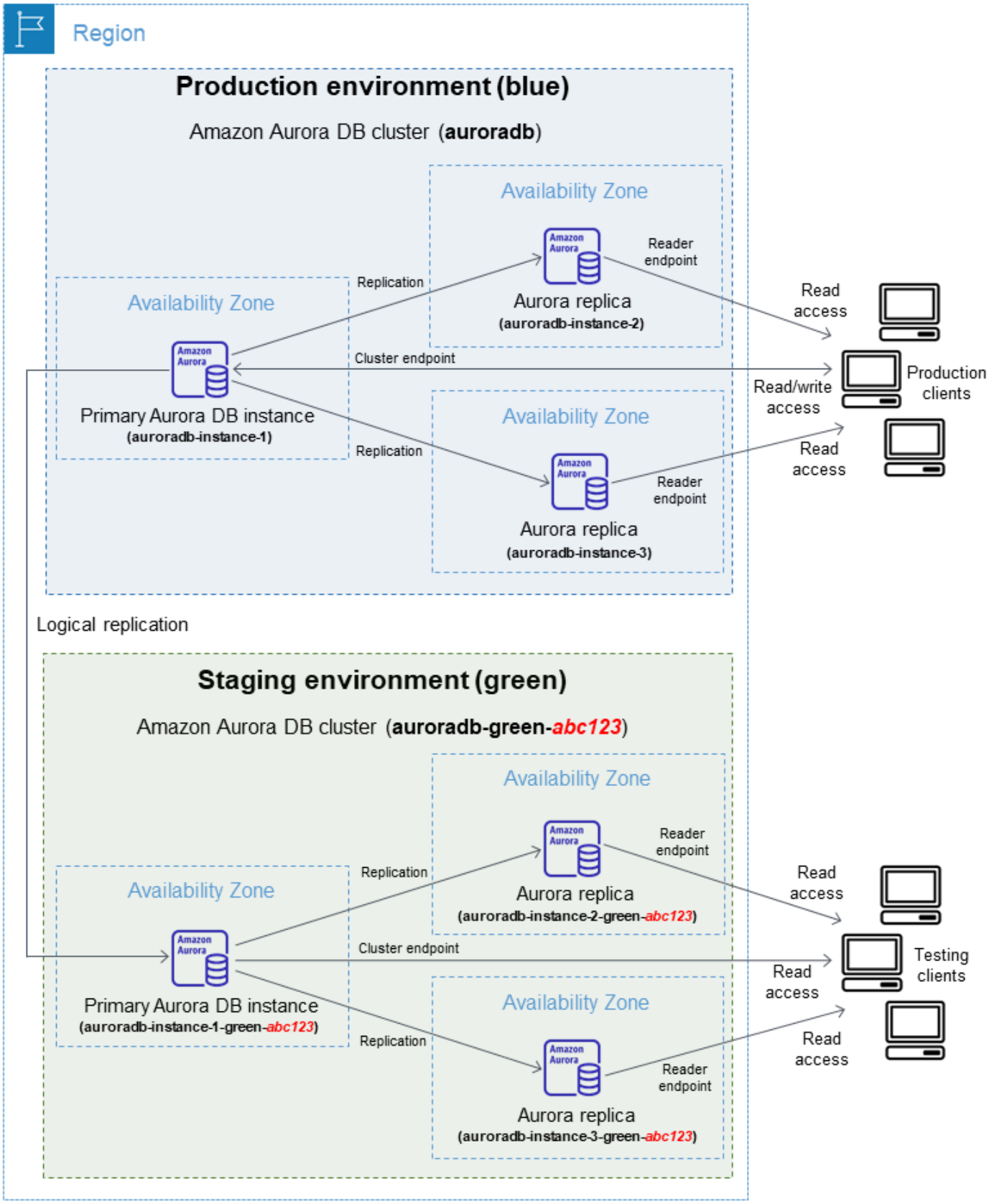
1. Identifique un clúster de base de datos de producción que requiera actualizaciones.

   En la imagen siguiente, se muestra un ejemplo de un clúster de base de datos de producción.



2. Cree la implementación azul/verde. Para obtener instrucciones, consulte Creación de una implementación azul/verde en Amazon Aurora.

   La siguiente imagen muestra un ejemplo de una implementación azul/verde del entorno de producción del paso 1. Al crear la implementación azul/verde, RDS copia la topología y la configuración completas del clúster de base de datos de Aurora para crear el entorno verde. Los nombres del clúster de base de datos copiado y de las instancias de base de datos se adjuntan con `-green-`*`random-characters`*. El entorno de almacenamiento provisional de la imagen contiene el clúster de base de datos (auroradb-green-*abc123*). También contiene las tres instancias de base de datos del clúster de base de datos (auroradb-instance1-green-*abc123*, auroradb-instance2-green-*abc123* y auroradb-instance3-green-*abc123*).


Region
Production environment (blue)
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2)
Replication
Availability Zone
Cluster endpoint
Read access
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Logical replication
Staging environment (green)
Amazon Aurora DB cluster (auroradb-green-abc123)
Availability Zone
Reader endpoint
Replication
Aurora replica
(auroradb-instance-2-green-abc123)
Availability Zone
Cluster endpoint
Read access
Testing clients
Read access
Primary Aurora DB instance
(auroradb-instance-1-green-abc123)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3-green-abc123)

Al crear la implementación azul/verde, puede actualizar la versión más alta del motor de base de datos y un grupo de parámetros de base de datos diferente para el clúster de base de datos del entorno verde. También puede especificar un grupo de parámetros de base de datos diferente para las instancias de base de datos del clúster de base de datos.

RDS también configura la replicación desde la instancia de base de datos principal en el entorno azul hasta la instancia de base de datos principal en el entorno verde.

> **Important**
>
> En la versión 3 de Aurora MySQL, tras crear la implementación azul/verde, el clúster de base de datos del entorno verde no permite las operaciones de escritura de forma predeterminada. Sin embargo, esto no se aplica a los usuarios que tienen el privilegio `CONNECTION_ADMIN`, incluido el usuario maestro de Aurora. Los usuarios con este privilegio pueden anular el comportamiento de `read_only`. Para obtener más información, consulte Modelo de privilegios basado en roles.

3. Realice cambios en el entorno de almacenamiento provisional.

   Por ejemplo, puede cambiar la clase de instancia de la base de datos que utilizan una o más instancias de base de datos en el entorno verde.

   Para obtener información acerca de la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

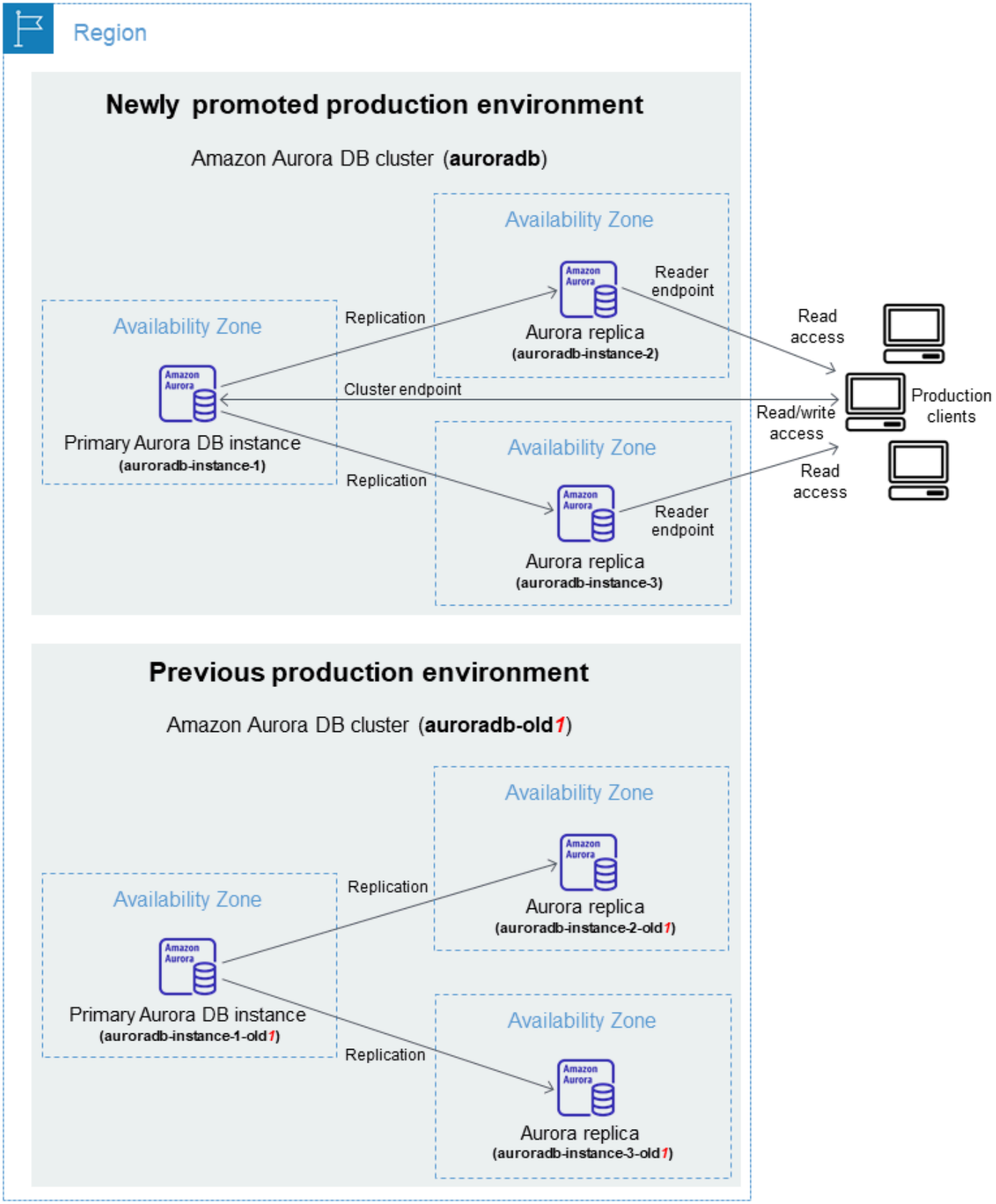
4. Ponga a prueba su entorno de almacenamiento temporal.

   Durante las pruebas, le recomendamos que mantenga como solo lectura las bases de datos de un entorno verde. Habilite las operaciones de escritura en el entorno verde con precaución, ya que pueden provocar conflictos de replicación. También pueden generar datos no deseados en las bases de datos de producción después de la conmutación. Para habilitar las operaciones de escritura en Aurora MySQL, ponga el parámetro `read_only` en `0` y reinicie la instancia de base de datos. En el caso de Aurora PostgreSQL, ponga el parámetro `default_transaction_read_only` en `off` en el nivel de sesión.

5. Cuando esté todo listo, realice una transición para hacer que el entorno de almacenamiento provisional sea el nuevo entorno de producción. Para obtener instrucciones, consulte Cambio de una implementación azul/verde en Amazon Aurora.

La conmutación provoca un tiempo de inactividad. El tiempo de inactividad suele ser inferior a un minuto, pero puede prolongarse en función de la carga de trabajo.

En la imagen siguiente, se muestran los clústeres de base de datos después de la conmutación.


Region
Newly promoted production environment
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Availability Zone
Replication
Aurora replica
(auroradb-instance-2)
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Previous production environment
Amazon Aurora DB cluster (auroradb-old1)
Availability Zone
Availability Zone
Replication
Aurora replica
(auroradb-instance-2-old1)
Primary Aurora DB instance
(auroradb-instance-1-old1)
Availability Zone
Replication
Aurora replica
(auroradb-instance-3-old1)

Tras la conmutación, el clúster de base de datos de Aurora en el entorno verde se convierte en el nuevo clúster de base de datos de producción. Los nombres y puntos de conexión del entorno de producción actual se asignan al entorno de producción al que le acaba de realizar la transición, por lo que no es necesario realizar cambios en la aplicación. Como resultado, el tráfico de producción ahora fluye al nuevo entorno de producción. El nombre del clúster de base de datos y de las instancias de base de datos del entorno azul se cambian de nombre añadiendo `-old`*n* al nombre actual, donde *n* es un número. Por ejemplo, suponga que el nombre de la instancia de base de datos en el entorno azul es `auroradb-instance-1`. Tras la conmutación, el nombre de la instancia de base de datos puede ser `auroradb-instance-1-old1`.

En el ejemplo de la imagen, se producen los siguientes cambios durante la conmutación:

- El clúster de base de datos de entorno verde `auroradb-green-abc123` pasa a ser el clúster de base de datos de producción denominado `auroradb`.
- La instancia de base de datos de entorno verde denominada `auroradb-instance1-green-abc123` se convierte en la instancia de base de datos de producción `auroradb-instance1`.
- La instancia de base de datos de entorno verde denominada `auroradb-instance2-green-abc123` se convierte en la instancia de base de datos de producción `auroradb-instance2`.
- La instancia de base de datos de entorno verde denominada `auroradb-instance3-green-abc123` se convierte en la instancia de base de datos de producción `auroradb-instance3`.
- El clúster de base de datos del entorno azul denominado `auroradb` se convierte en `auroradb-old1`.
- La instancia de base de datos del entorno azul denominada `auroradb-instance1` se convierte en `auroradb-instance1-old1`.
- La instancia de base de datos del entorno azul denominada `auroradb-instance2` se convierte en `auroradb-instance2-old1`.
- La instancia de base de datos del entorno azul denominada `auroradb-instance3` se convierte en `auroradb-instance3-old1`.

6. Si ya no necesita una implementación azul/verde, puede eliminarla. Para obtener instrucciones, consulte Eliminación de una implementación azul/verde en Amazon Aurora.

   Tras la conmutación, el entorno de producción anterior no se elimina, por lo que puede usarlo para realizar pruebas de regresión, si es necesario.

Autorización del acceso a las operaciones de la implementación azul/verde de Amazon Aurora

Los usuarios deben tener los permisos necesarios para realizar operaciones relacionadas con las implementaciones azul/verde. Puede crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. A continuación, puede asociar esas políticas a los roles o conjuntos de permisos de IAM que necesiten esos permisos. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.

El usuario que crea una implementación azul/verde debe tener permisos para realizar las siguientes operaciones de RDS:

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterEndpoint`

El usuario que cambia a una implementación azul/verde debe tener permisos para realizar las siguientes operaciones de RDS:

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBCluster`
- `rds:PromoteReadReplicaDBCluster`

El usuario que elimina una implementación azul/verde debe tener permisos para realizar las siguientes operaciones de RDS:

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBCluster`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterEndpoint`

Aurora aprovisiona y modifica los recursos en el entorno de almacenamiento provisional en su nombre. Estos recursos incluyen instancias de base de datos que utilizan una convención de nomenclatura definida internamente. Por lo tanto, las políticas de IAM asociadas no pueden contener patrones de nombres de recursos parciales, como `my-db-prefix-*`. Solo se admite el uso de comodines (*). En general, se recomienda utilizar etiquetas de recursos y otros atributos compatibles para controlar el acceso a estos recursos, en lugar de utilizar comodines. Para obtener más información, consulte Acciones, recursos y claves de condición de Amazon .

Permisos adicionales para las implementaciones azul/verde de base de datos de Aurora Global

Al crear implementaciones azul/verde para los clústeres de la base de datos de Aurora Global, además del permiso mencionado anteriormente, los usuarios necesitan los siguientes permisos para realizar operaciones de administración de la topología de clústeres global.

El usuario que crea una implementación azul/verde debe tener permisos para realizar las siguientes operaciones de RDS:

- `rds:CreateGlobalCluster`

El usuario que cambia a una implementación azul/verde debe tener permisos para realizar las siguientes operaciones de RDS:

- `rds:ModifyGlobalCluster`
- `rds:PromoteReadReplicaDBCluster`

El usuario que elimina una implementación azul/verde debe tener permisos para realizar las siguientes operaciones de RDS:

- `rds:DeleteGlobalCluster`

Limitaciones y consideraciones para implementaciones azul/verde de Amazon Aurora

Las implementaciones azul/verde en Amazon RDS requieren una consideración cuidadosa de factores como las ranuras de replicación, la administración de recursos, el tamaño de las instancias y los posibles impactos en el rendimiento de la base de datos. Las siguientes secciones proporcionan

orientación para ayudarle a optimizar su estrategia de implementación a fin de garantizar un tiempo de inactividad mínimo, transiciones fluidas y una administración eficaz del entorno de su base de datos.

Temas

- Limitaciones de las implementaciones azul/verde
- Limitaciones de base de datos global de Aurora para implementaciones azul/verde
- Consideraciones acerca de las implementaciones azul/verde

Limitaciones de las implementaciones azul/verde

Las siguientes limitaciones se aplican a las implementaciones azul/verde.

Temas

- Limitaciones generales de las implementaciones azul/verde
- Limitaciones de Aurora MySQL en implementaciones azul/verde
- Limitaciones de Aurora PostgreSQL para implementaciones azul/verde

Limitaciones generales de las implementaciones azul/verde

Las siguientes limitaciones generales se aplican a las implementaciones azul/verde:

- No puede detener e iniciar un clúster que forme parte de una implementación azul/verde.
- Las implementaciones azul/verde no admiten la administración de las contraseñas de los usuarios maestros con AWS Secrets Manager
- Si intenta forzar una búsqueda de datos anteriores en el clúster de base de datos azul, la implementación azul/verde se interrumpe y la transición se bloquea.
- Durante la conmutación, los entornos azul y verde no pueden tener integración sin ETL con Amazon Redshift. Debe eliminar la integración en primer lugar, realizar la conmutación y, a continuación, volver a crear la integración.
- El programador de eventos (parámetro `event_scheduler`) debe estar deshabilitado en el entorno verde al crear una implementación azul/verde. Esto evita que se generen eventos en el entorno verde que provoquen incoherencias.

- Las políticas de escalado automático en el clúster de base de datos azul no se copian en el entorno verde. Debe volver a configurarlas después de la transición, independientemente de si se configuraron inicialmente en el entorno azul o verde.
- No puede cambiar un clúster de base de datos sin cifrar por un clúster de base de datos con cifrado. Además, no puede cambiar un clúster de base de datos con cifrado por un clúster de base de datos sin cifrar.
- No puede cambiar un clúster de base de datos azul por una versión del motor superior a su clúster de base de datos verde correspondiente.
- Los recursos del entorno azul y el entorno verde deben estar en la misma Cuenta de AWS.
- Las implementaciones azul/verde no son compatibles con las siguientes características:
  - Amazon RDS Proxy
  - Réplicas de lectura entre regiones
  - Aurora Serverless v1Clústeres de base de datos de
  - CloudFormation

Limitaciones de Aurora MySQL en implementaciones azul/verde

Las siguientes limitaciones se aplican a las implementaciones azul/verde de Aurora MySQL:

- El clúster de base de datos de origen no puede contener ninguna base de datos con el nombre `tmp`. Las bases de datos con este nombre no se copiarán en el entorno verde.
- La instancia de base de datos azul no puede ser una réplica binlog externa.
- Si el clúster de base de datos de origen tiene habilitada la función de búsqueda de datos anteriores, el clúster de base de datos verde se crea sin soporte para la búsqueda de datos anteriores. Esto se debe a que la búsqueda de datos anteriores no funciona con la replicación de registros binarios (binlog), que es necesaria para las implementaciones azul/verde. Para obtener más información, consulte the section called “Búsqueda de datos anteriores de un clúster de base de datos”.
- Las implementaciones azul/verde no admiten el controlador JDBC de AWS para MySQL. Para obtener más información, consulte Known Limitations en GitHub.

Limitaciones de Aurora PostgreSQL para implementaciones azul/verde

Las siguientes limitaciones se aplican a las implementaciones azul/verde de Aurora PostgreSQL .

- Las tablas no registradas no se replican en el entorno verde a menos que el parámetro `rds.logically_replicate_unlogged_tables` esté establecido en 1 en el clúster de bases de datos azul. No modifique el valor de este parámetro después de crear una implementación azul/verde, a fin de evitar posibles errores de replicación en tablas no registradas.
- El clúster de base de datos azul no puede ser un origen lógico (publicador) ni una réplica (suscriptor).
- Si el clúster de bases de datos azul está configurado como el servidor externo de una extensión de contenedor de datos externo (FDW), debe usar el nombre del punto de conexión del clúster en lugar de las direcciones IP. Esto permite que la configuración siga funcionando después de la transición.
- En una implementación azul/verde, cada base de datos requiere una ranura de replicación lógica. A medida que aumenta el número de bases de datos, aumenta la sobrecarga de recursos, lo que puede provocar un retardo en la replicación, especialmente si el clúster de base de datos no se ha escalado lo suficiente. El impacto depende de factores como la carga de trabajo de la base de datos y el número de conexiones. Para mitigarlo, valore la posibilidad de escalar verticalmente la clase de instancia de base de datos o de reducir el número de bases de datos en el clúster de origen.
- Las implementaciones azul/verde son compatibles con Babelfish para Aurora PostgreSQL solo para la versión 15.7 y versiones posteriores a 15, y la versión 16.3 y versiones posteriores a 16.
- Si desea capturar los planes de ejecución en las réplicas de Aurora, debe proporcionar el punto de conexión del clúster de bases de datos al llamar a la función `apg_plan_mgmt.create_replica_plan_capture`. Esto garantiza que las capturas de planes sigan funcionando después de la transición. Para obtener más información, consulte the section called “Captura de planes de ejecución de Aurora PostgreSQL en réplicas”.
- El proceso de aplicación de replicación lógica en el entorno verde es de un solo subproceso. Si el entorno azul genera un gran volumen de tráfico de escritura, es posible que el entorno verde no pueda seguirle el ritmo. Esto puede provocar un retardo o un error en la replicación, sobre todo en el caso de las cargas de trabajo que producen un alto rendimiento continuo de escritura. Asegúrese de probar a fondo las cargas de trabajo. En los escenarios que requieran actualizaciones de versiones principales y administrar cargas de trabajo de escritura de gran volumen, considere enfoques alternativos como utilizar AWS Database Migration Service (AWS DMS) o la replicación lógica autoadministrada.
- La creación de nuevas particiones en tablas particionadas no es compatible durante las implementaciones azul/verde de Aurora. La creación de nuevas particiones implica operaciones de lenguaje de definición de datos (DDL)`CREATE TABLE`, que no se replican desde el entorno azul

al entorno verde. Sin embargo, las tablas particionadas existentes y sus datos se replicarán en el entorno verde.

- Las siguientes limitaciones se aplican a las extensiones de PostgreSQL:
  - La extensión `pg_partman` debe estar deshabilitada en el entorno azul al crear una implementación azul/verde. La extensión realiza operaciones de DDL, como `CREATE TABLE`, lo que rompe la replicación lógica del entorno azul al entorno verde.
  - La extensión `pg_cron` debe permanecer deshabilitada en todas las bases de datos verdes tras crear la implementación azul/verde. La extensión tiene programas de trabajo en segundo plano que se ejecutan como superusuarios y omiten la configuración de solo lectura del entorno verde, lo que podría provocar conflictos de replicación.
  - La extensión `apg_plan_mgmt` debe tener el parámetro `apg_plan_mgmt.capture_plan_baselines` establecido en `off` en todas las bases de datos verdes, a fin de evitar conflictos con las claves principales si se captura un plan idéntico en el entorno azul. Para obtener más información, consulte the section called “Descripción general de la administración de planes de consultas en Aurora PostgreSQL”.
  - Las extensiones `pglogical` y `pgactive` deben estar deshabilitadas en el entorno azul al crear una implementación azul/verde. Después de realizar una transición del entorno verde para que sea el nuevo entorno de producción, puede volver a habilitar las extensiones. Además, la base de datos azul no puede ser un suscriptor lógico de una instancia externa.
  - Si utiliza la extensión `pgAudit`, debe permanecer en las bibliotecas compartidas (`shared_preload_libraries`) de los grupos de parámetros de base de datos personalizados para las instancias de base de datos azules y verdes. Para obtener más información, consulte the section called “Configuración de la extensión pgAudit”.

Limitaciones específicas de la replicación lógica para las implementaciones azul/verde

PostgreSQL tiene ciertas restricciones relacionadas con la replicación lógica, que se traducen en limitaciones a la hora de crear implementaciones azul/verde para clústeres de bases de datos de Aurora PostgreSQL.

En la siguiente tabla, se describen las limitaciones de replicación lógica que se aplican a las implementaciones azul/verde para Aurora PostgreSQL. Para obtener más información, consulte Restricciones en la documentación de replicación lógica de PostgreSQL.

Limitación	Explicación
Las instrucciones del lenguaje de definición de datos (DDL), como `CREATE TABLE` y `CREATE SCHEMA`, no se replican desde el entorno azul al entorno verde.	Si Aurora detecta un cambio de DDL en el entorno azul, las bases de datos verdes pasan a un estado de Replicación degradada. Debe eliminar la implementación azul/verde y todas las bases de datos verdes y, a continuación, volver a crearla.
Las instrucciones del lenguaje de control de datos (DCL), como `GRANT` y `REVOKE`, no se replican desde el entorno azul al entorno verde.	Si Aurora detecta un intento de ejecutar una instrucción de DCL en el entorno azul, aparecerá un mensaje de advertencia. No hay ninguna configuración o API disponible para cambiar este comportamiento, ya que se trata de una limitación del proceso de implementación azul/verde.
Las operaciones `NEXTVAL` en los objetos de secuencia no están sincronizadas entre el entorno azul y el entorno verde.	Durante la transición, Aurora incrementa los valores de secuencia en el entorno verde para que coincidan con los del entorno azul. Si tiene miles de secuencias, esto puede retrasar la transición.
Los objetos grandes en el entorno azul no se replican en el entorno verde.	Si Aurora detecta la creación o modificación de objetos grandes en el entorno azul que están almacenados en la tabla de sistema `pg_largeobject` , las bases de datos verdes pasan a un estado de Replicación degradada. Debe eliminar la implementación azul/verde y todas las bases de datos verdes y, a continuación, volver a crearla.

Limitación	Explicación
Esto incluye los objetos grandes existentes como cualquier objeto grande recién creado o modificado o durante el proceso de implementación azul/verde.	
La actualización de las vistas materializadas interrumpe la replicación.	La actualización de las vistas materializadas en el entorno azul interrumpe la replicación en el entorno verde. Absténgase de actualizar las vistas materializadas en el entorno azul. Tras una transición, puede actualizarlos manualmente mediante el comando REFRESH MATERIALIZED VIEW o programar una actualización.
Las operaciones UPDATE y DELETE no están permitidas en las tablas que no tengan una clave principal.	Antes de crear una implementación azul/verde, asegúrese de que todas las tablas tengan una clave principal o usen `REPLICA IDENTITY FULL`. Sin embargo, use `REPLICA IDENTITY FULL` solo si no existe una clave principal o única, ya que afecta al rendimiento de la replicación. Para obtener más información, consulte la documentación de PostgreSQL.

Limitaciones de base de datos global de Aurora para implementaciones azul/verde

Además de las limitaciones generales y específicas del motor indicadas anteriormente, se aplican las siguientes limitaciones a las implementaciones azul/verde de la base de datos global de Aurora:

- Todas las operaciones se deben iniciar desde la misma región que el clúster de redacción de la base de datos global.

- Si se realiza una transición global o una conmutación por error global, la implementación azul/verde activa dejará de ser válida. La implementación azul/verde debe eliminarse y volver a crearse desde la nueva región principal.
- Para Aurora PostgreSQL, si tiene habilitado el reenvío de escritura global en su entorno de producción y crea una implementación azul/verde, el reenvío de escritura está desactivado en el clúster verde. Se habilita en el entorno verde solo después de la transición azul/verde, cuando el entorno verde se convierte en el nuevo entorno de producción. Tras la transición, el reenvío de escritura se desactiva en el clúster `-old1`.
- Si se modifica la topología de la base de datos global tras la creación de la implementación azul/verde, la implementación azul/verde activa dejará de ser válida. La implementación azul/verde debería eliminarse y volver a crearse desde la nueva región principal.
- Las instantáneas automatizadas se retienen según los días de retención de la copia de seguridad que se configuraron originalmente en el antiguo entorno azul. Las instantáneas automatizadas del antiguo clúster azul no se copian a verde.
- La conmutación por error global se admite durante una transición azul/verde, pero una transición global no se admite durante una transición azul/verde.
- Asegúrese de que el clúster de base de datos y los grupos de parámetros de base de datos para el entorno verde existan en todas las regiones secundarias con nombres idénticos. Si el grupo de parámetros de alguna región no está disponible, se utiliza el grupo de parámetros predeterminado de las regiones.
- Evite usar el RDS Proxy en algún miembro de la base de datos global durante la transición de implementación azul/verde.

Consideraciones acerca de las implementaciones azul/verde

Amazon RDS rastrea los recursos en las implementaciones azul/verde con el `DbiResourceId` y `DbClusterResourceId` de cada recurso. Este identificador de recurso es un identificador inmutable único de la Región de AWS para el recurso.

El ID del recurso es independiente del ID del clúster de base de datos. Todos aparecen en la configuración de la base de datos de la consola de RDS.

El nombre (ID de clúster) de un recurso cambia cuando conmuta una implementación azul/verde, pero cada recurso conserva el mismo identificador de recurso. Por ejemplo, un identificador de clúster de base de datos podría ser `mycluster` en el entorno azul. Tras la conmutación, es posible que el nombre del mismo clúster de base de datos se cambie a `mycluster-old1`. Sin embargo,

el identificador de recurso del clúster de base de datos no se cambia durante la conmutación. Por lo tanto, cuando se realiza una transición de los recursos verdes para que sean los nuevos recursos de producción, sus identificadores de recurso no coinciden con los identificadores de recurso azules que estaban en producción anteriormente.

Tras realizar una transición a una implementación azul/verde, valore la posibilidad de actualizar los ID de recursos a los de los recursos de producción que se acaban de promocionar para las características y los servicios integrados que utilizó con los recursos de producción. Específicamente, tenga en cuenta las siguientes actualizaciones:

- Si realiza el filtrado mediante la API de RDS y los identificadores de recursos, ajuste los identificadores de recursos utilizados en el filtrado después de la conmutación.
- Si utiliza CloudTrail para auditar recursos, ajuste los consumidores del CloudTrail para que realicen un seguimiento de los nuevos identificadores de recursos tras la conmutación. Para obtener más información, consulte Supervisión de llamadas a la API de Amazon Aurora en AWS CloudTrail.
- Si utiliza flujos de actividad de bases de datos para los recursos del entorno azul, ajuste la aplicación para supervisar los eventos de la base de datos para el nuevo flujo después de la conmutación. Para obtener más información, consulte Regiones y motores de base de datos Aurora admitidos para los flujos de actividad de bases de datos.
- Si utiliza la API de Información de rendimiento, ajuste los identificadores de los recursos en las llamadas a la API después de la conmutación. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

  Puede supervisar una base de datos con el mismo nombre después de la conmutación, pero no contiene los datos de antes de la conmutación.
- Si usa identificadores de recursos en las políticas de IAM, asegúrese de agregar los identificadores de los recursos a los que se les acaba de realizar una transición cuando sea necesario. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.
- Si tiene roles de IAM asociados al clúster de base de datos, asegúrese de volver a asociarlas tras la transición. Los roles asociados no se copian automáticamente en el entorno verde.
- Si se autentica en el clúster de base de datos con la autenticación de base de datos de IAM, asegúrese de que la política de IAM usada para acceder a la base de datos tenga las bases de datos azul y verde mostradas bajo el elemento `Resource` de la política. Esto es necesario para conectarse a la base de datos verde después del cambio. Para obtener más información, consulte the section called “Creación y uso de una política de IAM para el acceso a bases de datos de IAM”.

- Si desea restaurar una instantánea manual de un clúster de base de datos para un clúster de base de datos que formaba parte de una implementación azul/verde, asegúrese de restaurar la instantánea del clúster de base de datos correcta examinando la hora en que se tomó. Para obtener más información, consulte Restauración de una instantánea de clúster de base de datos.
- Tras la transición, las tareas de replicación de AWS Database Migration Service (AWS DMS) no se pueden reanudar porque el punto de control del entorno azul no es válido en el entorno verde. Debe volver a crear la tarea de DMS con un nuevo punto de control para continuar con la replicación.
- Amazon Aurora crea el entorno verde clonando el volumen de almacenamiento Aurora subyacente en el entorno azul. El volumen del clúster verde solo almacena los cambios incrementales que se realizan en el entorno verde. Si elimina el clúster de base de datos del entorno azul, el tamaño del volumen de almacenamiento de Aurora subyacente en el entorno verde aumenta hasta su tamaño completo. Para obtener más información, consulte the section called “Clonación de un volumen de clúster de base de datos de Aurora”.
- Al añadir una instancia de base de datos al clúster de base de datos en el entorno verde de una implementación azul/verde, la nueva instancia de base de datos no reemplazará a la instancia de base de datos del entorno azul cuando conmute. Sin embargo, la nueva instancia de base de datos se conserva en el clúster de base de datos y se convierte en una instancia de base de datos en el nuevo entorno de producción.
- Al eliminar una instancia de base de datos del clúster de base de datos en el entorno verde de una implementación azul/verde, no puede crear una nueva instancia de base de datos para reemplazarla en la implementación azul/verde.

  Si crea una nueva instancia de base de datos con el mismo nombre y ARN que la instancia de base de datos eliminada, tendrá una `DbiResourceId` diferente, por lo que no forma parte del entorno verde.

  Si se elimina una instancia de base de datos del clúster de base de datos en el entorno verde, se produce el siguiente comportamiento:

  - Si existe la instancia de base de datos del entorno azul con el mismo nombre, no se cambiará a la instancia de base de datos del entorno verde. El nombre de esta instancia de base de datos no se modificará añadiendo `-old`*n* al nombre de la instancia de base de datos.
  - Cualquier aplicación que apunte a la instancia de base de datos en el entorno azul seguirá utilizando la misma instancia de base de datos después de la conmutación.
- Si utiliza etiquetas de recursos para el control de acceso o la administración operativa, debe comprender que los cambios en las etiquetas no se sincronizan entre los entornos azules y verdes

hasta que se produce la transición. Al crear una implementación azul/verde, las etiquetas del entorno azul se copian en el entorno verde. Tras la creación, las modificaciones de etiquetas que realice en cualquiera de los entornos no se sincronizan automáticamente. Durante la transición, las etiquetas de entorno azules sustituyen a todas las etiquetas del entorno verde. Aplique todas las etiquetas necesarias al entorno azul antes de crear la implementación azul/verde o vuelva a aplicar las etiquetas necesarias al nuevo entorno de producción tras la transición. Para obtener más información acerca de las etiquetas, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.

Prácticas recomendadas para las implementaciones azul/verde de Amazon Aurora

Estos son los procedimientos recomendados para las implementaciones azul/verde.

Temas

- Prácticas recomendadas generales para las implementaciones azul/verde
- Prácticas recomendadas de Aurora MySQL para las implementaciones azul/verde
- Prácticas recomendadas de Aurora PostgreSQL para las implementaciones azul/verde
- Prácticas recomendadas de base de datos global de Aurora para las implementaciones azul/verde

Prácticas recomendadas generales para las implementaciones azul/verde

Tenga en cuenta las prácticas recomendadas generales al crear una implementación azul/verde.

- Pruebe minuciosamente el clúster de base de datos de Aurora en el entorno verde antes de realizar el cambio.
- Mantenga sus bases de datos del entorno verde en modo de solo lectura. Se recomienda habilitar las operaciones de escritura en el entorno verde con precaución, ya que pueden provocar conflictos de replicación. También pueden generar datos no deseados en las bases de datos de producción después de la conmutación.
- Si utiliza una implementación azul/verde para implementar cambios en el esquema, realice solo cambios compatibles con la replicación.

  Por ejemplo, puede añadir nuevas columnas al final de una tabla, sin interrumpir la replicación de la implementación azul a la implementación verde. Sin embargo, los cambios en el esquema, como

cambiar el nombre de las columnas o las tablas, interrumpen la replicación en la implementación verde.

Para obtener más información sobre los cambios compatibles con la replicación, consulte Replication with Differing Table Definitions on Source and Replica en la documentación de MySQL y Restrictions en la documentación de la replicación lógica de PostgreSQL.

- Utilice el punto de conexión del clúster, el punto de conexión del lector o el punto de conexión personalizado para todas las conexiones en ambos entornos. No utilice puntos de conexión de instancia ni puntos de conexión personalizados con listas estáticas o de exclusión.
- Al cambiar a una implementación azul/verde, siga las prácticas recomendadas de conmutación. Para obtener más información, consulte the section called “Prácticas recomendadas para realizar la conmutación”.

Prácticas recomendadas de Aurora MySQL para las implementaciones azul/verde

Tenga en cuenta las siguientes prácticas recomendadas a la hora de crear una implementación azul/verde a partir de un clúster de base de datos de Aurora MySQL.

- Si el entorno verde presenta un retraso en las réplicas, tenga en cuenta lo siguiente:
  - Deshabilite la retención de registros binarios si no es necesaria, o deshabilítela temporalmente hasta que la replicación se ponga al día. Para ello, vuelva a establecer el parámetro del clúster de base de datos `binlog_format` en `0` y reinicie la instancia de base de datos del escritor verde.
  - Establezca temporalmente el parámetro `innodb_flush_log_at_trx_commit` en `0` en el grupo de parámetros de base de datos verde. Una vez que la replicación se ponga al mismo nivel que los valores predeterminados de 1, vuelva a los valores predeterminados antes de la transición. Si se produce un cierre o bloqueo inesperado con el valor de los parámetros temporales, reconstruya el entorno verde para evitar que los datos se corrompan de forma no detectada. Para obtener más información, consulte the section called “Configuración de la frecuencia de vaciado del búfer de registro”.

Prácticas recomendadas de Aurora PostgreSQL para las implementaciones azul/verde

Tenga en cuenta las siguientes prácticas recomendadas a la hora de crear una implementación azul/verde a partir de un clúster de base de datos de Aurora PostgreSQL.

- Supervise la memoria caché de escritura de la replicación lógica de Aurora PostgreSQL y haga ajustes en el búfer de memoria caché si es necesario. Para obtener más información, consulte the section called “Supervisión de la memoria caché de escritura de la replicación lógica”.
- Aumente el valor del parámetro de base de datos `logical_decoding_work_mem` en el entorno azul. De este modo, se reduce la decodificación en el disco y, en su lugar, se utiliza memoria. Para obtener más información, consulte the section called “Ajuste de la memoria de trabajo para la descodificación lógica”.
  - Puede supervisar el desbordamiento de transacciones que se escribe en el disco mediante la métrica de CloudWatch `ReplicationSlotDiskUsage`. Esta métrica ofrece información sobre el uso del disco en las ranuras de replicación, lo que ayuda a identificar cuándo los datos de las transacciones superan la capacidad de la memoria y se almacenan en el disco. Puede supervisar la memoria liberable con la métrica `FreeableMemory` de CloudWatch. Para obtener más información, consulte the section called “Métricas de nivel de instancia para Amazon Aurora”.
  - En Aurora PostgreSQL versión 14 y posteriores, puede supervisar el tamaño de los archivos de desbordamiento lógico mediante la vista del sistema `pg_stat_replication_slots`.
- Actualice todas las extensiones de PostgreSQL a la versión más reciente antes de crear una implementación azul/verde. Para obtener más información, consulte the section called “Actualización de las extensiones de PostgreSQL”.
- Si utiliza la extensión `aws_s3`, otorgue acceso al clúster de bases de datos verde a Amazon S3 a través de un rol de IAM tras crear el entorno verde. Esto permite que los comandos de importación y exportación sigan funcionando después de la transición. Para obtener instrucciones, consulte the section called “Configuración del acceso a un bucket de Amazon S3”.
- Si especifica una versión de motor superior para el entorno verde, ejecute la operación `ANALYZE` en todas las bases de datos para actualizar la tabla de `pg_statistic`. Las estadísticas del optimizador no se transfieren durante una actualización de la versión principal, por lo que debe regenerar todas las estadísticas para evitar problemas de rendimiento. Para conocer prácticas recomendadas adicionales durante las actualizaciones de versiones principales, consulte the section called “Actualización a una versión principal”.
- Evite configurar desencadenadores como `ENABLE REPLICA` o `ENABLE ALWAYS` si el desencadenador se utiliza en el origen para manipular los datos. De lo contrario, el sistema de replicación propaga los cambios y ejecuta el desencadenador, lo que provoca la duplicación.
- Las transacciones de larga duración pueden provocar un retraso significativo en las réplicas. Para reducir el retraso en las réplicas, realice lo siguiente:

- Reduzca las subtransacciones y transacciones de larga duración que pueden retrasarse hasta que el entorno verde se ponga al mismo nivel que el entorno azul.
- Reduzca las operaciones masivas en el entorno azul hasta que el entorno verde se ponga al mismo nivel que el entorno azul.
- Inicie una operación de inmovilización de vacío manual en tablas ocupadas antes de crear la implementación azul/verde. Para obtener instrucciones, consulte Realización de una inmovilización de vacío manual.
- En la versión 12 y posteriores de PostgreSQL, deshabilite el parámetro `index_cleanup` en tablas grandes u ocupadas para mejorar la eficiencia del mantenimiento normal en las bases de datos azules. Para obtener más información, consulte Vaciado de una tabla lo más rápido posible.

  > **Note**
  >
  > Omitir regularmente la limpieza del índice durante la aspiración puede provocar una sobrecarga del índice, lo que podría degradar el rendimiento del escaneo. Como práctica recomendada, utilice este enfoque únicamente cuando utilice una implementación azul/verde. Una vez finalizada la implementación, se recomienda reanudar el mantenimiento y la limpieza periódicos de los índices.

- Se puede producir un retraso en la réplica si las instancias de base de datos azules y verdes tienen un tamaño insuficiente para la carga de trabajo. Asegúrese de que sus instancias de base de datos no estén alcanzando sus límites de recursos para el tipo de instancia. Para obtener más información, consulte the section called “Analice el uso de recursos con métricas de CloudWatch”.

- La replicación lenta puede provocar que los remitentes y los destinatarios se reinicien con frecuencia, lo que retrasa la sincronización. Para garantizar que permanezcan activos, deshabilite los tiempos de espera configurando el parámetro `wal_sender_timeout` en `0` en el entorno azul y el parámetro `wal_receiver_timeout` en `0` en el entorno verde.
- Revise el rendimiento de sus instrucciones UPDATE y DELETE y evalúe si la creación de un índice en la columna utilizada en la cláusula WHERE puede optimizar estas consultas. Esto puede mejorar el rendimiento cuando las operaciones se reproducen en un entorno verde. Para obtener más información, consulte the section called “Verificar los filtros de predicado para las consultas que generan esperas”.

- Si se utilizan desencadenadores, asegúrese de que no interfieran con la creación, actualización y eliminación de objetos `pg_catalog.pg_publication`, `pg_catalog.pg_subscription` y `pg_catalog.pg_replication_slots` objetos cuyos nombres comiencen por “rds”.
- Si Babelfish está activado en el clúster de base de datos de origen, los siguientes parámetros deben tener la misma configuración en el grupo de parámetros del clúster de base de datos de destino para el entorno verde que en el grupo de parámetros del clúster de base de datos de origen:
  - rds.babelfish_status
  - babelfishpg_tds.tds_default_numeric_precision
  - babelfishpg_tds.tds_default_numeric_scale
  - babelfishpg_tsql.default_locale
  - babelfishpg_tsql.migration_mode
  - babelfishpg_tsql.server_collation_name

  Para obtener más información sobre estos parámetros, consulte the section called “Configuración del grupo de parámetros del clúster de base de datos para Babelfish ”.

Prácticas recomendadas de base de datos global de Aurora para las implementaciones azul/verde

Además de las prácticas recomendadas generales y específicas del motor mostradas anteriormente, tenga en cuenta las siguientes prácticas recomendadas para la base de datos global de Aurora.

- Supervise las siguientes métricas de CloudWatch para identificar los períodos de baja actividad en el entorno de producción:
  - `DatabaseConnections`
  - `ActiveTransactions`

  Programe la transición azul/verde durante el periodo de mantenimiento planificado o durante un periodo de baja actividad.
- La duración de la transición azul/verde varía en función de la carga de trabajo y del número de regiones secundarias. Al iniciar una transición azul/verde, el servicio espera a que el retraso de la réplica llegue a cero antes de continuar. Se recomienda comprobar el retraso de la réplica antes de iniciar una transición.

- Si piensa utilizar un parámetro de base de datos o un grupo de parámetros de clúster de base de datos distinto del predeterminado para el entorno verde, cree el grupo de parámetros deseado con el mismo nombre en todas las regiones secundarias antes de iniciar la implementación azul/verde.

Creación de una implementación azul/verde en Amazon Aurora

RDS copia la topología y las características del entorno azul en un área de almacenamiento. Cuando la instancia de base de datos azul tiene réplicas de lectura, estas se copian como réplicas de la instancia verde. El almacenamiento asignado de todas las réplicas verdes coincide con la instancia principal verde, mientras que otros parámetros de almacenamiento se heredan de las réplicas azules.

Al crear una implementación azul/verde, se especifica el clúster de base de datos que se va a copiar en la implementación. El clúster de base de datos que elija es el clúster de base de datos de producción y se convierte en el clúster de base de datos en el entorno azul. RDS copia la topología del entorno azul en un área de almacenamiento provisional, junto con sus características configuradas. El clúster de base de datos se copia al entorno verde y RDS configura la replicación desde el clúster de base de datos en el entorno azul al clúster de base de datos en el entorno verde. RDS también copia todas las instancias de base de datos del clúster de base de datos en el clúster de base de datos.

Temas

- Preparación para una implementación azul/verde
- Especificación de cambios al crear una implementación azul/verde
- Creación de una implementación azul/verde
- Configuración para la creación de implementaciones azul/verde

Preparación para una implementación azul/verde

Hay ciertos pasos que debe seguir antes de crear una implementación azul/verde, en función del motor que ejecute su clúster de base de datos de Aurora.

Temas

- Preparación de un clúster de base de datos de Aurora MySQL para una implementación azul/verde
- Preparación de un clúster de base de datos de Aurora PostgreSQL para una implementación azul/verde

- Preparación de un clúster de base de datos (DB) global de Aurora para una implementación azul/verde

Preparación de un clúster de base de datos de Aurora MySQL para una implementación azul/verde

Antes de crear una implementación azul/verde para un clúster de base de datos de Aurora MySQL, el clúster debe asociarse a un grupo de parámetros de clúster de base de datos personalizado con el registro binario (`binlog_format`) activado. El registro binario es necesario para la replicación del entorno azul en el entorno verde. Aunque cualquier formato de binlog funciona, recomendamos `ROW` para reducir el riesgo de incoherencias en la replicación. Para obtener información sobre la creación de un grupo de parámetros de clúster de base de datos personalizado y la configuración de parámetros, consulte the section called “Grupos de parámetros de clúster de bases de datos”.

> **Note**
>
> Habilitar el registro binario aumenta el número de operaciones de E/S de escritura en disco en el clúster de base de datos. Puede supervisar el uso de IOPS con la métrica de CloudWatch `VolumeWriteIOPs`.

Tras habilitar el registro binario, reinicie el clúster de bases de datos para que los cambios entren en vigor. Las implementaciones azul/verde requieren que la instancia del escritor esté sincronizada con el grupo de parámetros del clúster de base de datos; de lo contrario, se producirá un error en la creación. Para obtener más información, consulte Reinicio de una instancia de base de datos dentro de un clúster de Aurora.

Además, se recomienda cambiar el periodo de retención de registros binarios a un valor que no sea `NULL`, con el fin de evitar que se depuren los archivos de registros binarios. Para obtener más información, consulte the section called “Establecimiento y muestra de la configuración del registro binario”.

Preparación de un clúster de base de datos de Aurora PostgreSQL para una implementación azul/verde

Antes de crear una implementación azul/verde para un clúster de bases de datos de Aurora PostgreSQL, haga lo siguiente.

- Asocie el clúster a un grupo de parámetros del clúster de bases de datos personalizado que tenga habilitada la replicación lógica (`rds.logical_replication`). La replicación lógica es necesaria para la replicación del entorno azul al entorno verde.

  Al habilitar la replicación lógica, también es necesario ajustar ciertos parámetros del clúster, como `max_replication_slots`, `max_logical_replication_workers` y `max_worker_processes`. Para obtener instrucciones sobre cómo habilitar la replicación lógica y ajustar estos parámetros, consulte the section called “Configuración de la replicación lógica”.

  Compruebe también que el parámetro `synchronous_commit` esté establecido en `on`.

  Tras configurar los parámetros necesarios, reinicie el clúster de bases de datos para que los cambios tengan efecto. Las implementaciones azul/verde requieren que la instancia del escritor esté sincronizada con el grupo de parámetros del clúster de base de datos; de lo contrario, se producirá un error en la creación. Para obtener más información, consulte Reinicio de una instancia de base de datos dentro de un clúster de Aurora.
- Confirme que el clúster de bases de datos ejecute una versión de Aurora PostgreSQL que sea compatible con las implementaciones azul/verde. Para obtener una lista de versiones compatibles, consulte the section called “Implementaciones azules/verdes con Aurora PostgreSQL”.
- Asegúrese de que todas las tablas del clúster de bases de datos tengan una clave principal. La replicación lógica de PostgreSQL no permite llevar a cabo operaciones UPDATE o DELETE en tablas que no tengan una clave principal.

Preparación de un clúster de base de datos (DB) global de Aurora para una implementación azul/verde

Antes de crear una implementación azul/verde para el clúster de base de datos (DB) global de Aurora, tenga en cuenta lo siguiente:

- Todas las operaciones se deben iniciar desde la misma región que el clúster de redacción de la base de datos global.
- Configuración del grupo de parámetros:
  - El entorno verde utiliza un grupo de parámetros nuevo que especifique o el mismo grupo de parámetros que el clúster azul (predeterminado).
  - Los grupos de parámetros personalizados se copian en el entorno verde.

- Si un grupo de parámetros especificado no existe en la región secundaria, el grupo de parámetros predeterminado de la región secundaria se utiliza para el entorno verde.

Especificación de cambios al crear una implementación azul/verde

Puede realizar los siguientes cambios en el clúster de base de datos en el entorno verde al crear la implementación azul/verde:

Puede realizar otras modificaciones en el clúster y sus instancias de base de datos en el entorno verde después de su implementación. Por ejemplo, puede especificar una versión superior del motor o un grupo de parámetros diferente.

Para obtener información acerca de la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Temas

- Especificación de una versión de motor superior
- Especificación de un grupo de parámetros de base de datos diferente

Especificación de una versión de motor superior

Puede especificar una versión superior del motor si desea probar una actualización del motor de base de datos. Tras la transición, la base de datos se actualiza a la versión principal o secundaria del motor de base de datos que especifique.

Especificación de un grupo de parámetros de base de datos diferente

Especifique un grupo de parámetros de clúster de base de datos que sea diferente del que utiliza el clúster de base de datos. Puede comprobar cómo afectan los cambios de parámetros al clúster de base de datos en el entorno verde o especificar un grupo de parámetros para una nueva versión principal del motor de base de datos en caso de una actualización.

Si especifica un grupo de parámetros de clúster de base de datos diferente, el grupo de parámetros especificado se asocia al clúster de base de datos en el entorno verde. Si no especifica un grupo de parámetros de clúster de base de datos diferente, el clúster de base de datos del entorno verde se asocia al mismo grupo de parámetros que el clúster de base de datos azul.

Creación de una implementación azul/verde

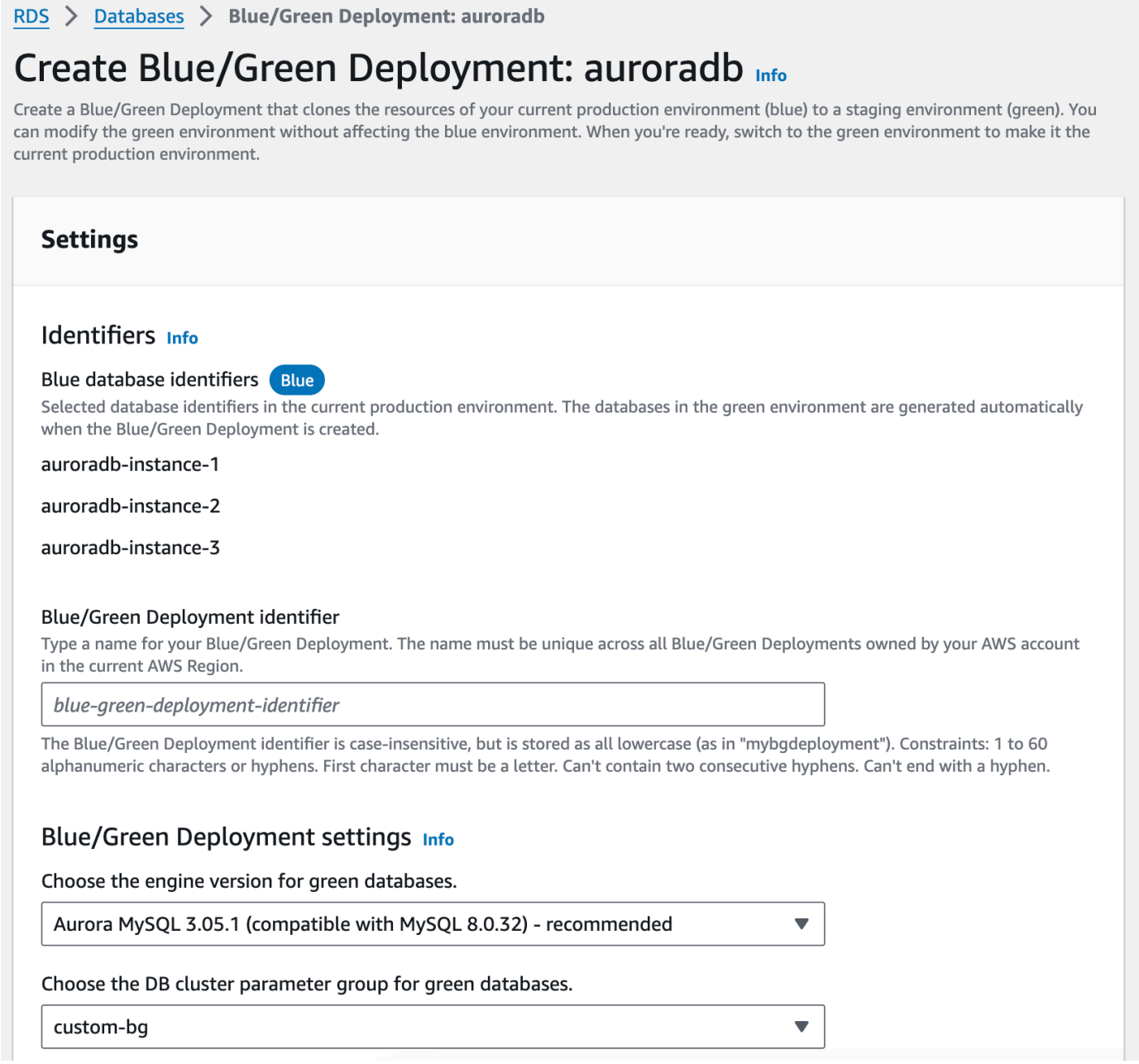
Puede crear una implementación azul/verde mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para crear una implementación azul/verde

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione el clúster de base de datos que desea copiar a un entorno verde.
3. Elija Acciones y Crear implementación azul/verde.

   Aparece la página Crear implementación azul/verde.



4. Revise los identificadores de la base de datos azul. Asegúrese de que coincidan con las instancias de base de datos esperadas en el entorno azul. Si no es así, seleccione Cancel (Cancelar).
5. En Nombre de implementación azul/verde, ingrese un nombre para la implementación azul/verde.
6. En el resto de secciones, especifique los ajustes de configuración del entorno verde. Para obtener más información acerca de cada configuración, consulte the section called “Opciones disponibles”.

Puede realizar otras modificaciones en las bases de datos en el entorno verde después de su implementación.

7. Seleccione Crear.

AWS CLI

Para crear una implementación azul/verde mediante la AWS CLI, utilice el comando create-blue-green-deployment. Para obtener información sobre todas las opciones disponibles, consulte the section called “Opciones disponibles”.

Example

Para Linux, macOS o Unix:

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name aurora-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb \
    --target-engine-version 8.0 \
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

Para Windows:

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name aurora-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb ^
    --target-engine-version 8.0 ^
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

API de RDS

Para crear una implementación azul/verde mediante la API de Amazon RDS, utilice la operación `CreateBlueGreenDeployment`. Para obtener más información acerca de cada opción, consulte the section called “Opciones disponibles”.

Configuración para la creación de implementaciones azul/verde

En la siguiente tabla se explican los ajustes que puede elegir al crear una implementación azul/verde. Para obtener más información sobre las opciones de la AWS CLI, consulte create-blue-

green-deployment. Para obtener más información sobre los parámetros de la API de RDS, consulte CreateBlueGreenDeployment.

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS

Identificador de implementación azul/verde	Un nombre de la implementación azul/verde.	Opción de la CLI:
		`--blue-green-deployment-name`
		Parámetro de la API:
		`BlueGreenDeploymentName`

Identificador de base de datos azul	El identificador de el clúster que desea copiar al entorno verde. Cuando utilice la CLI o la API, especifique el clúster del nombre de recurso de Amazon (ARN).	Opción de la CLI:
		`--source`
		Parámetro de la API:
		`Source`

Grupo de parámetros del clúster de base de datos para bases de datos verdes	Un grupo de parámetros para asociarlo a las bases de datos en el entorno verde.	Opción de la CLI:
		`--target-db-cluster-parameter-group-name`
		Parámetro de la API:
		`TargetDBClusterParameterGroupName`

Versión de motor para bases de datos verdes	Actualice los clústeres del entorno verde a la versión del motor de base de datos especificada.
	Si elige un clúster de bases de datos de Aurora PostgreSQL,
		Opción de la CLI:
		`--target-engine-version`
		Parámetro de la API de RDS:
		`TargetEngineVersion`

Configuración de la consola	Descripción de la configuración	Opción de la CLI y parámetro de la API de RDS
	revise y acepte las limitaciones de la replicación lógica. Para obtener más información, consulte the section called “Limitaciones específicas de la replicación lógica para las implementaciones azul/verde”.	

Visualización de una implementación azul/verde en Amazon Aurora

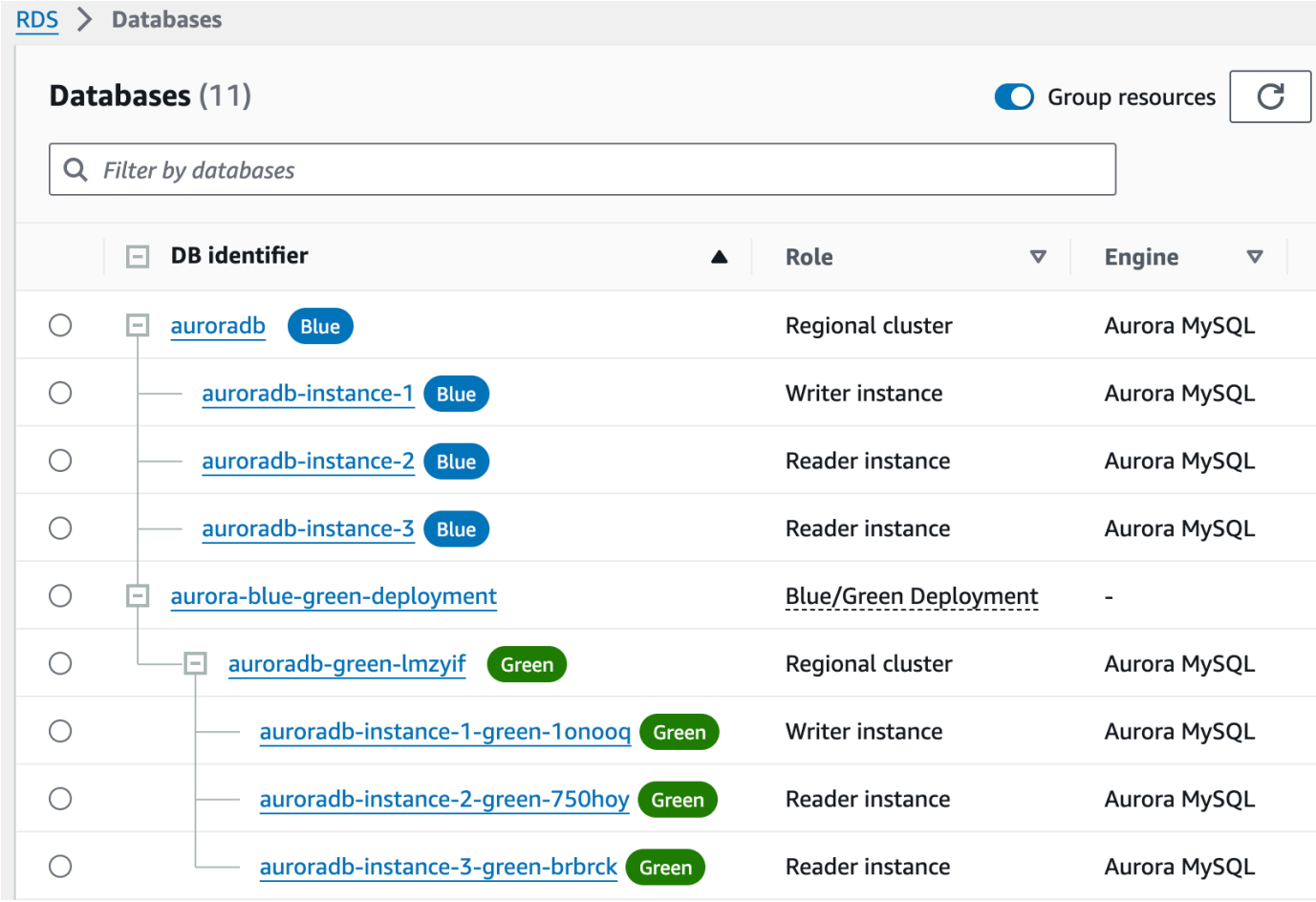
Puede ver los detalles de una implementación azul/verde mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

También puede ver los eventos y suscribirse a ellos para obtener información sobre una implementación azul/verde. Para obtener más información, consulte Eventos de implementación azul/verde.

Consola

Para ver los detalles de una implementación azul/verde

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos) y, a continuación, busque la implementación azul/verde en la lista.

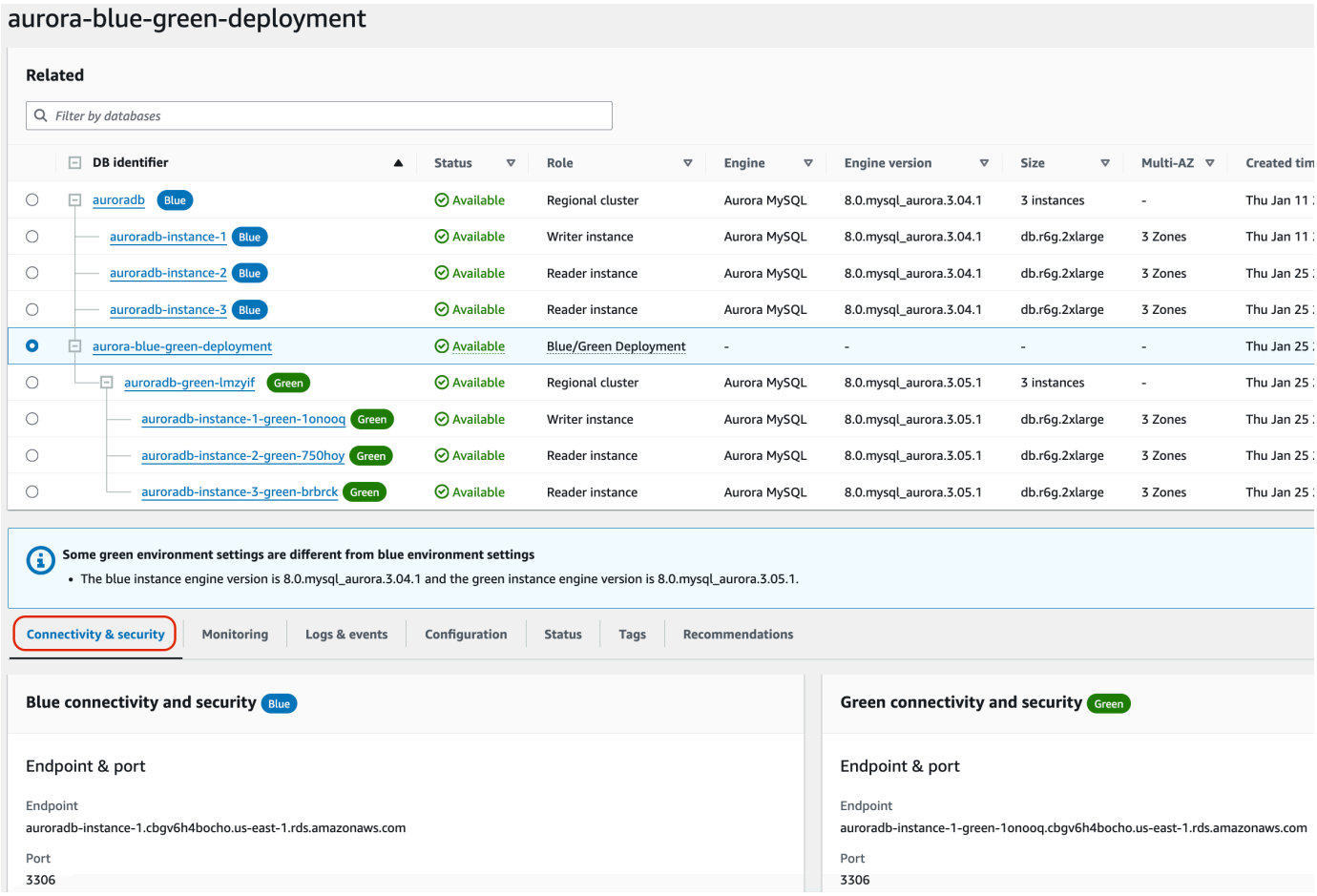


El valor de Role (Rol) para la implementación azul/verde es Blue/Green Deployment (Implementación azul/verde).

3. Elija el nombre de la implementación azul/verde que desee ver para mostrar sus detalles.

   Cada pestaña tiene una sección para la implementación azul y una sección para la implementación verde. Por ejemplo, en la pestaña Configuración, la versión del motor de base de datos puede ser diferente en el entorno azul y en el entorno verde si está actualizando la versión del motor de base de datos en el entorno verde.

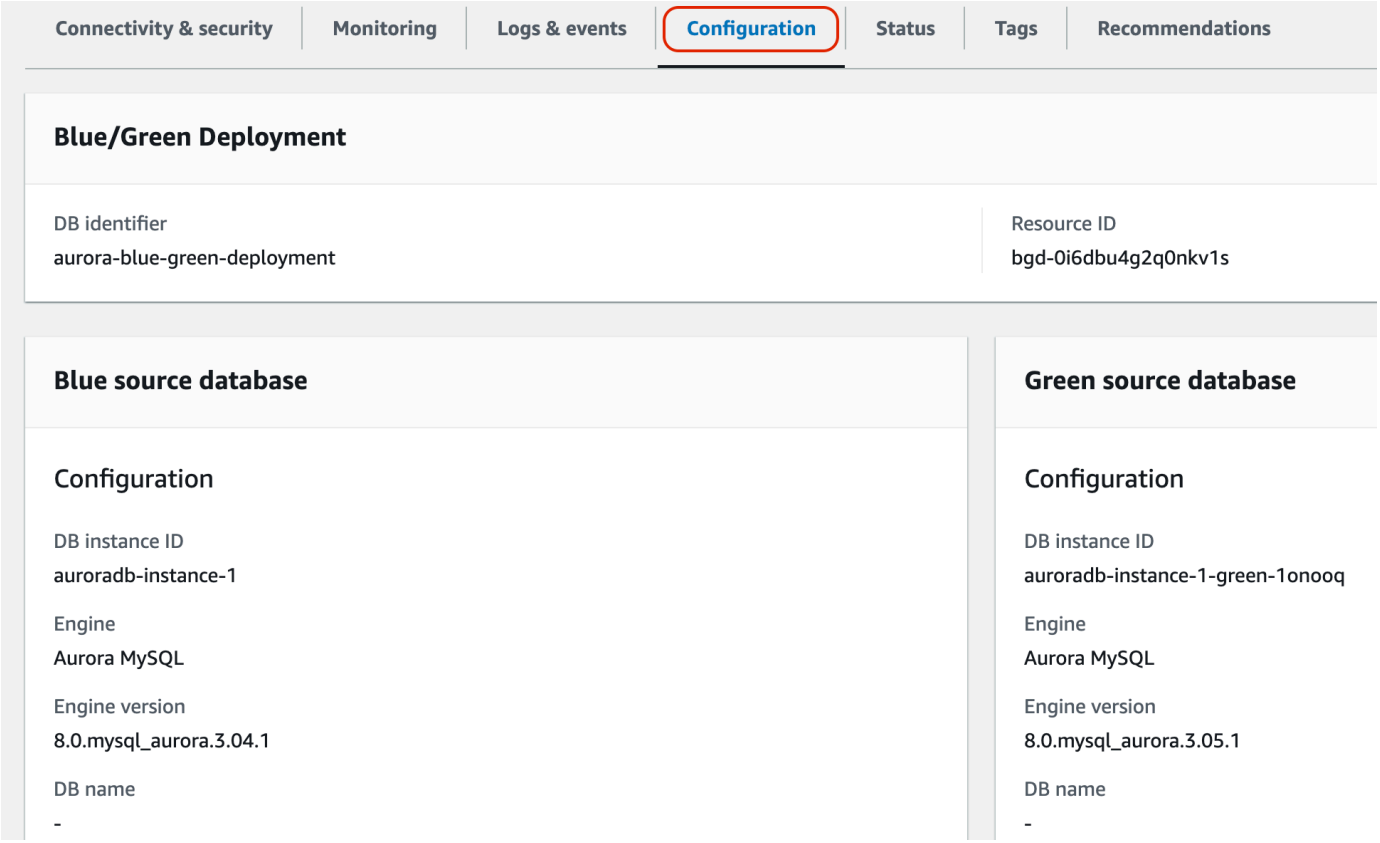
   En la siguiente imagen se muestra un ejemplo de la pestaña Conectividad y seguridad.



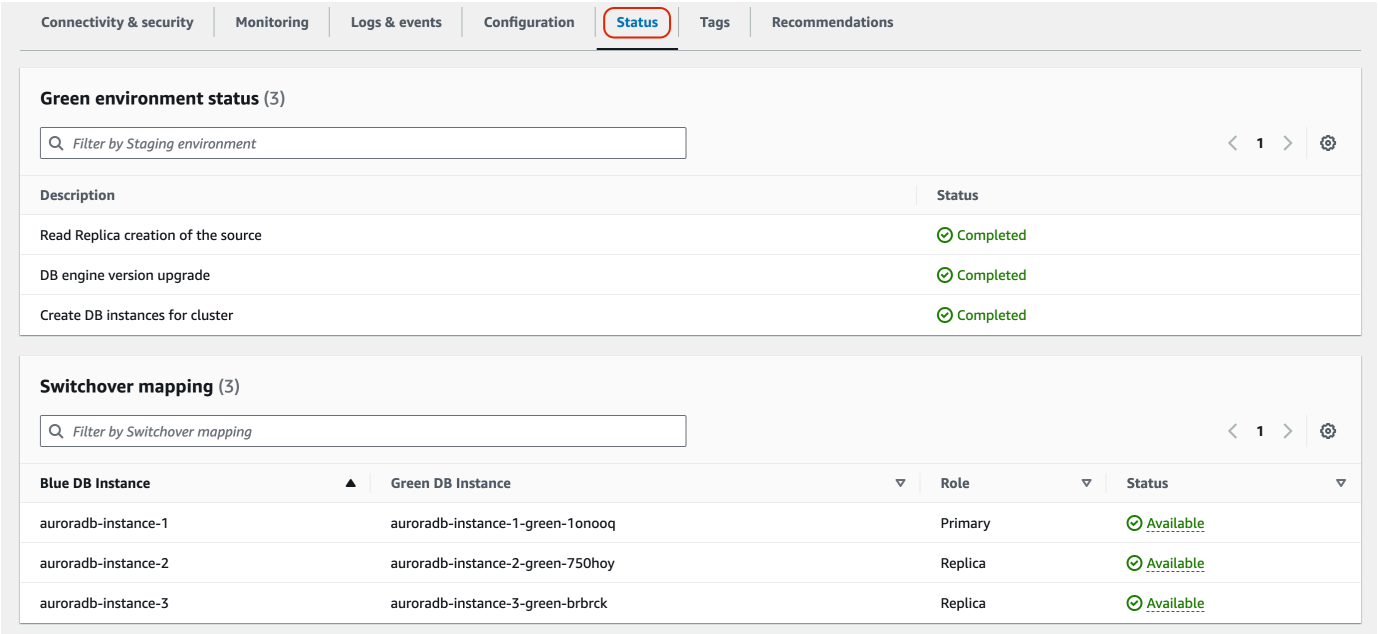
La pestaña Conectividad y seguridad también incluye una sección llamada Replicación, que muestra el estado actual de la replicación y el retraso de réplica entre los entornos azul y verde. Si el estado de replicación es `Replicating`, la implementación azul/verde se está replicando correctamente.

En el caso de las implementaciones azul/verde de Aurora PostgreSQL , el estado de la replicación puede cambiar a `Replication degraded` si realiza cambios en DDL o en objetos grandes no compatibles en el entorno azul. Para obtener más información, consulte the section called “Limitaciones específicas de la replicación lógica para las implementaciones azul/verde”.

En la siguiente imagen se muestra un ejemplo de la pestaña Configuración.



En la siguiente imagen se muestra un ejemplo de la pestaña Estado.



AWS CLI

Para ver los detalles de una implementación azul/verde mediante la AWS CLI, utilice el comando describe-blue-green-deployments.

Example Vea los detalles de una implementación azul/verde filtrando por su nombre

Con el comando describe-blue-green-deployments, puede filtrar por `--blue-green-deployment-name`.

En el siguiente ejemplo, se muestran los detalles de una implementación azul/verde denominada *`my-blue-green-deployment`*.

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example Para ver los detalles de una implementación azul/verde, especifique su identificador

Cuando use el comando describe-blue-green-deployments, puede especificar la opción `--blue-green-deployment-identifier`.

En el siguiente ejemplo, se muestran los detalles de una implementación azul/verde con el identificador *`bgd-1234567890abcdef`*.

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

API de RDS

Para ver los detalles de una implementación azul/verde mediante la API de Amazon RDS, utilice la opración `DescribeBlueGreenDeployments` y especifique el `BlueGreenDeploymentIdentifier`.

Cambio de una implementación azul/verde en Amazon Aurora

Una transición hace que el clúster de base de datos, incluidas sus instancias de base de datos, en el entorno verde se convierta en el clúster de base de datos de producción. Antes de conmutar, el tráfico de producción se dirige al clúster en el entorno azul. Tras conmutar, el tráfico de producción se dirige al clúster de base de datos en el entorno verde.

Cambiar una implementación azul/verde no es lo mismo que promocionar el clúster de base de datos verde dentro de la implementación azul/verde. Si promociona manualmente el clúster de base de datos seleccionando Promocionar en el menú Acciones, la replicación entre los entornos azul y verde se romperá y la implementación azul/verde entrará en el estado La configuración no es válida.

Temas

- Tiempo de espera de la conmutación
- Barreras de protección de la conmutación
- Acciones de conmutación
- Prácticas recomendadas para realizar la conmutación
- Verificación de las métricas de CloudWatch antes de la conmutación
- Monitoreo del retardo de réplica antes de la transición
- Conmutación de una implementación azul/verde
- Después de la conmutación

Tiempo de espera de la conmutación

Puede especificar un tiempo de espera para la conmutación de entre 30 y 3600 segundos (una hora). Si la conmutación dura más de lo especificado, los cambios se revierten y no se realiza ningún cambio en ninguno de los entornos. El valor predeterminado del tiempo de espera es de 300 segundos (cinco minutos).

Barreras de protección de la conmutación

Al iniciar una conmutación, Amazon RDS realiza algunas comprobaciones básicas para comprobar si los entornos azul y verde están preparados para ella. Estas comprobaciones se conocen como barreras de protección de la conmutación. Estas barreras de protección de la conmutación evitan que este se realice si los entornos no están preparados para ello. Por lo tanto, evitan que haya tiempos de inactividad más prolongados de lo esperado y evitan la pérdida de datos entre los entornos azul y verde que podría producirse si se iniciara la conmutación.

Amazon RDS ejecuta las siguientes comprobaciones de barreras de protección en el entorno verde:

- Estado de la replicación: comprueba si el estado de replicación del clúster de bases de datos es correcto. El clúster de base de datos verde es una réplica del clúster de base de datos azul.

- Retraso de replicación: comprueba si el retraso del clúster de bases de datos verde está dentro de los límites permisibles para la transición. Los límites permitidos se basan en el tiempo de espera especificado. El retardo de la réplica indica qué retardo tiene el clúster de base de datos verde con respecto al clúster de base de datos azul. Para obtener más información, consulte the section called “Monitoreo del retardo de réplica antes de la transición”.
- Escrituras activas: asegúrese de que no haya escrituras activas en el clúster de bases de datos verde.

Amazon RDS ejecuta las siguientes comprobaciones de barreras de protección en el entorno azul:

- Replicación externa: en el caso de Aurora PostgreSQL, se asegura de que el entorno azul no sea un origen lógico autoadministrado (publicador) o una réplica (suscriptor). Si es así, le recomendamos que elimine las ranuras de replicación autoadministradas y las suscripciones en todas las bases de datos del entorno azul, proceda con la transición y, a continuación, vuelva a crearlos para reanudar la replicación. En el caso de Aurora MySQL, comprueba si la base de datos azul no es una réplica binlog externa. Si es así, asegúrese de que no se esté replicando activamente.
- Escrituras activas de ejecución prolongada: asegúrese de que no haya escrituras activas de ejecución prolongada en el clúster de bases de datos azul, ya que pueden aumentar el retardo de la réplica.
- Instrucciones DDL de ejecución prolongada: asegúrese de que no haya instrucciones DDL de ejecución prolongada en el clúster de bases de datos azul, ya que pueden aumentar el retardo de la réplica.
- Cambios en PostgreSQL no compatibles: en el caso de las implementaciones azul/verde que utilizan la replicación lógica de Aurora PostgreSQL, se asegura de que no haya habido cambios de DDL ni adiciones o modificaciones de objetos grandes en el entorno azul. Para obtener más información, consulte the section called “Limitaciones específicas de la replicación lógica para las implementaciones azul/verde”.

  Si Amazon RDS detecta cambios no compatibles en PostgreSQL, cambia el estado de la replicación a `Replication degraded` y le notifica de que la transición no está disponible para la implementación azul/verde. Para continuar con la transición, le recomendamos que elimine y vuelva a crear la implementación azul/verde y todas las bases de datos verdes. Para ello, seleccione Acciones, Eliminar con bases de datos verdes.

Acciones de conmutación

Al conmutar una implementación azul/verde, RDS realiza las siguientes acciones:

1. Realiza comprobaciones de barreras de protección para verificar si los entornos azul y verde están listos para la conmutación.
2. Detiene las nuevas operaciones de escritura en el clúster de base de datos en ambos entornos.
3. Elimina las conexiones a las instancias de base de datos en ambos entornos y no permite nuevas conexiones.
4. Espera a que la replicación alcance el entorno verde para que el entorno verde esté sincronizado con el entorno azul.
5. Cambia el nombre del clúster y las instancias de base de datos en ambos entornos.

   RDS cambia el nombre del clúster y las instancias de base de datos del entorno verde para que coincidan con el clúster y las instancias de base de datos correspondientes del entorno azul. Por ejemplo, suponga que el nombre de una instancia de base de datos en el entorno azul es `mydb`. Suponga también que el nombre de la instancia de base de datos correspondiente en el entorno verde es `mydb-green-abc123`. Durante la conmutación, el nombre de la instancia de base de datos del entorno verde se cambia a `mydb`.

   RDS cambia el nombre del clúster y las instancias de base de datos en el entorno azul añadiendo `-old`*n* al nombre actual, donde *n* es un número. Por ejemplo, suponga que el nombre de una instancia de base de datos en el entorno azul es `mydb`. Tras la conmutación, el nombre de la instancia de base de datos puede ser `mydb-old1`.

   RDS también cambia el nombre de los puntos de conexión del entorno verde para que coincidan con los puntos de conexión correspondientes del entorno azul, de modo que no sea necesario realizar cambios en la aplicación.
6. Permite conexiones a bases de datos en ambos entornos.
7. Permite operaciones de escritura en el clúster de base de datos del nuevo entorno de producción.

   Tras la transición, el clúster de base de datos de producción anterior solo permite operaciones de lectura. Incluso si activa la escritura en el clúster de base de datos, permanece como de solo lectura hasta que elimine la implementación azul/verde.

Puede supervisar el estado de una conmutación mediante Amazon EventBridge. Para obtener más información, consulte the section called “Eventos de implementación azul/verde”.

Durante la transición, las etiquetas de entorno azules sustituyen a todas las etiquetas en los recursos del entorno verde. Las etiquetas que haya agregado directamente a los recursos del entorno verde se sobrescriben durante este proceso. Para obtener más información acerca de las etiquetas, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.

Si se inicia la conmutación y, a continuación, se detiene antes de finalizar por cualquier motivo, los cambios se revierten y no se realiza ningún cambio en ninguno de los entornos.

Prácticas recomendadas para realizar la conmutación

Antes de la transición, le recomendamos encarecidamente que siga los procedimientos recomendadas y complete las siguientes tareas:

- Pruebe minuciosamente los recursos en el entorno verde. Asegúrese de que funcionan de manera adecuada y eficiente.
- Supervise las métricas relevantes de Amazon CloudWatch. Para obtener más información, consulte the section called “Verificación de las métricas de CloudWatch antes de la conmutación”.
- Identifique el mejor momento para realizar la conmutación.

  Durante la conmutación, las escrituras se interrumpen en las bases de datos de ambos entornos. Identifique un momento en el que el tráfico sea menor en su entorno de producción. Las transacciones de larga duración, como las DDL activas, pueden aumentar el tiempo de la conmutación, lo que se traduce en un tiempo de inactividad más prolongado para las cargas de trabajo de producción.

  Si hay una gran cantidad de conexiones en el clúster de base de datos y las instancias de base de datos, considere la posibilidad de reducirlas manualmente hasta la cantidad mínima necesaria para su aplicación antes de cambiar a la implementación azul/verde. Una forma de lograrlo consiste en crear un script que supervise el estado de la implementación azul/verde y comience a limpiar las conexiones cuando detecte que el estado ha cambiado a `SWITCHOVER_IN_PROGRESS`.
- Asegúrese de que el clúster y las instancias de base de datos de ambos entornos se encuentren en el estado `Available`.
- Asegúrese de que el clúster de base de datos del entorno verde se encuentre en buen estado y se esté replicando.
- Asegúrese de que las configuraciones de red y cliente no aumenten el tiempo de vida (TTL) de la caché de DNS más de cinco segundos, que es el valor predeterminado para las zonas DNS de Aurora.

De lo contrario, las aplicaciones seguirán enviando tráfico de escritura al entorno azul después del cambio.

- Para las implementaciones azul/verde de Aurora PostgreSQL, haga lo siguiente:
  - Revise las limitaciones de la replicación lógica y tome las medidas necesarias antes de la transición. Para obtener más información, consulte the section called “Limitaciones específicas de la replicación lógica para las implementaciones azul/verde”.
  - Ejecute la operación `ANALYZE` para actualizar la tabla `pg_statistics`. Esto reduce el riesgo de problemas de rendimiento tras la transición.

> **Note**
>
> Durante una conmutación, no puede modificar ningún clúster de base de datos incluido en la conmutación.

Verificación de las métricas de CloudWatch antes de la conmutación

Antes de conmutar una implementación azul/verde, le recomendamos que compruebe el valor s de las siguientes métricas en Amazon CloudWatch.

- `DatabaseConnections`: utilice esta métrica para calcular el nivel de actividad de la implementación azul/verde y asegúrese de que el valor esté en un nivel aceptable para su implementación antes de realizar la conmutación. Si Información sobre rendimiento está activado, `DBLoad` es una métrica más precisa.
- `ActiveTransactions`: si `innodb_monitor_enable` está configurado en `all` en el grupo de parámetros de base de datos para cualquiera de sus instancias de base de datos, utilice esta métrica para comprobar si hay un número elevado de transacciones activas que puedan impedir la conmutación.

Para obtener más información, consulte the section called “Métricas de CloudWatch para Amazon Aurora”.

Monitoreo del retardo de réplica antes de la transición

Antes de conmutar una implementación azul/verde, asegúrese de que el retraso de réplica esté próximo a cero para reducir el tiempo de inactividad.

Aurora MySQL

Para las implementaciones azul/verde de MySQL , compruebe la métrica `AuroraBinlogReplicaLag` de CloudWatch en el entorno verde para identificar el retraso de réplica actual. Para obtener más información, consulte the section called “Diagnóstico y resolución de retardos entre réplicas de lectura”.

Aurora PostgreSQL

Para las implementaciones azul/verde de PostgreSQL , compruebe la métrica `OldestReplicationSlotLag` de CloudWatch en el entorno verde para identificar el retraso de réplica actual. Para obtener más información, consulte the section called “Métricas de nivel de instancia para Amazon Aurora”.

Además, puede ejecutar la siguiente consulta SQL en el entorno azul:

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
WHERE slot_type = 'logical';

slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

`confirmed_flush_lsn` representa el número de secuencia de registro (LSN) más reciente que se envió a la réplica. `pg_current_wal_lsn` representa la ubicación actual de la base de datos. Un valor 0 en `lsn_distance` significa que la réplica está funcionando al mismo ritmo.

Conmutación de una implementación azul/verde

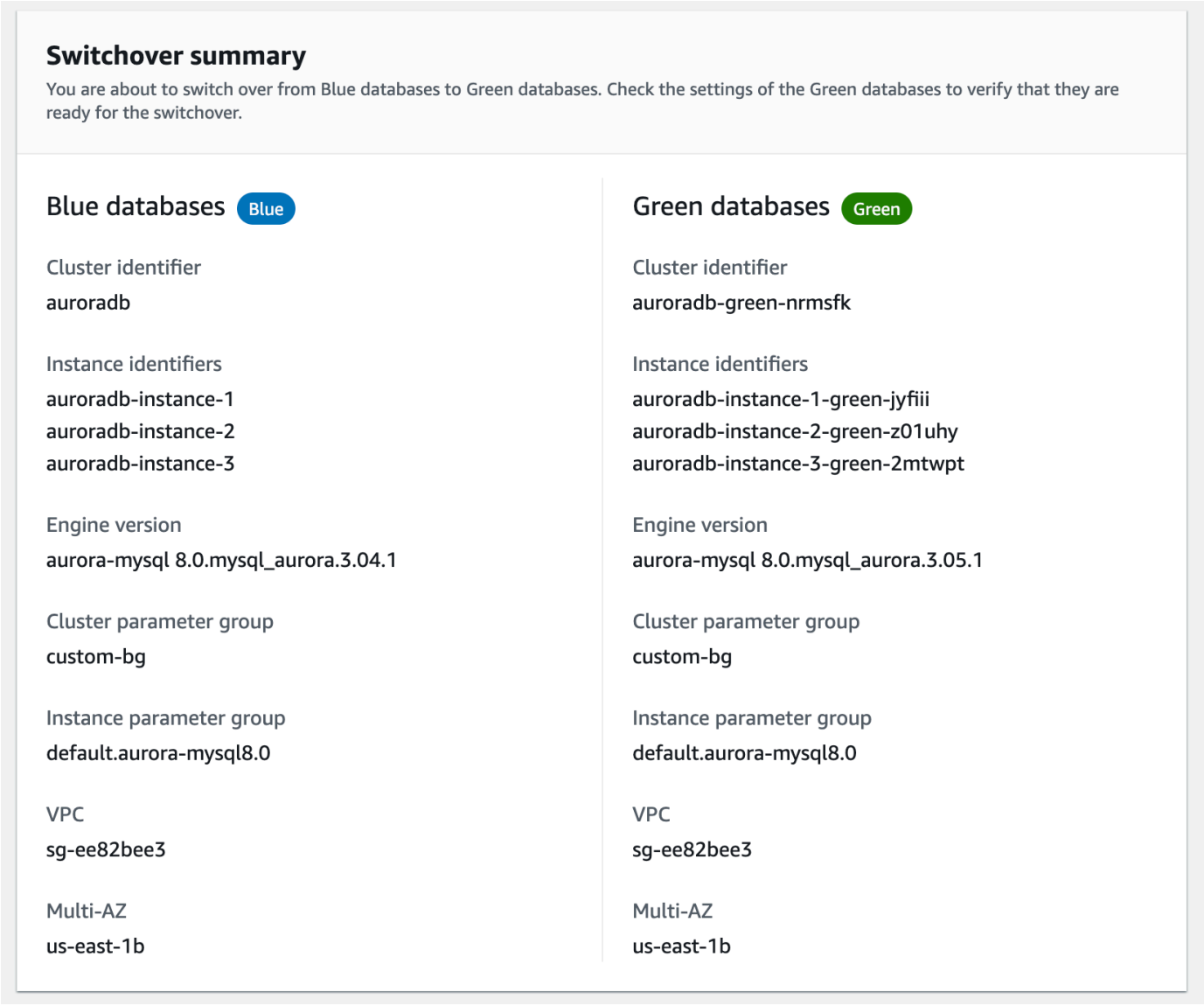
Puede conmutar una implementación azul/verde mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para conmutar una implementación azul/verde

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la implementación azul/verde que desee conmutar.
3. En Actions (Acciones), elija Switch over (Conmutar).

   Aparece la página Switch over (Conmutar).



4. En la página Switch over (Conmutar), consulte el resumen de la conmutación. Asegúrese de que los recursos de ambos entornos coincidan con lo que espera. Si no es así, seleccione Cancel (Cancelar).
5. En Ajustes de tiempo de espera, introduzca el límite de tiempo para la transición.
6. Si el clúster ejecuta Aurora PostgreSQL, revise y confirme las recomendaciones previas a la transición. Para obtener más información, consulte the section called “Limitaciones específicas de la replicación lógica para las implementaciones azul/verde”.
7. Elija Switch over (Conmutar).

AWS CLI

Para conmutar una implementación azul/verde mediante la AWS CLI, utilice el comando switchover-blue-green-deploy con las siguientes opciones:

- `--blue-green-deployment-identifier`: especifique el ID de recurso de la implementación azul/verde.
- `--switchover-timeout`: especifique el límite de tiempo para la conmutación, en segundos. El valor predeterminado es 300.

Example Conmutar una implementación azul/verde

Para Linux, macOS o Unix:

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

Para Windows:

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

API de RDS

Para conmutar una implementación azul/verde mediante la API de Amazon RDS, utilice la operación `SwitchoverBlueGreenDeployment` con los siguientes parámetros:

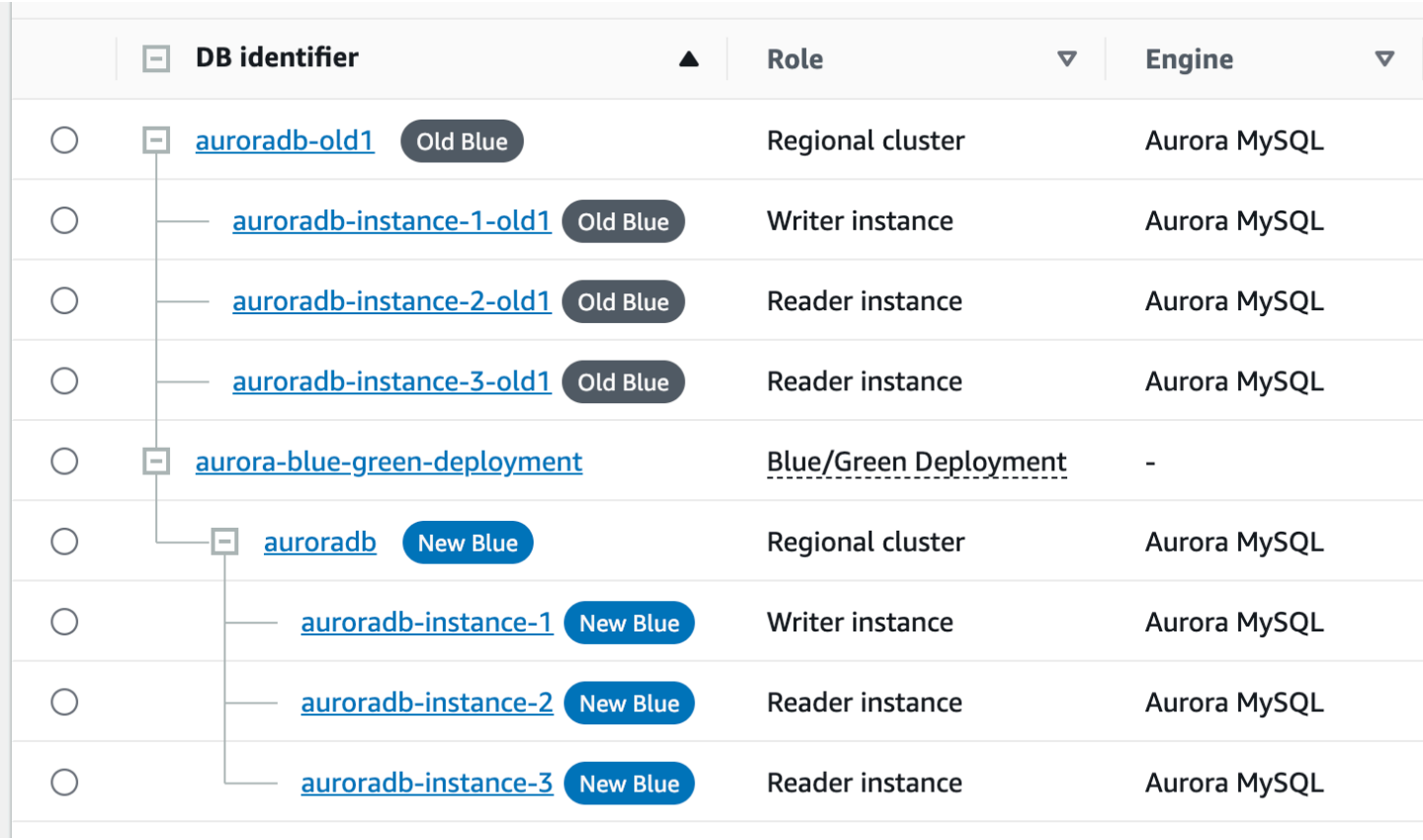
- `BlueGreenDeploymentIdentifier`: especifique el ID de recurso de la implementación azul/verde.
- `SwitchoverTimeout`: especifique el límite de tiempo para la conmutación, en segundos. El valor predeterminado es 300.

Después de la conmutación

Tras una conmutación, se conservan el clúster y las instancias de base de datos del entorno azul anterior. A estos recursos se les aplican los costos estándar. La replicación y el registro binario entre los entornos azul y verde se detienen.

RDS cambia el nombre del clúster y las instancias de base de datos en el entorno azul añadiendo `-old`*n* al nombre del recurso actual, donde *n* es un número. Se fuerza el clúster de base de datos azul a un estado de solo lectura. Incluso si activa las operaciones de escritura, permanece como de solo lectura hasta que elimine la implementación azul/verde. RDS cambia el nombre de las instancias de base de datos y el clúster de base de datos en el entorno verde `-new`*n*.

Si elimina el recurso de implementación azul/verde, RDS retiene los recursos `-old`*n* y `-new`*n*.



Actualización del nodo principal para los consumidores

RDS ofrece réplicas de lectura totalmente administradas. Sin embargo, también ofrece la opción de configurar réplicas autoadministradas, también conocidas como réplicas externas. Las réplicas externas permiten utilizar recursos de terceros como destinos de replicación.

Tras realizar la transición de una implementación azul/verde de Aurora MySQL, si el clúster de base de datos azul tenía réplicas externas o consumidores de registros binarios antes de la transición, debe actualizar su nodo principal tras la transición para mantener la continuidad de la replicación.

Para actualizar el nodo principal

1. Tras la transición, la instancia de base de datos de escritura que anteriormente estaba en el entorno verde emite un evento que contiene el nombre del archivo de registro maestro y la posición del registro maestro. Para localizar el evento, vaya a la consola de RDS y seleccione Eventos en el panel de navegación izquierdo.
2. Filtre por eventos en los que el origen sea el nombre de la antigua instancia de base de datos del escritor verde, antes de realizar la transición.
3. Localice el evento que contiene las coordenadas del registro binario. El mensaje del evento es similar a: `Binary log coordinates in green environment after switchover: file mysql-bin-changelog.000003 and position 40134574`.
4. Asegúrese de que el consumidor o la réplica hayan aplicado todos los registros binarios del antiguo entorno azul. A continuación, utilice las coordenadas del registro binario proporcionadas para reanudar la replicación en los consumidores. Por ejemplo, si ejecuta una réplica de MySQL en EC2, puede usar los siguientes comandos:

   MySQL 8.0.22 y versiones anteriores principales y secundarias

   ```
   CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-changelog.000003', MASTER_LOG_POS=40134574;
   ```

   MySQL 8.0.23 y versiones posteriores principales y secundarias

   ```
   CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
    SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
   ```

Eliminación de una implementación azul/verde en Amazon Aurora

Puede eliminar una implementación azul/verde antes o después de cambiarla.

Al eliminar una implementación azul/verde antes de cambiarla, Amazon RDS elimina opcionalmente el clúster de base de datos en el entorno verde:

- Si decide eliminar el clúster de base de datos en el entorno verde (`--delete-target`), debe tener desactivada la protección de eliminación.
- Si no elimina el clúster de base de datos en el entorno verde (`--no-delete-target`), eso significa que se retiene el clúster, pero ya no forma parte de una implementación azul/verde. En el caso de Aurora MySQL, la replicación continúa entre los entornos. En el caso de Aurora PostgreSQL, el entorno verde se convierte en un entorno independiente, por lo que se detiene la replicación.

La opción de eliminar las bases de datos verdes no está disponible en la consola después de la conmutación. Al eliminar las implementaciones azul/verde mediante la AWS CLI, no puede especificar la opción `--delete-target` si el estado de la implementación es `SWITCHOVER_COMPLETED`.

> **Important**
>
> Tras eliminar una implementación azul/verde, RDS elimina las protecciones de solo lectura del clúster de base de datos de producción anterior. Si el parámetro `read_only` está deshabilitado para el clúster de base de datos, volverá a permitir las operaciones de escritura.

Puede eliminar una implementación azul/verde mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

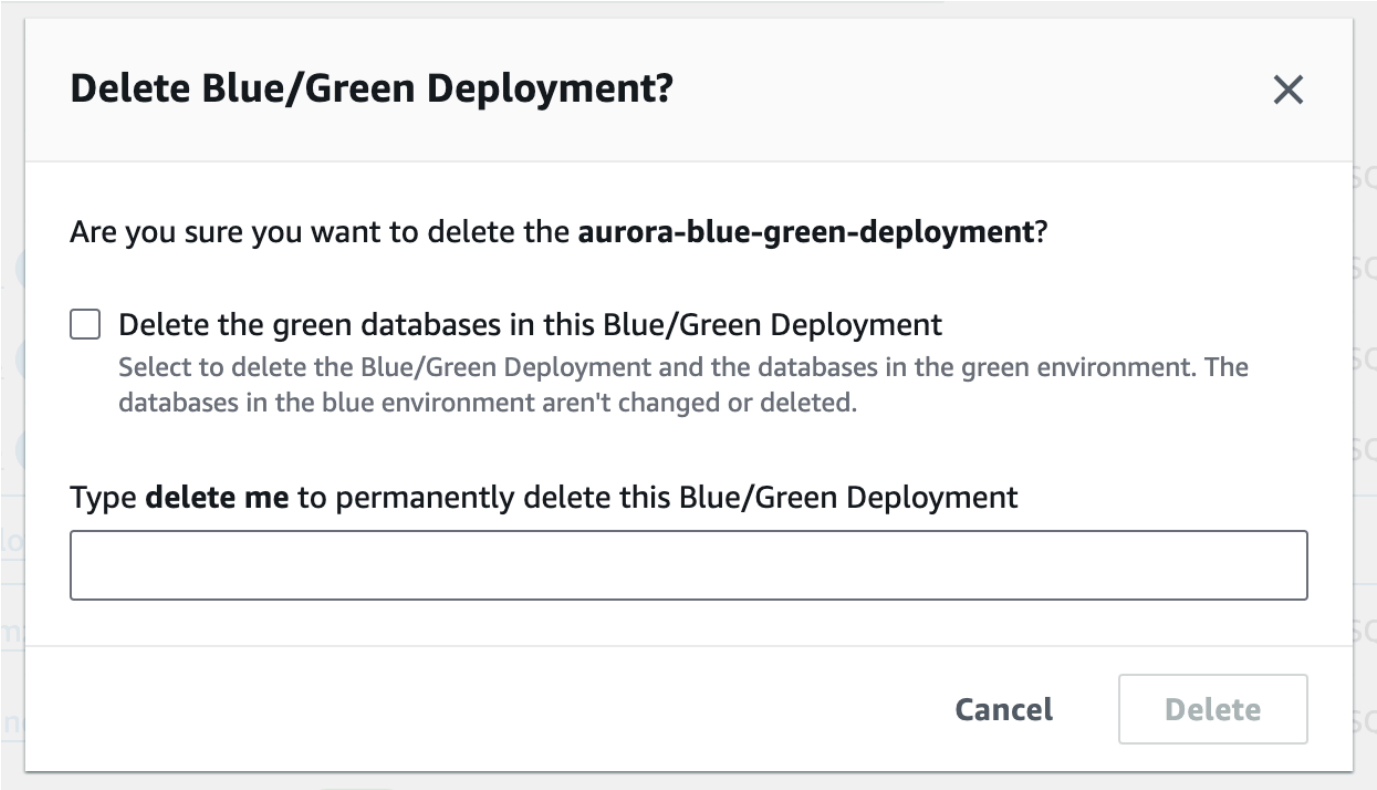
Consola

Para eliminar una implementación azul/verde

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la implementación azul/verde que desee eliminar.
3. En Actions (Acciones), seleccione Delete (Eliminar).

   Aparece la ventana ¿Eliminar la implementación azul/verde?



   Para eliminar las bases de datos verdes, seleccione Eliminar las bases de datos verdes en esta implementación azul/verde.
4. En el cuadro, escriba **`delete me`**.
5. Elija Eliminar.

AWS CLI

Para eliminar una implementación azul/verde mediante la AWS CLI, utilice el comando delete-blue-green-deployment con las siguientes opciones:

- `--blue-green-deployment-identifier`: el identificador de recurso de la implementación azul/verde que se va a eliminar.

- `--delete-target`: especifica que se elimina el clúster de base de datos del entorno verde. No puede especificar esta opción si la implementación azul/verde tiene un estado de `SWITCHOVER_COMPLETED`.
- `--no-delete-target`: especifica que se conserva el clúster de base de datos en el entorno verde.

Example Eliminar una implementación azul/verde y el clúster de base de datos del entorno verde

Para Linux, macOS o Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

Para Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Example Elimine una implementación azul/verde pero conserve el clúster de base de datos en el entorno verde

Para Linux, macOS o Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

Para Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

API de RDS

Para eliminar una implementación azul/verde mediante la API de Amazon RDS, utilice la operación `DeleteBlueGreenDeployment` con los siguientes parámetros:

- `BlueGreenDeploymentIdentifier`: el identificador de recurso de la implementación azul/verde que se va a eliminar.
- `DeleteTarget`: especifique `TRUE` para eliminar el clúster de base de datos en el entorno verde o para conservarlo. No puede ser `TRUE` si la implementación azul/verde tiene un estado de `SWITCHOVER_COMPLETED`.

Copias de seguridad y restauración de un clúster de base de datos de Amazon Aurora

Estos temas proporcionan información sobre la realización de copias de seguridad y la restauración de clústeres de bases de datos de Amazon Aurora.

> **Tip**
>
> Las características de alta disponibilidad de Aurora y las capacidades de copia de seguridad automática ayudan a mantener sus datos protegidos sin necesidad de una configuración exhaustiva por su parte. Antes de implementar una estrategia de copia de seguridad, obtenga información sobre las formas en que Aurora mantiene varias copias de los datos y lo ayuda a acceder a ellas en varias instancias de base de datos y regiones de AWS. Para obtener más información, consulte Alta disponibilidad para Amazon Aurora.

Temas

- Información general de copias de seguridad y restauración de un clúster de base de datos Aurora
- Retener copias de seguridad automatizadas
- Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora
- Creación de una instantánea de clúster de base de datos
- Restauración de una instantánea de clúster de base de datos
- Copia de una instantánea de clúster de base de datos
- Compartir una instantánea de clúster de base de datos
- Exportación de datos del clúster de base de datos a Amazon S3
- Exportación de datos de instantánea del clúster de bases de datos a Amazon S3
- Restauración de un clúster de base de dato a un momento indicado
- Eliminación de una instantánea de clúster de base de datos
- Tutorial: Restaurar un clúster de base de datos de Amazon Aurora desde una instantánea de clúster de base de datos

Información general de copias de seguridad y restauración de un clúster de base de datos Aurora

En los siguientes temas se describen las copias de seguridad de Aurora y cómo restaurar el clúster de base de datos de Aurora.

Contenido

- Copias de seguridad
  - Uso de AWS Backup
- Intervalo de copia de seguridad
- Restauración de datos
- Clonación de base de datos para Aurora
- Backtrack

Copias de seguridad

Aurora crea copias de seguridad del volumen de clúster automáticamente y retiene los datos de restauración durante la duración del periodo de retención de copia de seguridad. Las copias de seguridad automatizadas de Aurora son continuas y progresivas para que se pueda restaurar con rapidez a cualquier punto durante el periodo de retención de copia de seguridad. No se produce ningún impacto en el desempeño ni ninguna interrupción del servicio de base de datos durante la escritura de los datos de copia de seguridad. Puede especificar un periodo de retención de copia de seguridad de 1 a 35 días cuando cree o modifique un clúster de base de datos. Las copias de seguridad automatizadas de Aurora se almacenan en Amazon S3. Para obtener más información sobre la retención de copias de seguridad automatizadas, consulte Retener copias de seguridad automatizadas

Si desea conservar los datos más allá del periodo de retención de copia de seguridad, puede realizar una instantánea de los datos del volumen de clúster. Las instantáneas del clúster de base de datos de Aurora no caducan. Puede crear un nuevo clúster de bases de datos a partir de la instantánea. Para obtener más información, consulte Creación de una instantánea de clúster de base de datos.

> **Note**
>
> - Para todos los clústeres de base de datos de Amazon Aurora, el periodo de retención de copia de seguridad predeterminado es de un día con independencia del procedimiento empleado para crear el clúster.
> - No es posible desactivar las copias de seguridad automatizadas en Aurora. El clúster de base de datos administra el período de retención de copia de seguridad para Aurora.

Los costos de almacenamiento de copias de seguridad dependen de la cantidad de datos de copia de seguridad e instantáneas de Aurora y de cuánto tiempo los conserve. Para obtener más información sobre el almacenamiento asociado con las copias de seguridad y las instantáneas de Aurora, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora. Para obtener información sobre los precios del almacenamiento de copias de seguridad de Aurora, consulte Precios de Amazon RDS para Aurora. Después de que un clúster de Aurora asociado con una instantánea se borre, el almacenamiento de esa instantánea incurre en los costos de almacenamiento de copias de seguridad estándares de Aurora.

Uso de AWS Backup

También puede utilizar AWS Backup para administrar copias de seguridad de clústeres de base de datos de Amazon Aurora.

Las instantáneas administradas por AWS Backup se consideran instantáneas del clúster de base de datos manuales, pero no se cuentan para la cuota de instantáneas del clúster de base de datos para Aurora. Las instantáneas creadas con AWS Backup tienen nombres con `awsbackup:job-`*`AWS-Backup-job-number`*. Para obtener más información sobre AWS Backup, consulte AWS Backup Developer Guide.

También puede utilizar AWS Backup para administrar copias de seguridad automatizadas de clústeres de base de datos de Amazon Aurora. Si su clúster de base de datos está asociado a un plan de copia de seguridad en AWS Backup, puede utilizar ese plan para la recuperación en un momento dado. Las copias de seguridad automatizadas (continuas) administradas por AWS Backup tienen nombres con `continuous:cluster-`*`AWS-Backup-job-number`*. Para obtener más información, consulte Restauración de un clúster de base de datos a un momento especificado mediante AWS Backup.

Intervalo de copia de seguridad

Los backups automatizados se producen a diario durante la ventana de copia de seguridad preferida. Si la la copia de seguridad requiere más tiempo del asignado a la ventana de copia de seguridad, la copia de seguridad continúa cuando finaliza la ventana hasta que se completa. El periodo de copia de seguridad no se puede solapar con el periodo de mantenimiento semanal del clúster de base de datos.

Las copias de seguridad automatizadas de Aurora son continuas e incrementales, pero la ventana de copia de seguridad se utiliza para crear una copia de seguridad diaria del sistema que se conserva dentro del período de retención de la copia de seguridad. Puede copiar la copia de seguridad para conservarla fuera del período de retención.

> **Note**
>
> Cuando crea un clúster de base de datos mediante Consola de administración de AWS, no puede especificar una ventana de copia de seguridad. Sin embargo, puede especificar una ventana de copia de seguridad al crear un clúster de base de datos mediante la API AWS CLI o RDS.

Si no especifica un período de copia de seguridad preferido al crear el clúster de base de datos, Aurora asigna un período de copia de seguridad predeterminado de 30 minutos. Este periodo se selecciona al azar dentro de un bloque de 8 horas por cada Región de AWS. En la tabla siguiente se enumeran los bloques de tiempo para cada Región de AWS desde la que se asignan los periodos de copia de seguridad predeterminados.

Nombre de la región	Región	Bloque de tiempo
Este de EE. UU. (Norte de Virginia)	us-east-1	03:00–11:00 UTC
Este de EE. UU. (Ohio)	us-east-2	03:00 — 11:00 UTC
Oeste de EE. UU. (Norte de California)	us-west-1	06:00 — 14:00 UTC

Nombre de la región	Región	Bloque de tiempo
Oeste de EE. UU. (Oregón)	us-west-2	06:00–14:00 UTC
África (Ciudad del Cabo)	af-south-1	03:00–11:00 UTC
Asia-Pacífico (Hong Kong)	ap-east-1	06:00-14:00 UTC
Asia-Pacífico (Hyderabad)	ap-south-2	06:30 – 14:30 UTC
Asia-Pacífico (Yakarta)	ap-southeast-3	08:00 a 16:00 h UTC
Asia-Pacífico (Malasia)	ap-southeast-5	09:00–17:00 UTC
Asia-Pacífico (Melbourne)	ap-southeast-4	11:00–19:00 UTC
Asia Pacífico (Bombay)	ap-south-1	16:30 — 00:30 UTC
Asia-Pacífico (Nueva Zelanda)	ap-southeast-6	13:00-21:00 UTC
Asia Pacífico (Osaka)	ap-northeast-3	00:00 — 08:00 UTC
Asia-Pacífico (Seúl)	ap-northeast-2	13:00 — 21:00 UTC
Asia-Pacífico (Singapur)	ap-southeast-1	14:00 — 22:00 UTC
Asia Pacífico (Sídney)	ap-southeast-2	12:00 — 20:00 UTC
Asia-Pacífico (Taipéi)	ap-east-2	09:00-17:00 UTC

Nombre de la región	Región	Bloque de tiempo
Asia-Pacífico (Tailandia)	ap-southeast-7	08:00-16:00 UTC
Asia-Pacífico (Tokio)	ap-northeast-1	13:00 — 21:00 UTC
Canadá (centro)	ca-central-1	03:00-11:00 UTC
Oeste de Canadá (Calgary)	ca-west-1	18:00 — 02:00 UTC
China (Pekín)	cn-north-1	06:00–14:00 UTC
China (Ningxia)	cn-northwest-1	06:00–14:00 UTC
Europe (Fráncfort)	eu-central-1	20:00 — 04:00 UTC
Europe (Irlanda)	eu-west-1	22:00 — 06:00 UTC
Europe (Londres)	eu-west-2	22:00-06:00 UTC
Europa (Milán)	eu-south-1	02:00 — 10:00 UTC
Europa (París)	eu-west-3	07:29 — 14:29 UTC
Europa (España)	eu-south-2	02:00 — 10:00 UTC
Europa (Estocolmo)	eu-north-1	23:00 — 07:00 UTC
Europa (Zúrich)	eu-central-2	02:00 — 10:00 UTC
Israel (Tel Aviv)	il-central-1	03:00-11:00 UTC
México (central)	mx-central-1	19:00 — 03:00 UTC
Medio Oriente (Baréin)	me-south-1	06:00-14:00 UTC
Medio Oriente (EAU)	me-central-1	05:00 a 13:00 h UTC

Nombre de la región	Región	Bloque de tiempo
América del Sur (São Paulo)	sa-east-1	23:00 — 07:00 UTC
AWS GovCloud (Este de EE. UU.)	us-gov-east-1	17:00 — 01:00 UTC
AWS GovCloud (Oeste de EE. UU.)	us-gov-west-1	06:00–14:00 UTC

Restauración de datos

Puede recuperar sus datos si crea un nuevo clúster de base de datos Aurora a partir de los datos de copia de seguridad retenidos por Aurora, de una instantánea del clúster de base de datos que haya guardado o de una copia de seguridad automatizada retenida. Puede restaurar rápidamente una nueva copia del clúster de base de datos creada a partir de los datos de backup de cualquier momento dado durante el periodo de retención de copia de seguridad. La naturaleza continua e incremental de las copias de seguridad de Aurora durante el período de retención de copia de seguridad implica que no es necesario realizar instantáneas de los datos con demasiada frecuencia para mejorar los tiempos de restauración.

El último momento restaurable de un clúster de base de datos es el momento más reciente al que se puede restaurar dicho clúster. Por lo general, esto ocurre en un plazo de 5 minutos del momento actual en el caso de un clúster de base de datos activo o 5 minutos del momento de eliminación del clúster para una copia de seguridad automatizada retenida.

El primer momento restaurable especifica el primer momento del período de retención de copia de seguridad al que se puede restaurar el volumen del clúster.

Para determinar el primer o el último momento para el que se puede restaurar un clúster de base de datos, busque los valores `Latest restorable time` o `Earliest restorable time` en la consola de RDS. Para obtener más información acerca de cómo ver estos valores, consulte Visualización de copias de seguridad retenidas.

Puede determinar cuándo se ha completado la restauración de un clúster de base de datos comprobando los valores `Latest restorable time` y `Earliest restorable time`. Estos valores devolverán NULL hasta que la operación de restauración se haya completado. No puede

solicitar una operación de copia de seguridad o restauración si `Latest restorable time` o `Earliest restorable time` devuelven el valor NULL.

Para obtener más información acerca la restauración de un clúster de base de datos a un momento especificado, consulte Restauración de un clúster de base de dato a un momento indicado.

Clonación de base de datos para Aurora

También puede utilizar la clonación de bases de datos para clonar las bases de datos de su clúster de base de datos e Aurora en un nuevo clúster de base de datos, en lugar de restaurar una instantánea de clúster de base de datos. Las bases de datos clonadas usan un espacio adicional mínimo cuando se crean inicialmente. Los datos se copian solo cuando cambian los datos, ya sea en la bases de datos de origen o en las bases de datos clonadas. Puede crear varios clones desde el mismo clúster de base de datos o crear clones adicionales incluso desde otros clones. Para obtener más información, consulte Clonación de un volumen de clúster de base de datos de Amazon Aurora.

Backtrack

Aurora MySQL ahora admite "rebobinar" un clúster de base de datos a un momento específico, sin restaurar datos desde una copia de seguridad. Para obtener más información, consulte Búsqueda de datos anteriores de un clúster de base de datos de Aurora.

Retener copias de seguridad automatizadas

Al eliminar un clúster de base de datos aprovisionado o de Aurora Serverless v2, puede retener copias de seguridad automatizadas. Esto le permite restaurar un clúster de base de datos a un momento específico dentro del período de retención de la copia de seguridad, incluso después de eliminar el clúster.

Las copias de seguridad automatizadas contienen instantáneas del sistema y registros de transacciones de un clúster de base de datos. También incluyen las propiedades del clúster de base de datos, como la clase de instancia de base de datos, que son necesarias para restaurarlo a un clúster activo.

Puede restaurar o eliminar copias de seguridad automatizadas con la Consola de administración de AWS, la API de RDS y la AWS CLI.

> **Note**
>
> No puede retener copias de seguridad automatizadas para los clústeres de bases de datos de Aurora Serverless v1.

Temas

- Periodo de retención
- Visualización de copias de seguridad retenidas
- Costos de retención
- Limitaciones
- Eliminación de las copias de seguridad automatizadas retenidas

Periodo de retención

Las instantáneas del sistema y los registros de transacciones en una copia de seguridad automatizada retenida caducan del mismo modo que para el clúster de base de datos de origen. La configuración del período de retención del clúster de origen también se aplica a las copias de seguridad automatizadas. Dado que no se crean nuevas instantáneas ni registros para este clúster, las copias de seguridad automatizadas retenidas acaban por caducar por completo. Una vez

finalizado el período de retención, seguirá reteniendo las instantáneas manuales del clúster de base de datos, pero todas las copias de seguridad automatizadas caducarán.

Puede eliminar copias de seguridad automatizadas retenidas con la consola, la AWS CLI o la API de RDS. Para obtener más información, consulte Eliminación de las copias de seguridad automatizadas retenidas.

A diferencia de una copia de seguridad automatizada retenida, una instantánea final no caduca. Sugerimos intensamente que tome una instantánea final aunque conserve las copias de seguridad automatizadas, ya que las copias de seguridad conservadas vencen completamente al final.

Visualización de copias de seguridad retenidas

Para ver las copias de seguridad automatizadas retenidas, elija Copias de seguridad automatizadas en el panel de navegación y, a continuación, elija Retenidas. Para ver instantáneas individuales asociadas a una copia de seguridad automatizada retenida, elija Snapshots (Instantáneas) en el panel de navegación. También puede describir instantáneas individuales asociadas con una copia de seguridad automatizada conservada. Desde ahí, puede restaurar una instancia de base de datos directamente a partir de una de esas instantáneas.

Para describir las copias de seguridad automatizadas retenidas mediante AWS CLI, utilice el siguiente comando:

```
aws rds describe-db-cluster-automated-backups --db-cluster-resource-
id DB_cluster_resource_ID
```

Para describir sus copias de seguridad automatizadas retenidas mediante la API de RDS, llame a la `DescribeDBClusterAutomatedBackups` acción con el parámetro `DbClusterResourceId`.

Costos de retención

No se aplica ningún cargo adicional por el almacenamiento de copias de seguridad de hasta el 100 % del almacenamiento total de la base de datos de Aurora para cada clúster de base de datos de Aurora. Tampoco se aplica ningún cargo adicional hasta un día cuando se retienen copias de seguridad automatizadas después de eliminar un clúster de base de datos. Se cobrarán las copias de seguridad que se retengan durante más de un día.

No hay cobros adicionales para los registros de transacción o los metadatos de instancia. Todas las demás reglas de precios para copias de seguridad se aplican a los clústeres restaurables. Para obtener más información, consulte la Página de precios de Amazon Aurora.

Limitaciones

Las copias de seguridad automatizadas conservadas tienen las siguientes limitaciones:

- El número máximo de copias de seguridad automatizadas conservadas en una región de AWS es 40. No está incluido en la cuota de los clústeres de bases de datos. Puede tener hasta 40 clústeres de base de datos en ejecución, 40 instancias de base de datos en ejecución y 40 copias de seguridad automatizadas retenidas para clústeres de bases de datos al mismo tiempo.

  Para obtener más información, consulte Cuotas en Amazon Aurora.
- Las copias de seguridad automatizadas conservadas no contienen información sobre parámetros o grupos de opciones.
- Puede restaurar un clúster eliminado a un momento dado que esté dentro del período de retención en el momento de la eliminación.
- Una copia de seguridad automatizada retenida no se puede modificar, ya que consiste en copias de seguridad del sistema, registros de transacciones y las propiedades del clúster de base de datos que existían en el momento en el que eliminó el clúster de origen.

Eliminación de las copias de seguridad automatizadas retenidas

Puede eliminar las copias de seguridad automatizadas retenidas cuando no sean necesarias.

Consola

Para eliminar una copia de seguridad automatizada retenida

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Automated backups (Copias de seguridad automatizadas).
3. Elija la pestaña Retenidas.

4. Elija la copia de seguridad automatizada retenida que desea eliminar.

5. En Actions (Acciones), elija Delete (Eliminar).
6. En la página de confirmación, ingrese **`delete me`** y elija Delete (Eliminar).

AWS CLI

Puede eliminar una copia de seguridad automatizada retenida mediante el comando de la AWS CLI delete-db-cluster-automated-backup con la siguiente opción:

- `--db-cluster-resource-id`: el identificador de recurso para el clúster de base de datos de origen.

  Puede encontrar el identificador de recurso del clúster de base de datos de origen de una copia de seguridad automatizada retenida al ejecutar el comando de la AWS CLI describe-db-cluster-automated-backups.

Example

En este ejemplo, se elimina la copia de seguridad automatizada retenida para el clúster de base de datos de origen que tiene el ID del recurso `cluster-123ABCEXAMPLE`.

Para Linux, macOS o Unix:

```
aws rds delete-db-cluster-automated-backup \
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

En Windows:

```
aws rds delete-db-cluster-automated-backup ^
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

API de RDS

Puede eliminar una copia de seguridad automatizada retenida mediante la operación de API de Amazon RDS DeleteDBClusterAutomatedBackup con el siguiente parámetro:

- `DbClusterResourceId`: el identificador de recurso para el clúster de base de datos de origen.

  Puede encontrar el identificador de recurso de la instancia de base de datos de origen de una copia de seguridad automatizada retenida mediante la operación de la API de Amazon RDS DescribeDBClusterAutomatedBackups.

Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora

Amazon Aurora mantiene dos tipos de copias de seguridad: copias de seguridad automatizadas (continuas) e instantáneas.

Almacenamiento de copias de seguridad automatizadas

La copia de seguridad automatizada (continua) de un clúster almacena de forma incremental todos los cambios de la base de datos dentro de un período de retención específico para poder realizar una restauración a cualquier momento dentro de ese período de retención. Los períodos de retención pueden oscilar entre 1 y 35 días. Las copias de seguridad automatizadas son incrementales y se cobran en función de la cantidad de almacenamiento necesaria para la restauración a cualquier momento dentro del período de retención.

Aurora también ofrece una cantidad gratuita de uso de copias de seguridad. Esta cantidad de uso gratuita equivale al tamaño del volumen del clúster más reciente (tal como se representa en la métrica `VolumeBytesUsed` de Amazon CloudWatch). Esta cantidad se resta del uso calculado de la copia de seguridad automatizada. Además, las copias de seguridad automatizadas cuyo período de retención sea de solo 1 día son gratuitas.

Por ejemplo, la copia de seguridad automatizada tiene un período de retención de 7 días y desea restaurar el clúster a su estado de hace cuatro días. Aurora utiliza los datos incrementales almacenados en la copia de seguridad automatizada para volver a crear el estado del clúster en ese momento exacto hace cuatro días.

La copia de seguridad automatizada almacena toda la información necesaria para poder restaurar el clúster a cualquier momento del período de retención. Esto significa que almacena todos los cambios durante el período de retención, incluida la escritura de información nueva o la eliminación de la información existente. En el caso de las bases de datos en las que se producen muchos cambios, el tamaño de la copia de seguridad automatizada aumenta con el tiempo. Cuando una base de datos deje de experimentar cambios, puede esperar que el tamaño de la copia de seguridad automatizada disminuya, a medida que los cambios almacenados anteriormente salgan del período de retención.

El uso total facturado para la copia de seguridad automatizada nunca supera el tamaño del volumen acumulado del clúster durante el período de retención. Por ejemplo, si su período de retención es de 7 días y el volumen del clúster es de 100 GB todos los días, el uso de copias de seguridad automatizadas facturado nunca superará los 700 GB (100 GB * 7).

Almacenamiento de instantáneas

Las instantáneas de clúster de bases de datos son siempre copias de seguridad completas que capturan el tamaño del volumen del clúster cuando las crea. Tanto si toma instantáneas manualmente como a través de un plan de Copias de seguridad de AWS, Aurora las trata como instantáneas manuales. Aurora ofrece almacenamiento gratuito e ilimitado para las instantáneas dentro del periodo de retención de las copias de seguridad automatizadas. Después de que una instantánea manual esté fuera de este periodo, incurrirá en cargos por GB-mes. Las instantáneas del sistema automatizadas siguen siendo gratuitas a menos que las copie. Debido a que las copias de seguridad automatizadas no cubren las copias instantáneas, AWS siempre las factura.

Para obtener información general sobre las copias de seguridad de Aurora, consulte Copias de seguridad. Para obtener información sobre los precios del almacenamiento de copias de seguridad de Aurora, consulte la página de Precios de Amazon Aurora.

Métricas de Amazon CloudWatch para almacenamiento de copias de seguridad de Aurora

Puede monitorear los clústeres de Aurora y crear informes que utilicen las métricas de Amazon CloudWatch con la consola de CloudWatch. Puede utilizar las métricas de Amazon CloudWatch para revisar y monitorizar la cantidad de almacenamiento utilizado por sus copias de seguridad de Aurora. Estas métricas se calculan de forma independiente para cada clúster de base de datos de Aurora.

- `BackupRetentionPeriodStorageUsed` representa la cantidad de almacenamiento de copias de seguridad, en bytes, utilizado para almacenar copias de seguridad automáticas en el momento actual.
  - El valor depende del tamaño del volumen del clúster y de la cantidad de cambios (escrituras y actualizaciones) que se realicen en el clúster de base de datos durante el período de retención. Esto se debe a que la copia de seguridad automatizada debe almacenar todos los cambios incrementales realizados en el clúster para poder realizar una restauración a cualquier momento.
  - Esta métrica no resta el nivel gratuito de uso de copias de seguridad que proporciona Aurora.
  - Esta métrica emite un único punto de datos diario del uso de las copias de seguridad automatizadas registrado ese día.
- `SnapshotStorageUsed` representa la cantidad de almacenamiento de copias de seguridad utilizado, en bytes, para almacenar instantáneas manuales más allá del período de retención de la copia de seguridad automatizada.

- El valor depende del número de instantáneas que mantenga más allá del período de retención de la copia de seguridad automatizada y del tamaño de cada instantánea.
- El tamaño de cada instantánea es el tamaño del volumen del clúster en el momento en que se realiza la instantánea.
- Las instantáneas son copias de seguridad completas, no incrementales.
- Esta métrica emite un punto de datos diario por cada instantánea que se cobra. Para obtener el uso total diario de instantáneas, calcule la suma de esta métrica durante un período de 1 día.

- `TotalBackupStorageBilled` representa las métricas de todo el uso de copias de seguridad facturado, en bytes, para el clúster determinado:

  ```
  BackupRetentionPeriodStorageUsed + SnapshotStorageUsed - free tier
  ```

  - Esta métrica emite un punto de datos diario para el valor `BackupRetentionPeriodStorageUsed` menos el nivel gratuito de uso de copias de seguridad que proporciona Aurora. Este nivel gratuito equivale al último tamaño registrado del volumen del clúster de base de datos. Este punto de datos representa el uso facturado real de las copias de seguridad automatizadas.
  - Esta métrica emite puntos de datos diarios individuales para todos los valores `SnapshotStorageUsed`.
  - Para obtener el uso total diario de copias de seguridad facturado, calcule la suma de esta métrica durante un período de 1 día. Esto suma todo el uso de instantáneas facturado con el uso de copias de seguridad automatizadas facturado, para obtener el uso total de copias de seguridad facturado.

Para obtener más información acerca de cómo usar las métricas de CloudWatch, consulte Disponibilidad de métricas de Aurora en la consola de Amazon RDS..

Cálculo del uso del almacenamiento de copias de seguridad

El uso de una copia de seguridad automatizada se calcula teniendo en cuenta todos los registros incrementales que deben almacenarse para poder realizar una restauración a cualquier momento dentro del período de retención de la copia de seguridad. Estos cambios incluyen no solo el número de operaciones de escritura, sino también el tamaño y el alcance de las modificaciones de los datos. Cada tipo de operación (INSERT, UPDATE, DELETE) crea registros de cambios que se deben conservar para la recuperación en un momento dado. Por lo tanto, aunque dos bases de datos puedan tener el mismo número de operaciones de escritura (IOPS), sus requisitos de

almacenamiento de copia de seguridad podrían diferir significativamente en función del volumen de datos que se modifica en cada transacción.

Por ejemplo, tiene una copia de seguridad automatizada con un período de retención de 7 días. El tamaño del volumen de su clúster justo antes del período de retención era de 100 GB, por lo que esa es la cantidad mínima que Aurora tiene que almacenar. A continuación, tiene la siguiente actividad en los próximos 7 días, en la que el tamaño incremental del registro es la cantidad de almacenamiento necesaria para almacenar los registros de cambios procedentes de las escrituras y actualizaciones de la base de datos.

Día	Tamaño incremental del registro (GB)
1	10
2	15
3	25
4	20
5	10
6	25
7	30
Total	135

Estos datos significan que el uso calculado de la copia de seguridad automatizada para su copia de seguridad es el siguiente:

```
100 GB (volume size before retention period) + 135 GB (size of incremental records) =
 235 GB total backup usage
```

El uso facturado luego resta el nivel de uso gratuito. Suponga que el tamaño más reciente de su volumen es de 200 GB:

```
235 GB total backup usage - 200 GB (latest volume size) = 35 GB billed backup usage
```

Preguntas frecuentes

¿Cuándo se me facturan las instantáneas?

Se le facturarán las instantáneas manuales que estén fuera del período de retención (más antigua que) de la copia de seguridad automatizada.

¿Qué es una instantánea manual?

Una instantánea manual es una instantánea a la que se aplica una de las siguientes condiciones:

- Solicitada manualmente por usted
- Tomada por un servicio de copias de seguridad automatizado como AWS Backup
- Copiada de una instantánea del sistema automatizado para conservarla fuera del período de retención

¿Qué ocurre con mis instantáneas manuales si elimino mi clúster de base de datos?

Las instantáneas no caducan hasta que las elimine.

Cuando se elimina el clúster de base de datos, las instantáneas manuales que tomó con anterioridad siguen existiendo. Si esas instantáneas antes no se facturaban porque estaban dentro del período de retención de copias de seguridad automatizadas, ahora ya no están cubiertas y todas comienzan a facturarse a su tamaño completo por su uso.

¿Cómo puedo reducir mis costos de almacenamiento de copias de seguridad?

Hay algunas maneras de reducir los costos relacionados con el uso de las copias de seguridad:

- Elimine las instantáneas manuales que se encuentren fuera del período de retención de copias de seguridad automatizadas. Esto incluye las instantáneas que ha tomado usted y las instantáneas que podría haber tomado su plan de AWS Backup. Asegúrese de revisar su plan de AWS Backup para asegurarse de que no guarde instantáneas que no esperaba fuera del período de retención.
- Evalúe las escrituras y actualizaciones de la base de datos para ver si puede reducir la cantidad de cambios que realiza. Dado que nuestra copia de seguridad automatizada almacena todos los cambios incrementales dentro del período de retención, al reducir la cantidad de actualizaciones que haga también se reducen los gastos de copia de seguridad automatizada.
- Evalúe si tendría sentido reducir el período de retención de la copia de seguridad automatizada. La reducción del período de retención significa que la copia de seguridad almacena menos días de datos incrementales, lo que podría reducir el costo total de la copia de seguridad. Sin embargo, la reducción de este período de retención también podría provocar que algunas

instantáneas comiencen a facturarse porque ahora están fuera del período de retención. Asegúrese de comprobar todos los costos adicionales de instantáneas en los que podría incurrir antes de decidir si este es el curso de acción adecuado para usted.

¿Cómo se factura el almacenamiento de copias de seguridad?

El almacenamiento de copias de seguridad se factura por GB al mes.

Esto significa que el uso del almacenamiento de copias de seguridad se cobra como el promedio ponderado del uso durante el mes en cuestión. Estos son algunos ejemplos de un mes de 30 días:

- El uso de copias de seguridad facturado es de 100 GB durante los 30 días del mes. Su cargo es el siguiente:

```
(100 GB * 30) / 30 = 100 GB-month
```

- El uso de copias de seguridad facturado es de 100 GB durante los primeros 15 días del mes y, a continuación, de 0 GB durante los últimos 15. Su cargo es el siguiente:

```
(100 GB * 15 + 0 GB * 15) / 30 = 50 GB-month
```

- El uso de copias de seguridad facturado es de 50 GB durante los primeros 10 días del mes, 100 GB durante los siguientes 10 días y, luego, 150 GB durante los últimos 10 días. Su cargo es el siguiente:

```
(50 GB * 10 + 100 GB * 10 + 150 GB * 10) / 30 = 100 GB-month
```

¿Cómo afecta la configuración de búsqueda de datos anteriores de mi clúster de base de datos al uso del almacenamiento de copias de seguridad?

La configuración de búsqueda de datos anteriores para un clúster de base de datos de Aurora no afecta el volumen de datos de copias de seguridad para ese clúster. Amazon factura el almacenamiento para datos de búsquedas anteriores por separado. Para obtener información sobre los datos de búsquedas anteriores de Aurora, consulte la página de Precios de Amazon Aurora.

¿Cómo se aplican los costos de almacenamiento a las instantáneas compartidas?

Si comparte una instantánea con otro usuario, seguirá siendo el propietario de la misma. Los costos de almacenamiento se aplican al propietario de la instantánea. Si elimina una instantánea compartida de su propiedad, nadie podrá acceder a ella.

Para mantener el acceso a una instantánea compartida propiedad de otra persona, puede copiar la contraseña. Este procedimiento lo convierte en el propietario de la nueva contraseña. Los costos de almacenamiento para la instantánea copiada se aplican a su cuenta.

Para obtener más información sobre cómo compartir instantáneas, consulte Compartir una instantánea de clúster de base de datos. Para obtener más información sobre cómo copiar instantáneas, consulte Copia de una instantánea de clúster de base de datos.

Creación de una instantánea de clúster de base de datos

Amazon RDS crea una instantánea del volumen de almacenamiento del clúster de base de datos, creando una copia de seguridad de todo el clúster, y no solo de las bases de datos individuales. Cuando se crea una instantánea de un clúster de base de datos, se debe identificar la instancia del clúster de base de datos cuya copia de seguridad se va a realizar y, a continuación, se debe asignar un nombre a la instantánea del clúster de base de datos para poder restaurarla posteriormente. La cantidad de tiempo que tarda en crearse una instantánea de clúster de base de datos varía con el tamaño de sus bases de datos. Debido a que la instantánea incluye todo el volumen de almacenamiento, el tamaño de los archivos (por ejemplo, archivos temporales) también afecta la cantidad de tiempo que tarda en crearse la instantánea.

> **Note**
>
> Su instancia de base de datos debe tener el estado `available` para poder realizar una instantánea de la base de datos.

A diferencia de las copias de seguridad automatizadas, las instantáneas manuales no están sujetas al periodo de retención de copia de seguridad. Las instantáneas no caducan.

Para copias de seguridad a largo plazo, se recomienda exportar datos de instantáneas a Amazon S3. Si la versión principal de su motor de base de datos ya no es compatible, no puede restaurar a esa versión desde una instantánea. Para obtener más información, consulte Exportación de datos de instantánea del clúster de bases de datos a Amazon S3.

Puede crear una instantánea de clúster de base de datos usando la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para crear una instantánea de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).

   Aparece la lista Instantáneas manuales.
3. Elija Take Snapshot (Realizar una instantánea).

Aparece la ventana Take DB Snapshot (Realizar una instantánea de base de datos).

4. Para el Tipo de instantánea, seleccione Clúster de base de datos.
   Haga una instantánea de base de datos.
5. Seleccione el clúster de base de datos para el que desea tomar una instantánea.
6. Introduzca el nombre de la instantánea.
7. Elija Take Snapshot (Realizar una instantánea).

   Aparecerá la página Instantáneas manuales, con el estado de la nueva instantánea de clúster de base de datos mostrada como `Creating`. Después de que su estado es `Available`, puede ver su tiempo de creación.

AWS CLI

Cuando se crea una instantánea de un clúster de base de datos con la AWS CLI, se debe identificar el clúster de base de datos cuya copia de seguridad se va a realizar y, a continuación, se debe asignar un nombre a la instantánea del clúster de base de datos para poder restaurarla posteriormente. Puede hacerlo utilizando el comando AWS CLI de la `create-db-cluster-snapshot` con los siguientes parámetros:

- `--db-cluster-identifier`
- `--db-cluster-snapshot-identifier`

En este ejemplo, va a crear una instantánea de clúster de base de datos denominada *`mydbclústersnapshot`* para un clúster de base de datos denominado *`mydbclúster`*.

Example

Para Linux, macOS o:Unix

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mydbcluster \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

En:Windows

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mydbcluster ^
```

```
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API de RDS

Cuando se crea una instantánea de un clúster de base de datos con la API de Amazon RDS, se debe identificar el clúster de base de datos cuya copia de seguridad se va a realizar y, a continuación, se debe asignar un nombre a la instantánea del clúster de base de datos para poder restaurarla posteriormente. Para ello, use el comando `CreateDBClusterSnapshot` de la API de Amazon RDS con los siguientes parámetros:

- DBclústerIdentifier
- DBclústerSnapshotIdentifier

Determinación de si la instantánea del clúster de base de datos está disponible

Puede verificar que la instantánea del clúster de bases de datos está disponible si busca en Snapshots (Instantáneas) en la pestaña Maintenance & backups (Mantenimiento y copias de seguridad) de la página de detalles del clúster en la Consola de administración de AWS, si usa el comando `describe-db-cluster-snapshots` de la CLI o si usa la acción de la API `DescribeDBClusterSnapshots`.

También puede utilizar el comando de la CLI `wait db-cluster-snapshot-available` para sondear la API cada 30 segundos hasta que la instantánea esté disponible.

Restauración de una instantánea de clúster de base de datos

Amazon RDS crea una instantánea del volumen de almacenamiento del clúster de base de datos, creando una copia de seguridad de todo el clúster, y no solo de las bases de datos individuales. Para crear un nuevo clúster de base de datos, puede restaurar a partir de una instantánea de base de datos. Debe indicar el nombre de la instantánea del clúster de base de datos desde la que se hará la restauración y, después, indicar un nombre para el nuevo clúster de base de datos que se creó con la restauración. No puede restaurar desde una instantánea de un clúster de base de datos a un clúster existente; al restaurar se crea un nuevo clúster de base de datos.

Important

No puede restaurar una instantánea en una versión del motor de base de datos que haya superado la fecha de finalización del soporte estándar de Aurora. Solo puede acceder a una base de datos después de que se haya actualizado correctamente a una versión compatible. Para obtener más información sobre las versiones de motor de base de datos de Aurora admitidas, consulte Soporte extendido de Amazon RDS con Amazon Aurora.
Si se produce un error en la actualización a una versión compatible del clúster, el estado del clúster cambia a `upgrade_failed` y Aurora crea una instantánea final con el prefijo `rds-final`. Para acceder a la base de datos restaurada en la versión obsoleta tras un error de actualización, contacte con AWS Support.

Puede usar el clúster de base de datos restaurados tan pronto como su estado sea `available`.

Puede usar CloudFormation para restaurar un clúster de base de datos desde una instantánea de clúster de base de datos. Para obtener más información, consulte AWS::RDS::DBCluster en la AWS CloudFormationGuía del usuario.

Note

Si se comparte una instantánea manual de un clúster de base de datos, ya sea cifrada o sin cifrar, las cuentas autorizadas de AWS podrán restaurar directamente un clúster de base de datos a partir de la instantánea en lugar de hacer una copia de ella y restaurarla. Para obtener más información, consulte Compartir una instantánea de clúster de base de datos.

Para obtener información sobre la restauración de un clúster de base de datos Aurora o un clúster global con una versión del Soporte extendido de RDS, consulte Restauración de un clúster de base de datos de Aurora con Soporte extendido de Amazon RDS.

Consideraciones relativas al grupo de parámetros

Recomendamos retener el grupo de parámetros de base de datos y el grupo de parámetros del clúster de base de datos de todas las instantáneas de clúster de base de datos que cree para así poder asociar los grupos de parámetros correctos al clúster de base de datos restaurado.

El grupo de parámetros de base de datos predeterminado y el grupo de parámetros de clúster de base de datos se asocian al clúster restaurado, a menos que elija otros diferentes. No hay disponible ninguna configuración de parámetros personalizada en los grupos de parámetros predeterminados.

Puede especificar los grupos de parámetros al restaurar el clúster de base de datos.

Para obtener más información acerca de los grupos de parámetros de base de datos y grupos de parámetros de clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Consideraciones relativas al grupo de seguridad

Al restaurar un clúster de base de datos, la nube virtual privada (VPC) predeterminada, el grupo de subredes de base de datos y el grupo de seguridad de la VPC se asocian a la instancia restaurada, a menos que elija otras distintas.

- Si utiliza la consola de Amazon RDS, puede especificar un grupo de seguridad de VPC personalizado para asociarlo con el clúster o crear un nuevo grupo de seguridad de la VPC.
- Si utiliza la AWS CLI, puede especificar un grupo de seguridad de VPC personalizado para asociarlo con el clúster. Para ello, incluya la opción `--vpc-security-group-ids` en el comando `restore-db-cluster-from-snapshot`.
- Si está utilizando la API de Amazon RDS, puede incluir el parámetro `VpcSecurityGroupIds.VpcSecurityGroupId.N` en la acción `RestoreDBClusterFromSnapshot`.

En cuanto finalice la restauración y su nuevo clúster de base de datos esté disponible, también puede cambiar la configuración de la VPC mediante la modificación del clúster de base de datos. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Consideraciones sobre Amazon Aurora

Con Aurora, se restaura una instantánea de clúster de base de datos en un clúster de base de datos.

Con Aurora MySQL y Aurora PostgreSQL, puede restaurar una instantánea de un clúster de base de datos en un clúster de base de datos de Aurora Serverless. Para obtener más información, consulte Restauración de un clúster de bases de datos de Aurora Serverless v1.

Con Aurora MySQL, puede restaurar una instantánea de clúster de base de datos desde un clúster sin una consulta paralela a un clúster con consulta paralela. Debido a que la consulta paralela generalmente se usa con tablas muy grandes, el mecanismo de instantáneas es la forma más rápida de ingerir grandes volúmenes de datos en un clúster habilitado para consultas paralelas de Aurora MySQL. Para obtener más información, consulte Consulta paralela para Amazon Aurora MySQL.

Restauración a partir de una instantánea

Puede restaurar un clúster de base de datos desde una instantánea de clúster de base de datos utilizando la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para restaurar un clúster de base de datos desde una instantánea de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea de clúster de base de datos desde la que desea restaurar.
4. En Acciones, elija Restaurar instantánea.

   Aparece la página Restaurar instantánea.
5. Elija la versión del motor de base de datos a la que desea restaurar el clúster de base de datos.

   De forma predeterminada, la instantánea se restaura a la misma versión del motor de base de datos que el clúster de base de datos de origen, si esa versión está disponible.
6. En DB Instance Identifier (Identificador de instancias de bases de datos), escriba el nombre de la instancia de base de datos restaurada. Tenga en cuenta que Amazon RDS deriva el identificador del clúster de bases de datos del identificador de la instancia de base de datos que indique.
7. Especifique otras opciones, como la configuración de almacenamiento del clúster de base de datos.

Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.

8. Elija Restore DB Cluster (Restaurar clúster de base de datos).

AWS CLI

Para restaurar un clúster de base de datos desde una instantánea de clúster de base de datos, use el comando restore-db-cluster-from-snapshot de la AWS CLI.

En este ejemplo, se restaura a partir de una instantánea de clúster de base de datos creada previamente con el nombre `mydbclustersnapshot`. Restaura a un clúster de base de datos nuevo con el nombre `mynewdbcluster`.

Puede especificar otros ajustes, como la versión del motor de base de datos. Si no especifica una versión del motor, el clúster de base de datos se restaura a la versión del motor predeterminada.

Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.

Example

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine aurora-mysql|aurora-postgresql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewdbcluster ^
    --snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql|aurora-postgresql
```

Una vez restaurado el clúster de base de datos, debe añadirlo al grupo de seguridad que utilizaba el clúster de base de datos empleado para crear la instantánea de base de datos si desea tener la misma funcionalidad del clúster de base de datos anterior.

> **Important**
>
> Si usa la consola para restaurar un clúster de base de datos, Amazon RDS crea automáticamente la instancia de base de datos principal (escritor) del clúster de base de datos. Si usa la AWS CLI para restaurar un clúster de base de datos, debe crear expresamente la instancia principal del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos. Si no crea la instancia de base de datos principal, los puntos de conexión del clúster de base de datos permanecen con el estado `creating`.
> Llame al comando create-db-instance de la AWS CLI para crear la instancia principal del clúster de base de datos. Incluya el nombre del clúster de base de datos como valor de la opción `--db-cluster-identifier`.

API de RDS

Para restaurar un clúster de base de datos desde una instantánea de clúster de base de datos, llame a la operación de API RDS RestoreDBClusterFromSnapshot con los parámetros siguientes:

- `DBClusterIdentifier`
- `SnapshotIdentifier`

> **Important**
>
> Si usa la consola para restaurar un clúster de base de datos, Amazon RDS crea automáticamente la instancia de base de datos principal (escritor) del clúster de base de datos. Si usa la API de RDS para restaurar un clúster de base de datos, debe crear expresamente la instancia principal del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos. Si no crea la instancia de base de datos principal, los puntos de conexión del clúster de base de datos permanecen con el estado `creating`.
> Llame a la operación de API de RDS CreateDBInstance para crear la instancia principal para el clúster de base de datos. Incluya el nombre del clúster de base de datos como valor del parámetro `DBClusterIdentifier`.

Copia de una instantánea de clúster de base de datos

Con Amazon Aurora, puede copiar copias de seguridad automatizadas o instantáneas de clúster de bases de datos manuales. Después de copiar una instantánea, la copia es una instantánea manual. Puede hacer varias copias de una copia de seguridad automatizada o instantánea manual, pero cada copia debe tener un identificador único.

Puede copiar una instantánea en la misma Región de AWS, entre Regiones de AWS y puede copiar instantáneas compartidas. Puede copiar las instantáneas a otra Región de AWS o cuenta en un solo paso.

> **Note**
>
> Amazon le factura en función de la cantidad de datos de copias de seguridad e instantáneas de Amazon Aurora que conserve y el periodo de tiempo que los conserve. Para obtener más información sobre el almacenamiento asociado con las copias de seguridad y las instantáneas de Aurora, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora. Para obtener más información acerca de los precios de almacenamiento de Aurora, consulte Precios de Amazon RDS para Aurora.

Revise las limitaciones y los factores relacionados con la copia de instantáneas de clústeres de bases de datos. Para copiar instantáneas de clústeres de bases de datos, consulte uno de los siguientes temas.

- Copia de una instantánea de clúster de base de datos con la Consola de administración de AWS
- Copia de una instantánea de clúster de base de datos sin cifrar con la AWS CLI o la API de Amazon RDS
- Copia de una instantánea de clúster de base de datos cifrada con la AWS CLI o la API de Amazon RDS
- Copia de una instantánea de clúster de base de datos entre cuentas

Copia de una instantánea de clúster de base de datos con la Consola de administración de AWS

Use los procedimientos de este tema para copiar una instantánea de clúster de base de datos. Si el motor de base de datos de origen es Aurora, su instantánea es una instantánea de clúster de base de datos.

Para cada cuenta AWS, puede copiar hasta cinco instantáneas de clúster de base de datos a la vez de una Región de AWS a otra. Puede copiar instantáneas de clúster de base de datos cifradas y sin cifrar. Si copia una instantánea de clúster de base de datos en otra Región de AWS crea una instantánea de clúster de base de datos manual que se conserva en esa Región de AWS. Al copiar una instantánea de clúster de base de datos fuera de la Región de AWS de origen, se producen cargos por transferencia de datos de Amazon RDS.

Para obtener más información acerca de los precios de las transferencias de datos, consulte Precios de Amazon RDS.

Una vez que la copia de la instantánea de clúster de base de datos se ha creado en la nueva Región de AWS, se comporta como los demás instantáneas de clúster de base de datos de esa Región de AWS.

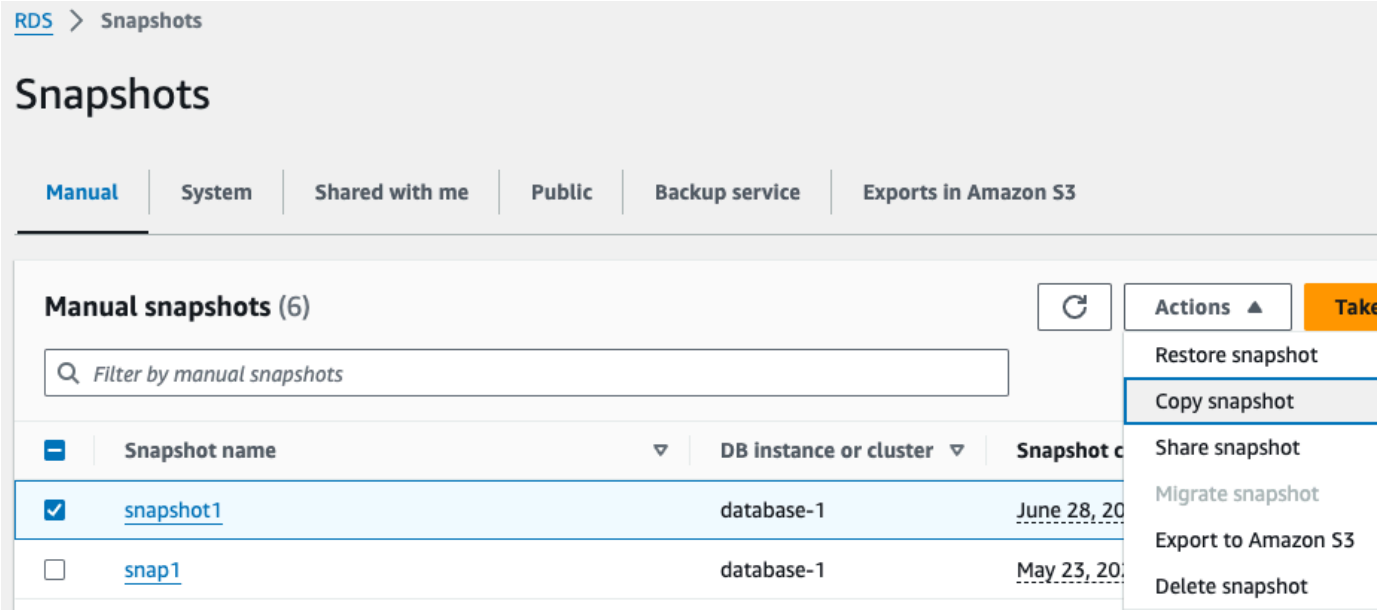
Este procedimiento sirve para copiar instantáneas de clúster de base de datos cifradas o sin cifrar, en la misma Región de AWS o entre regiones.

Para cancelar una operación de copia una vez que está en curso, elimine la instantánea del clúster de base de datos de destino mientras está en el estado copying.

Antes de copiar una instantánea de clúster de base de datos, revise Limitaciones y Factores importantes sobre la copia de instantáneas.

Para copiar una instantánea de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Seleccione la instantánea del clúster de base de datos que quiera copiar.
4. Seleccione Acciones y, a continuación, Eliminar instantánea.



5. (Opcional) Para copiar la instantánea de clúster de base de datos en una Región de AWS diferente, elija esa Región de AWS en Destination Region (Región de destino).
6. Introduzca el nombre de la copia de la instantánea del clúster de base de datos en New DB Snapshot Identifier (Nuevo identificador de instantánea de base de datos).
7. Para copiar las etiquetas y los valores de la instantánea en la copia de la instantánea, elija Copy Tags.
8. Elija Copy Snapshot.

Limitaciones

A continuación se indican algunas limitaciones al copiar instantáneas:

- No puede copiar una instantánea en o desde las siguientes:Regiones de AWS
  - China (Pekín)
  - China (Ningxia)
- Puede copiar una instantánea entre AWS GovCloud (EE. UU. Este) y AWS GovCloud (EE. UU. Oeste). Sin embargo, no puede copiar una instantánea entre estas AWS GovCloud (US) y las Regiones de AWS comerciales.
- Si elimina una instantánea de origen antes de que la instantánea de destino esté disponible, la copia de la instantánea podría generar un error. Compruebe que la instantánea de destino tiene el estado `AVAILABLE` antes de eliminar una instantánea de origen.

- Puede tener hasta cinco solicitudes de copia de instantánea en curso en una única región de destino por cuenta.
- Si solicita varias copias de instantáneas de la misma instancia de base de datos de origen, se ponen a la cola internamente. Las copias solicitadas posteriormente no se iniciarán hasta que se completen las copias de instantáneas anteriores. Para obtener más información, consulte el tema sobre por qué la creación de instantáneas de EBS o la AMI de EC2 es lenta en el Centro de conocimientos de AWS.
- Dependiendo de las Regiones de AWS implicadas y de la cantidad de datos que se vayan a copiar, una copia de instantánea entre regiones puede tardar horas en completarse. En algunos casos, puede haber un gran número de solicitudes de copia de instantáneas entre regiones desde una región. En estos casos, Amazon RDS puede poner nuevas solicitudes de copia entre regiones desde esa región de origen en una cola hasta que alguna de las copias en curso se complete. No se muestra ninguna información de progreso sobre las solicitudes de copia mientras están en la cola. La información de progreso se muestra cuando comienza la copia.
- Aurora no admite las instantáneas incrementales. Las copias de instantáneas de clúster de base de datos Aurora siempre se almacenan como copias completas. Una copia de la instantánea completa contiene todos los datos y metadatos necesarios para restaurar el clúster de base de datos.

Factores importantes sobre la copia de instantáneas

A continuación, se presentan los factores que deben tenerse en cuenta al copiar instantáneas.

Temas

- Factores importantes sobre la copia de instantáneas compartidas
- Factores importantes sobre la copia de instantáneas cifradas de clúster de base de datos
- Factores importantes sobre la copia de instantáneas entre regiones
- Factores importantes sobre grupos de parámetros

Factores importantes sobre la copia de instantáneas compartidas

Puede copiar instantáneas compartidas con usted por otras cuentas de AWS. En algunos casos, puede copiar una instantánea cifrada que se ha compartido desde otra cuenta de AWS. En estos casos, debe tener acceso a la AWS KMS key que se utilizó para cifrar la instantánea. Para obtener más información, consulte Cómo compartir instantáneas cifradas.

Copia entre regiones y entre cuentas en un solo paso

Para copiar una instantánea entre regiones y cuentas en una sola acción, primero debe compartir la instantánea con la cuenta de AWS de destino. Si la instantánea está cifrada, también debe compartir la clave de AWS KMS con la cuenta de AWS de destino. Si la instantánea está cifrada con la clave de AWS KMS predeterminada, primero debe copiarla para volver a cifrarla con una clave administrada por el cliente antes de compartirla con la cuenta de destino. Una vez compartida, puede iniciar una copia en esa cuenta (dentro o entre regiones) desde la cuenta de destino.

Factores importantes sobre la copia de instantáneas cifradas de clúster de base de datos

Puede copiar una instantánea que se haya cifrado con una clave de KMS. Si copia una instantánea cifrada, la copia de la instantánea se debe cifrar también. Si copia una instantánea cifrada dentro de la misma Región de AWS, puede cifrar la copia con la misma clave de KMS que la instantánea original. O bien puede especificar una clave de KMS diferente.

Si copia una instantánea cifrada entre regiones, debe especificar una clave de KMS valida en la de destino Región de AWS. Puede ser una clave de KMS específica de la región o una clave de varias regiones. Para obtener más información sobre las claves de varias regiones, consulte Uso de claves de varias regiones en AWS KMS.

Para obtener más información sobre la administración de claves de AWS KMS para Amazon RDS, consulte Administración de AWS KMS key.

La instantánea de origen permanece cifrada durante todo el proceso de copia. Para obtener más información, consulte Limitaciones de los clústeres de base de datos cifrados de Amazon Aurora.

> **Note**
>
> Para instantáneas del clúster de base de datos Amazon Aurora, no es posible cifrar una instantánea del clúster de base de datos sin cifrar al copiar la instantánea.

Para copiar instantáneas cifradas de clúster de base de datos, consulte los siguientes temas.

- Copia de una instantánea de clúster de base de datos cifrada con la AWS CLI o la API de Amazon RDS
- Copia de una instantánea de clúster de base de datos entre cuentas

Factores importantes sobre la copia de instantáneas entre regiones

Puede copiar instantáneas de clúster de base de datos en Regiones de AWS. Sin embargo, existen ciertas restricciones y consideraciones para la copia de instantáneas entre regiones.

Dependiendo de las Regiones de AWS implicadas y de la cantidad de datos que se vayan a copiar, una copia de instantánea entre regiones puede tardar horas en completarse.

En algunos casos, puede haber un gran número de solicitudes de copia de instantáneas entre regiones desde una Región de AWS. En estos casos, Amazon RDS puede poner nuevas solicitudes de copia entre regiones desde esa Región de AWS de origen en una cola hasta que alguna de las copias en curso se complete. No se muestra ninguna información de progreso sobre las solicitudes de copia mientras están en la cola. La información de progreso se muestra cuando comienza la copia.

Se aplican cargos por transferencia de datos a la copia de instantáneas entre regiones. La copia de instantáneas entre regiones crea copias completas de los datos de destino, pero los cargos por transferencia de datos son incrementales. Los datos incrementales incluyen tanto los nuevos datos que se han añadido a una base de datos de cliente desde la última copia como cualquier cambio realizado en los datos existentes. Para obtener más información, consulte Creating backup copies across Regiones de AWS en la Guía para desarrolladores de AWS Backup.

> **Note**
>
> Aurora copia la cantidad mínima de datos necesaria para crear una copia completa de una instantánea en la región de destino. Se aplican cargos por transferencia de datos al copiar instantáneas entre regiones.

Factores importantes sobre grupos de parámetros

Cuando se copia una instantánea entre regiones, la copia no incluye el grupo de parámetros empleado por el clúster de base de datos original. Cuando se restaura una instantánea para crear un nuevo clúster de base de datos, el clúster de base de datos usa el grupo de parámetros predeterminado para la Región de AWS en la que se creó. Para aplicar al clúster de la base de datos los mismos parámetros que al original, se debe hacer lo siguiente:

1. En la Región de AWS de destino, cree un grupo de parámetros del clúster de base de datos con la misma configuración que el clúster de base de datos original. Si ya existe uno en la nueva Región de AWS, puede usarlo.
2. Después de restaurar la instantánea en la Región de AWS de destino, modifique el nuevo clúster de base de datos y agregue el grupo de parámetros nuevo o ya existente del paso anterior.

Copia de una instantánea de clúster de base de datos sin cifrar con la AWS CLI o la API de Amazon RDS

Utilice los procedimientos que se describen en las siguientes secciones para copiar una instantánea de clúster de base de datos sin cifrar con la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

Para cancelar una operación de copia una vez que está en curso, elimine la instantánea del clúster de base de datos de destino identificado por `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mientras está en el estado copying.

Consola

Para copiar una instantánea de clúster de base de datos mediante la Consola de administración de AWS, consulte Copia de una instantánea de clúster de base de datos con la Consola de administración de AWS.

AWS CLI

Para copiar una instantánea del clúster, utilice el comando copy-db-clúster-snapshot de la AWS CLI. Si desea copiar la instantánea en otra Región de AWS , ejecute el comando en la Región de AWS en la que se va a copiar la instantánea.

Las siguientes opciones se usan para copiar una instantánea de clúster de base de datos sin cifrar:

- `--source-db-cluster-snapshot-identifier`: identificador de la instantánea del clúster de base de datos que se va a copiar. Si desea copiar la instantánea en otra Región de AWS, este identificador debe estar en el formato de ARN para la Región de AWS de origen.
- `--target-db-cluster-snapshot-identifier`: identificador de la nueva copia de la instantánea de clúster de base de datos.

El código siguiente crea una copia de la instantánea del clúster de base de datos `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` llamado `myclustersnapshotcopy` en la Región de AWS en la que se ejecuta el comando. Cuando se crea la copia, todas las etiquetas de la instantánea original se copian en la copia de la instantánea.

Example

Para Linux, macOS o:Unix

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --copy-tags
```

En:Windows

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --copy-tags
```

API de RDS

Para copiar una instantánea de clúster de base de datos, utilice la operación CopyDBclústerSnapshot de la API de Amazon RDS. Si desea copiar la instantánea en otra Región de AWS, lleve a cabo la acción en la Región de AWS en la que se va a copiar la instantánea.

Los siguientes parámetros se usan para copiar una instantánea de clúster de base de datos sin cifrar:

- `SourceDBClusterSnapshotIdentifier`: identificador de la instantánea del clúster de base de datos que se va a copiar. Si desea copiar la instantánea en otra Región de AWS, este identificador debe estar en el formato de ARN para la Región de AWS de origen.
- `TargetDBClusterSnapshotIdentifier`: identificador de la nueva copia de la instantánea de clúster de base de datos.

El siguiente código crea una copia de una instantánea `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` llamada `myclustersnapshotcopy` en la región EE.UU. Oeste (Norte de California). Cuando se crea la copia, todas las etiquetas de la instantánea original se copian en la copia de la instantánea.

Example

```
https://rds.us-west-1.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20130805
   &TargetDBSnapshotIdentifier=myclustersnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copia de una instantánea de clúster de base de datos cifrada con la AWS CLI o la API de Amazon RDS

Utilice los procedimientos que se describen en las siguientes secciones para copiar una instantánea de clúster de base de datos cifrada con la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

Para cancelar una operación de copia una vez que está en curso, elimine la instantánea del clúster de base de datos de destino identificado por `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mientras está en el estado copying.

Consola

Para copiar una instantánea de clúster de base de datos mediante la Consola de administración de AWS, consulte Copia de una instantánea de clúster de base de datos con la Consola de administración de AWS.

AWS CLI

Para copiar una instantánea del clúster, utilice el comando copy-db-clúster-snapshot de la AWS CLI. Si desea copiar la instantánea en otra Región de AWS , ejecute el comando en la Región de AWS en la que se va a copiar la instantánea.

Las siguientes opciones se usan para copiar una instantánea de clúster de base de datos cifrada:

- `--source-db-cluster-snapshot-identifier`: identificador de la instantánea del clúster de base de datos cifrada que se va a copiar. Si desea copiar la instantánea en otra Región de AWS, este identificador debe estar en el formato de ARN para la Región de AWS de origen.
- `--target-db-cluster-snapshot-identifier`: identificador de la nueva copia de la instantánea de clúster de base de datos cifrada.
- `--kms-key-id`: identificador de la clave de KMS que se va a utilizar para cifrar la copia de la instantánea del clúster de base de datos.

  Puede utilizar esta opción si la instantánea del clúster de base de datos está cifrada, si la instantánea se va a copiar en la misma Región de AWS y si desea especificar una nueva clave de KMS para cifrar la copia. De lo contrario, la copia de la instantánea del clúster de base de datos se cifra con la misma clave de KMS que la instantánea del clúster de base de datos de origen.

  Debe usar esta opción si la instantánea del clúster de base de datos está cifrada y va a copiar la instantánea en otra Región de AWS. En ese caso, debe especificar una clave de KMS para la Región de AWS de destino.

El siguiente ejemplo de código copia la instantánea del clúster de base de datos cifrada de la región EE.UU. Oeste (Oregón) a la región US East (N. Virginia). El comando se llama en la región US East (N. Virginia).

Example

Para Linux, macOS o:Unix

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --kms-key-id my-us-east-1-key
```

En:Windows

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:aurora-cluster1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --kms-key-id my-us-east-1-key
```

El parámetro `--source-region` es necesario cuando se copia una instantánea de clúster de base de datos cifrada entre las regiones AWSGovCloud (Este de EE. UU.) y AWS GovCloud (EE. UU. Oeste). Para `--source-region`, especifique la Región de AWS de la instancia de base de datos de origen. La Región de AWS especificada en `source-db-cluster-snapshot-identifier` debe coincidir con la Región de AWS especificada para `--source-region`.

Si `--source-region` no se especificó, especifique un valor de `--pre-signed-url`. Una URL prefirmada es una URL que contiene una solicitud firmada de Signature Version 4 para el comando `copy-db-cluster-snapshot` que se llama en la Región de AWS de origen. Para obtener más información acerca de la opcion `pre-signed-url`, consulte copy-db-clúster-snapshot en la Referencia de los comandos de AWS CLI.

API de RDS

Para copiar una instantánea de clúster de base de datos, utilice la operación CopyDBclústerSnapshot de la API de Amazon RDS. Si desea copiar la instantánea en otra Región de AWS, lleve a cabo la acción en la Región de AWS en la que se va a copiar la instantánea.

Los siguientes parámetros se usan para copiar una instantánea de clúster de base de datos cifrada:

- `SourceDBClusterSnapshotIdentifier`: identificador de la instantánea del clúster de base de datos cifrada que se va a copiar. Si desea copiar la instantánea en otra Región de AWS, este identificador debe estar en el formato de ARN para la Región de AWS de origen.
- `TargetDBClusterSnapshotIdentifier`: identificador de la nueva copia de la instantánea de clúster de base de datos cifrada.
- `KmsKeyId`: identificador de la clave de KMS que se va a utilizar para cifrar la copia de la instantánea del clúster de base de datos.

  Puede utilizar este parámetro si la instantánea del clúster de base de datos está cifrada, si la instantánea se va a copiar en la misma Región de AWS y si desea especificar una nueva clave de KMS que se utilizará para cifrar la copia. De lo contrario, la copia de la instantánea del clúster de base de datos se cifra con la misma clave de KMS que la instantánea del clúster de base de datos de origen.

Debe usar este parámetro si la instantánea del clúster de base de datos está cifrada y va a copiar la instantánea en otra Región de AWS. En ese caso, debe especificar una clave de KMS para la Región de AWS de destino.

- `PreSignedUrl`: si desea copiar la instantánea en otra Región de AWS, debe especificar el parámetro de `PreSignedUrl`. El valor de `PreSignedUrl` debe ser una URL que contenga una solicitud firmada de Signature Version 4 para la acción `CopyDBClusterSnapshot` que se debe llamar en la Región de AWS de origen desde la que se va a copiar la instantánea de clúster de base de datos. Para obtener más información acerca del uso de una URL prefirmada, consulte CopyDBclústerSnapshot.

El siguiente ejemplo de código copia la instantánea del clúster de base de datos cifrada de la región EE.UU. Oeste (Oregón) a la región US East (N. Virginia). La acción se llama en la región US East (N. Virginia).

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBClusterSnapshot
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBClusterSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBClusterSnapshotIdentifier%253Darn%25253Aaws%25253Ards
%25253Aus-west-2%25253A123456789012%25253Acluster-snapshot%25253Aaurora-cluster1-
snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
```

```
    &SourceDBClusterSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20161115
    &TargetDBClusterSnapshotIdentifier=myclustersnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

El parámetro `PreSignedUrl` es necesario cuando se copia una instantánea de clúster de base de datos cifrada entre las regiones AWSGovCloud (Este de EE. UU.) y AWS GovCloud (EE. UU. Oeste). El valor de `PreSignedUrl` debe ser una URL que contenga una solicitud firmada de Signature Version 4 para la operación `CopyDBClusterSnapshot` que se debe llamar en la Región de AWS de origen desde la que se va a copiar la instantánea de clúster de base de datos. Para obtener más información acerca del uso de una URL prefirmada, consulte CopyDBclústerSnapshot en la Amazon RDS API Reference.

Para generar una URL prefirmada de forma automática y no manual, use el comando copy-db-clúster-snapshot de la AWS CLI con la opción `--source-region`.

Copia de una instantánea de clúster de base de datos entre cuentas

Puede habilitar otras cuentas de AWS para copiar las instantáneas de clúster de base de datos que especifique mediante las acciones `ModifyDBClusterSnapshotAttribute` de la API de Amazon RDS y de `CopyDBClusterSnapshot`. Solo puede copiar instantáneas de clúster de base de datos entre cuentas en la misma Región de AWS. El proceso de copia entre cuentas funciona del modo que se describe a continuación, donde la cuenta A hace que la instantánea esté disponible para la copia y la cuenta B lo copia.

1. Con la cuenta A, llame a `ModifyDBClusterSnapshotAttribute` y especifique **`restore`** para el parámetro `AttributeName` y el ID de la cuenta B para el parámetro `ValuesToAdd`.
2. (Si la instantánea está cifrada) Con la cuenta A, actualice la política de claves para la clave de KMS. Para ello, añada primero el ARN de la cuenta B como `Principal` y a continuación permita la acción `kms:CreateGrant`.
3. (Si la instantánea está cifrada) Con la cuenta B, elija o cree un usuario y asocie a ese usuario una política de IAM que le permita copiar una instantánea de clúster de base de datos cifrada usando la clave de KMS.

4. Con la cuenta B, llame a `CopyDBClusterSnapshot` y use el parámetro `SourceDBClusterSnapshotIdentifier` para especificar el ARN de la instantánea del clúster de base de datos que se va a copiar, que debe incluir el ID de la cuenta A.

Para ver una lista de todas las AWS cuentas con permiso para restaurar una instantánea de clúster de base de datos, use la operación DescribeDBSnapshotAttributes or DescribeDBclústerSnapshotAttributes de la API.

Para eliminar el permiso de uso compartido de una cuenta AWS, utilice la acción `ModifyDBSnapshotAttribute` o `ModifyDBClusterSnapshotAttribute` con `AttributeName` definido como `restore` y el ID de la cuenta que desea eliminar en el parámetro `ValuesToRemove`.

Copia de una instantánea de clúster de base de datos sin cifrar en otra cuenta

Use el siguiente procedimiento para copiar una instantánea de clúster de base de datos sin cifrar en otra cuenta de la misma Región de AWS.

1. En la cuenta de origen de la instantánea del clúster de base de datos, llame a `ModifyDBClusterSnapshotAttribute` especificando **`restore`** para el parámetro `AttributeName` y el ID de la cuenta de destino para el parámetro `ValuesToAdd`.

   La ejecución del siguiente ejemplo con la cuenta `987654321` permite que dos identificadores de cuenta AWS, `123451234512` y `123456789012`, restauren la instantánea de clúster de base de datos llamada `manual-snapshot1`.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. En la cuenta de destino, llame a `CopyDBClusterSnapshot` y use el parámetro `SourceDBClusterSnapshotIdentifier` para especificar el ARN de la instantánea del clúster de base de datos que se va a copiar, que debe incluir el ID de la cuenta de origen.

   La ejecución del siguiente ejemplo con la cuenta `123451234512` copia la instantánea del clúster de base de datos `aurora-cluster1-snapshot-20130805` de la cuenta `987654321` y crea una instantánea de clúster de base de datos llamado `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copia de una instantánea de clúster de base de datos cifrada en otra cuenta

Use el siguiente procedimiento para copiar una instantánea de clúster de base de datos cifrada en otra cuenta de la misma Región de AWS.

1. En la cuenta de origen de la instantánea del clúster de base de datos, llame a `ModifyDBClusterSnapshotAttribute` especificando **`restore`** para el parámetro `AttributeName` y el ID de la cuenta de destino para el parámetro `ValuesToAdd`.

   La ejecución del siguiente ejemplo con la cuenta `987654321` permite que dos identificadores de cuenta AWS, `123451234512` y `123456789012`, restauren la instantánea de clúster de base de datos llamada `manual-snapshot1`.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
```

```
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. En la cuenta de origen de la instantánea del clúster de base de datos, cree una clave de KMS personalizada en la misma Región de AWS que la instantánea del clúster de base de datos cifrada. Al crear la clave administrada por el cliente, le da acceso a ella a la Cuenta de AWS de destino. Para obtener más información, consulte Creación de una clave administrada por el cliente y concesión de acceso a ella.
3. Copie y comparta la instantánea con la Cuenta de AWS de destino. Para obtener más información, consulte Copia y compartición de la instantánea desde la cuenta de origen.
4. En la cuenta de destino, llame a `CopyDBClusterSnapshot` y use el parámetro `SourceDBClusterSnapshotIdentifier` para especificar el ARN de la instantánea del clúster de base de datos que se va a copiar, que debe incluir el ID de la cuenta de origen.

   La ejecución del siguiente ejemplo con la cuenta `123451234512` copia la instantánea del clúster de base de datos `aurora-cluster1-snapshot-20130805` de la cuenta `987654321` y crea una instantánea de clúster de base de datos llamado `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
```

```
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Compartir una instantánea de clúster de base de datos

Al utilizar Amazon RDS, puede compartir una instantánea manual de clúster de base de datos de las siguientes maneras:

- Al compartir una instantánea manual de clúster de base de datos, ya sea cifrada o no cifrada, habilita a las cuentas autorizadas de AWS a copiar la instantánea.
- Si se comparte una instantánea manual de un clúster de base de datos, ya sea cifrada o sin cifrar, las cuentas autorizadas de AWS podrán restaurar directamente un clúster de base de datos a partir de la instantánea en lugar de hacer una copia de ella y restaurarla.

> **Note**
>
> Para compartir una instantánea automatizada de clúster de base de datos, cree una instantánea manual de clúster de base de datos al copiar la instantánea automatizada y, a continuación, comparta esa copia. Este proceso también se aplica a los recursos generados por Backup de AWS.

Para obtener más información acerca de la copia de instantáneas, consulte Copia de una instantánea de clúster de base de datos. Para obtener más información sobre cómo restaurar una instancia de base de datos desde una instantánea de clúster de base de datos, consulte Restauración de una instantánea de clúster de base de datos.

Para obtener más información acerca de cómo restaurar un clúster de base de datos a partir de una instantánea de clúster de base de datos, consulte Información general de copias de seguridad y restauración de un clúster de base de datos Aurora.

Puede compartir una instantánea manual con otras 20 Cuentas de AWS como máximo.

Cuando se comparten instantáneas manuales con otras Cuentas de AWS, se aplican las restricciones siguientes:

- Cuando se restaura un clúster de base de datos a partir de una instantánea compartida mediante la AWS Command Line Interface (AWS CLI) o la API de Amazon RDS, se debe especificar el nombre de recurso de Amazon (ARN) de la instantánea compartida como identificador de instantánea.

Aprenda a compartir instantáneas, instantáneas públicas e instantáneas cifradas en las siguientes secciones. También descubrirá cómo detener el uso compartido de instantáneas.

Temas

- Uso compartido de una instantánea
- Uso compartido de instantáneas públicas
- Cómo compartir instantáneas cifradas
- Cancelación del uso compartido de instantáneas

Uso compartido de una instantánea

Puede compartir una instantánea de clúster de base de datos usando la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Con la consola de Amazon RDS, puede compartir una instantánea manual de un clúster de base de datos con un máximo de 20 Cuentas de AWS. También puede utilizar la consola para dejar de compartir una instantánea manual con una o varias cuentas.

Para compartir una instantánea manual de clúster de base de datos mediante la consola de Amazon RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Seleccione la instantánea manual que desea compartir.
4. En Actions) (Acciones), elija Share Snapshot (Compartir instantánea).
5. Elija una de las siguientes opciones para DB Snapshot Visibility (Visibilidad de instantánea de base de datos).

    - Si el origen está sin cifrar, elija Público para permitir que todas las Cuentas de AWS restauren un clúster de base de datos a partir de la instantánea de clúster de base de datos manual, o elija Privado para permitir que únicamente las Cuentas de AWS que especifique restauren un clúster de base de datos a partir de una instantánea de clúster de base de datos manual.

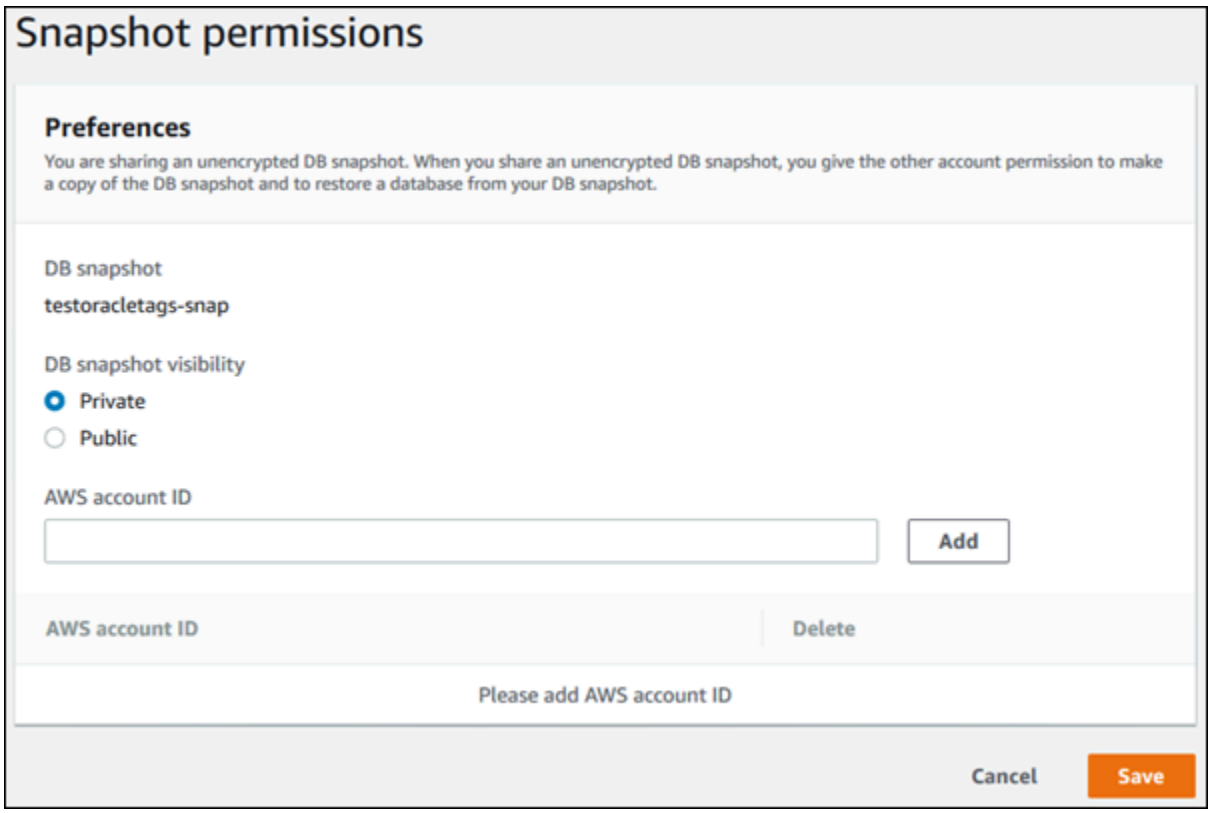
> **Warning**
>
> Si establece Visibilidad de instantánea de base de datos como Pública, todas las Cuentas de AWS pueden restaurar un clúster de base de datos a partir de una instantánea de clúster de base de datos manual y tener acceso a sus datos. No comparta como Public (Pública) ninguna instantánea de clúster de base de datos manual que contenga información confidencial.
> Para obtener más información, consulte Uso compartido de instantáneas públicas.

- Si el original está cifrado, DB Snapshot Visibility (Visibilidad de instantánea de base de datos) se establece en Private (Privada), ya que las instantáneas cifradas no se pueden compartir como públicas.

> **Note**
>
> Las instantáneas que se hayan cifrado con la AWS KMS key predeterminada no se pueden compartir. Para obtener información acerca de cómo solucionar este problema, consulte Cómo compartir instantáneas cifradas.

6. Para ID de cuenta de AWS, escriba el identificador de Cuenta de AWS para una cuenta a la que desea permitir restaurar un clúster de instancia de base de datos desde su instantánea manual y, luego, elija Agregar. Repita esta acción para incluir identificadores de Cuenta de AWS adicionales, hasta un máximo de 20 Cuentas de AWS.

   Si comete un error al añadir un identificador de Cuenta de AWS a la lista de cuentas permitidas, puede eliminarlo de la lista seleccionando Eliminar a la derecha del identificador incorrecto de la Cuenta de AWS.



7. Después de añadir los identificadores de todas las Cuentas de AWS a las que desea permitir la restauración de la instantánea manual, elija Guardar para guardar los cambios.

AWS CLI

Para compartir una instantánea de clúster de base de datos, use el comando `aws rds modify-db-cluster-snapshot-attribute`. Use el parámetro `--values-to-add` para añadir la lista de los ID de Cuentas de AWS que tienen autorización para restaurar la instantánea manual.

Example de compartir una instantánea con una sola cuenta

El siguiente ejemplo habilita el identificador de Cuenta de AWS `123456789012` para restaurar la instantánea de clúster de base de datos denominada `cluster-3-snapshot`.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier cluster-3-snapshot \
--attribute-name restore \
--values-to-add 123456789012
```

En:Windows

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier cluster-3-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example de compartir una instantánea con varias cuentas

El siguiente ejemplo habilita dos identificadores de Cuenta de AWS, `111122223333` y `444455556666`, para restaurar la instantánea del clúster de base de datos denominada `manual-cluster-snapshot1`.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

En:Windows

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Al utilizar el símbolo del sistema de Windows, debe aplicar escape con comillas dobles (") en código JSON al ponerlas como prefijo con una barra invertida (\).

Para enumerar las Cuentas de AWS habilitadas para restaurar una instantánea, utilice el comando `describe-db-cluster-snapshot-attributes` de la AWS CLI.

API de RDS

También puede compartir una instantánea manual de clúster de base de datos con otras Cuentas de AWS mediante la API de Amazon RDS. Para ello, llame a la operación `ModifyDBClusterSnapshotAttribute`. Especifique `restore` en `AttributeName` y utilice

el parámetro `ValuesToAdd` para añadir la lista de los ID de las Cuentas de AWS que tienen autorización para restaurar la instantánea manual.

Para hacer que una instantánea manual sea pública y puedan restaurarla todas las Cuentas de AWS, utilice el valor `all`. Sin embargo, tenga cuidado de no añadir el valor `all` para las instantáneas manuales que contienen información confidencial que no desea que esté disponible para todas las Cuentas de AWS. Además, tampoco especifique `all` para las instantáneas cifradas, ya que dichas instantáneas no pueden hacerse públicas.

Para ver una lista de todas las Cuentas de AWS que tienen permiso para restaurar una instantánea, utilice la operación `DescribeDBClusterSnapshotAttributes` de la API.

Uso compartido de instantáneas públicas

Puede compartir una instantánea manual sin cifrar como pública, lo que hace que esté disponible para todas las Cuentas de AWS. Al compartir una instantánea como pública, asegúrese de que no contiene información privada.

Cuando una instantánea se comparte públicamente, da todos los permisos de Cuentas de AWS tanto para copiar la instantánea como para crear los clústeres de base de datos de ella.

No se le facturará el almacenamiento de copia de seguridad de instantáneas públicas propiedad de otras cuentas. Solo se le facturan las instantáneas de su propiedad.

Si copia una instantánea pública, es el propietario de la copia. Se le facturará el almacenamiento de copia de seguridad de su copia instantánea. Si crea un clúster de base de datos desde una instantánea pública, se le facturará ese clúster de base de datos. Para obtener información acerca de los precios de Amazon Aurora, consulte la página de precios de Aurora.

Solo puede eliminar las instantáneas públicas de su propiedad. Para eliminar una instantánea compartida o pública, debe iniciar sesión en la Cuenta de AWS propietaria de la instantánea.

Visualización de instantáneas públicas propiedad de otras Cuentas de AWS

Puede ver instantáneas públicas propiedad de otras cuentas en una Región de AWS particular en la pestaña Público de la página Instantáneas de la consola de Amazon RDS. Sus instantáneas (las que pertenecen a su cuenta) no aparecen en esta pestaña.

Para ver instantáneas públicas

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, elija Instantáneas.
3. Seleccione la pestaña Public (Público).

   Aparecen las instantáneas públicas. Puede ver qué cuenta posee una instantánea pública en la columna Owner (Propietario).

   > **Note**
   >
   > Es posible que tenga que modificar las preferencias de la página, seleccionando el icono de engranaje en la parte superior derecha de la lista Public snapshots (Instantáneas públicas), para ver esta columna.

Consulta de sus propias instantáneas públicas

Puede utilizar el siguiente comando de la AWS CLI (solo para Unix) a fin de buscar las instantáneas públicas de su Cuenta de AWS en una Región de AWS concreta.

```
aws rds describe-db-cluster-snapshots --snapshot-type public --include-public |
 grep account_number
```

El resultado devuelto es similar al siguiente ejemplo si tiene instantáneas públicas.

```
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot1",
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot2",
```

Uso compartido de instantáneas públicas de versiones obsoletas del motor de base de datos

No se admite la restauración ni la copia de instantáneas públicas a partir de versiones obsoletas del motor de base de datos. Para poder restaurar o copiar su instantánea pública existente no compatible, realice los siguientes pasos:

1. Marque la instantánea como privada.
2. Restaurare la instantánea.
3. Actualice el clúster de base de datos restaurado a una versión del motor compatible.

4. Cree una instantánea.
5. Vuelva a compartir la instantánea públicamente.

Cómo compartir instantáneas cifradas

Puede compartir instantáneas de clúster de base de datos que se han cifrado "en reposo" utilizando el algoritmo de cifrado AES-256, como se describe en Cifrado de recursos de Amazon Aurora.

Cuando se comparten instantáneas cifradas, se aplican las siguientes restricciones:

- No se pueden compartir instantáneas cifradas como públicas.
- No se puede compartir una instantánea que se ha cifrado utilizando la clave de KMS predeterminada de la Cuenta de AWS que compartió la instantánea.

  Para obtener más información sobre la administración de claves de AWS KMS para Amazon RDS, consulte Administración de AWS KMS key.

Para solucionar el problema de la clave de KMS predeterminada, realice las siguientes tareas:

1. Creación de una clave administrada por el cliente y concesión de acceso a ella.
2. Copia y compartición de la instantánea desde la cuenta de origen.
3. Copia de la instantánea compartida en la cuenta de destino.

Creación de una clave administrada por el cliente y concesión de acceso a ella

En primer lugar, debe crear una clave KMS personalizada en la misma Región de AWS que la instantánea del clúster de la base de datos cifrada. Al crear la clave administrada por el cliente, le da acceso a ella a otra Cuenta de AWS.

Para crear una clave administrada por el cliente y dar acceso a ella

1. Inicie sesión en la Consola de administración de AWS desde la Cuenta de AWS de origen.
2. Abra la consola de AWS KMS en https://console.aws.amazon.com/kms.
3. Para cambiar la Región de AWS, utilice el Selector de regiones ubicado en la esquina superior derecha de la página.
4. En el panel de navegación, elija Claves administradas por el cliente.

5. Elija Create key.
6. En la página Configurar clave:
    a. En Tipo de clave, seleccione Simétrica.
    b. En Uso de claves, seleccione Cifrar y descifrar.
    c. Expanda Advanced options (Opciones avanzadas).
    d. En Origen del material de claves, seleccione Externo.
    e. En Regionalidad, seleccione Clave de una sola región.
    f. Elija Siguiente.
7. En la página Agregar etiquetas:
    a. Para Alias, introduzca un nombre que mostrar para su clave KMS, por ejemplo **`share-snapshot`**.
    b. (Opcional) Introduzca una descripción de su clave KMS.
    c. (Opcional) Agregue etiquetas a su clave KMS.
    d. Elija Siguiente.
8. En la página Definir permisos de administración de claves, elija Siguiente.
9. En la página Definir permisos de uso de claves:
    a. En Otras Cuentas de AWS, seleccione Agregar otra Cuenta de AWS.
    b. Introduzca el ID de la Cuenta de AWS a la que desee conceder acceso.

        Puede conceder acceso a varias Cuentas de AWS.
    c. Elija Siguiente.
10. Revise su clave KMS y, a continuación, seleccione Finalizar.

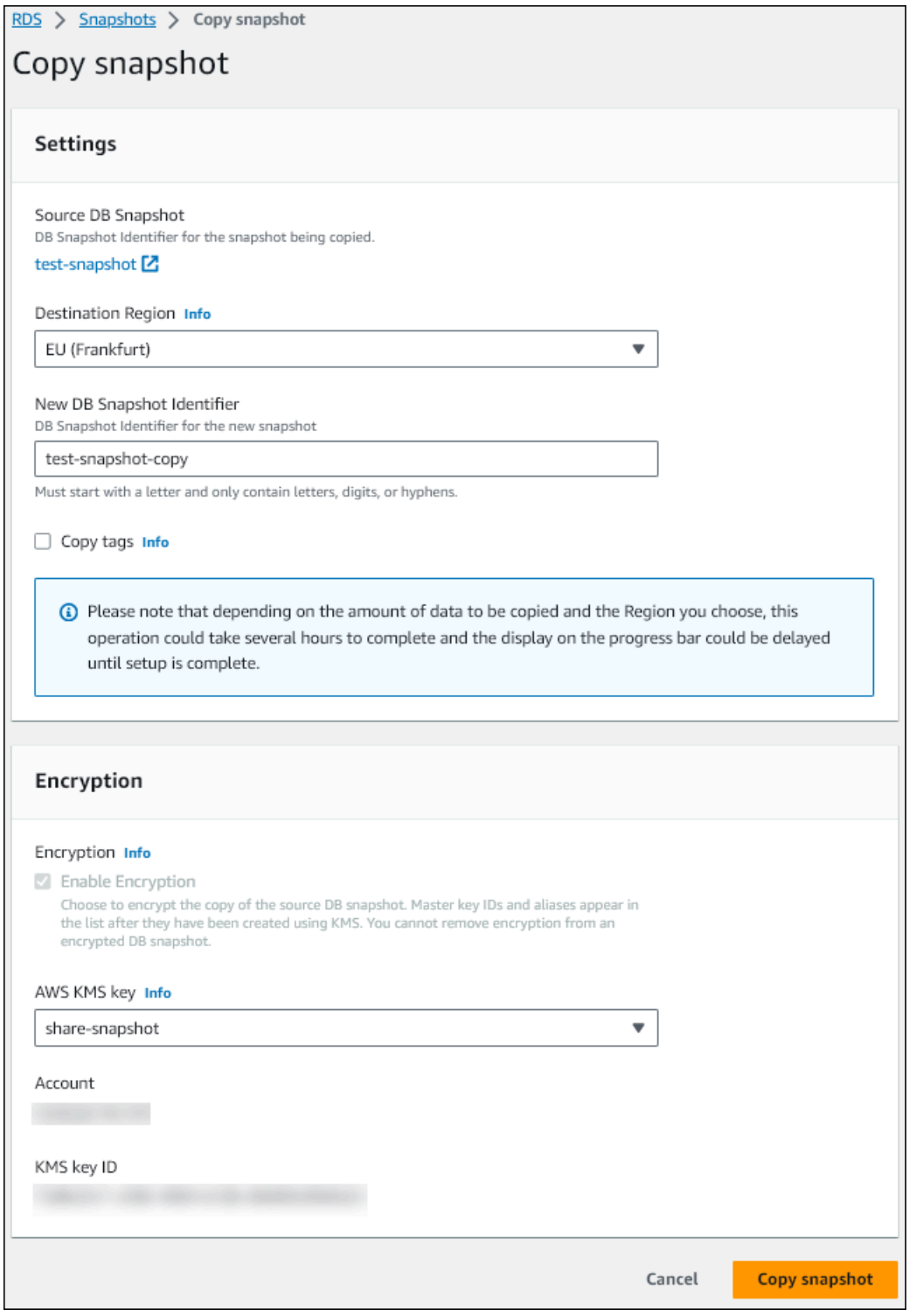
Copia y compartición de la instantánea desde la cuenta de origen

A continuación, debe copiar la instantánea del clúster de base de datos de origen en una nueva instantánea mediante la clave administrada por el cliente. A continuación, la debe compartir con la Cuenta de AWS de destino.

Para copiar y compartir la instantánea

1. Inicie sesión en la Consola de administración de AWS desde la Cuenta de AWS de origen.

2. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
3. En el panel de navegación, elija Instantáneas.
4. Seleccione la instantánea del clúster de base de datos que quiera copiar.
5. En Actions (Acciones), elija Copy snapshot (Copiar instantánea).
6. En la página Copiar instantánea:
    a. Para Región de destino, elija la Región de AWS en la que creó la clave administrada por el cliente en el procedimiento anterior.
    b. Introduzca el nombre de la copia de la instantánea del clúster de base de datos en New DB Snapshot Identifier (Nuevo identificador de instantánea de base de datos).
    c. Para AWS KMS key, elija la clave administrada por el cliente que ha creado.



d. Elija Copy Snapshot (Copiar instantánea).

7. Cuando la copia de la instantánea esté disponible, selecciónela.
8. En Actions) (Acciones), elija Share Snapshot (Compartir instantánea).
9. En la página Permisos de la instantánea:

a. Introduzca el ID de la Cuenta de AWS con la que vaya a compartir la copia de la instantánea y, a continuación, seleccione Agregar.
b. Seleccione Guardar.

La instantánea ya se ha compartido.

Copia de la instantánea compartida en la cuenta de destino

Ahora puede copiar la instantánea compartida en la Cuenta de AWS de destino.

Para copiar la instantánea compartida

1. Inicie sesión en la Consola de administración de AWS desde la Cuenta de AWS de destino.
2. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
3. En el panel de navegación, elija Instantáneas.
4. Seleccione la pestaña Compartido conmigo.
5. Seleccione la instantánea compartida.
6. En Actions (Acciones), elija Copy snapshot (Copiar instantánea).
7. Elija la configuración para copiar la instantánea como en el procedimiento anterior, pero utilice una AWS KMS key que pertenezca a la cuenta de destino.

   Elija Copy Snapshot (Copiar instantánea).

Cancelación del uso compartido de instantáneas

Para dejar de compartir una instantánea de un clúster de base de datos, debe eliminar el permiso de la Cuenta de AWS de destino.

Consola

Para dejar de compartir una instantánea manual de clúster de base de datos con una Cuenta de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).

3. Seleccione la instantánea manual que desea dejar de compartir.
4. Elija Actions (Acciones) y, a continuación, Share Snapshot (Compartir instantánea).
5. Para eliminar el permiso de una Cuenta de AWS, elija Eliminar para el identificador de cuenta de AWS correspondiente a esa cuenta en la lista de cuentas autorizadas.
6. Elija Guardar para guardar los cambios.

CLI

Para quitar un identificador de Cuenta de AWS de la lista, use el parámetro `--values-to-remove`.

Example de detener el uso compartido de instantáneas

En el siguiente ejemplo se impide que el ID 444455556666 de Cuenta de AWS se restaure desde la instantánea.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

En:Windows

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

API de RDS

Para eliminar el permiso de uso compartido de una Cuenta de AWS, utilice la operación `ModifyDBClusterSnapshotAttribute` con `AttributeName` establecido en `restore` y el parámetro `ValuesToRemove`. Para marcar una instantánea manual como privada, elimine el valor `all` de la lista de valores del atributo `restore`.

Exportación de datos del clúster de base de datos a Amazon S3

Puede exportar datos desde un clúster de base de datos de Amazon Aurora activo a un bucket de Amazon S3. El proceso de exportación se ejecuta en segundo plano y no afecta al rendimiento del clúster de la base de datos activa.

De forma predeterminada, se exportan todos los datos del clúster de base de datos. Sin embargo, también puede optar por exportar conjuntos específicos de bases de datos, esquemas o tablas.

Amazon Aurora clona el clúster de base de datos, extrae los datos del clon y los almacena en un bucket de Amazon S3. Los datos se almacenan en formato Apache Parquet comprimido y consistente. Los archivos individuales de Parquet suelen tener un tamaño de entre 1 y 10 MB.

El rendimiento más rápido que se puede obtener al exportar datos de instantáneas para las versiones 2 y 3 de Aurora MySQL no se aplica a la exportación de datos de clústeres de bases de datos. Para obtener más información, consulte Exportación de datos de instantánea del clúster de bases de datos a Amazon S3.

Se le cobrará por exportar todo el clúster de base de datos, ya exporte todos los datos o parte de ellos. Para obtener más información, consulte la Página de precios de Amazon Aurora.

Después de exportar los datos, puede analizar los datos exportados directamente con herramientas como Amazon Athena o Amazon Redshift Spectrum. Para obtener más información sobre cómo utilizar Athena para leer los datos de Parquet, consulte Parquet SerDe en Guía del usuario de Amazon Athena. Para obtener más información sobre cómo utilizar Redshift Spectrum para leer datos de Parquet, vea Uso de COPY con formatos de datos de columnas en la Guía para desarrolladores de bases de datos Amazon Redshift.

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos y entre Regiones de AWS. Para obtener más información sobre la disponibilidad en versiones y regiones de la exportación de datos de clústeres de base de datos a S3, consulte Regiones y motores de base de datos Aurora admitidos para exportar datos del clúster a Amazon S3.

Utilice el siguiente proceso para exportar datos de clústeres de base de datos a un bucket de Amazon S3. Para obtener más detalles, consulte las siguientes secciones.

Información general de la exportación de datos de un clúster de base de datos

1. Identifique el clúster de base de datos cuyos datos desea exportar.
2. Configure el acceso al bucket de Amazon S3.

Un bucket es un contenedor de objetos o archivos de Amazon S3. Para proporcionar la información necesario para obtener acceso a un bucket, siga los siguientes pasos:

a. Identifique el bucket de S3 al que se van a exportar los datos del clúster de base de datos. El bucket de S3; debe estar en la misma región de AWS que el clúster de base de datos. Para obtener más información, consulte Identificación del bucket de Amazon S3 para exportación.

b. Cree un rol de AWS Identity and Access Management (IAM) que conceda a la tarea de exportación del clúster de base de datos acceso al bucket de S3. Para obtener más información, consulte Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM.

3. Cree una AWS KMS key de cifrado simétrica para el cifrado del lado del servidor. La tarea de exportación del clúster utiliza la clave KMS para configurar el cifrado del lado del servidor de AWS KMS al escribir los datos de exportación en S3.

   La política de clave KMS debe incluir los permisos `kms:CreateGrant` y `kms:DescribeKey`. Para obtener más información acerca del uso de claves KMS en Amazon Aurora, consulte Administración de AWS KMS key.

   Además, si tiene una instrucción deny en la política de claves KMS, asegúrese de excluir explícitamente la entidad principal del servicio de AWS `export.rds.amazonaws.com`.

   Puede utilizar una clave de KMS en su cuenta de AWS o puede utilizar una clave KMS en diversas cuentas. Para obtener más información, consulte Uso de un AWS KMS key en diversas cuentas.

4. Exporte el clúster de base de datos a Amazon S3 mediante la consola o el comando `start-export-task` de la CLI. Para obtener más información, consulte Creación de tareas de exportación del clúster de base de datos.

5. Para obtener acceso a los datos exportados al bucket de Amazon S3, consulte Carga, descarga y administración de objetos en la Guía del usuario de Amazon Simple Storage Service.

En las siguientes secciones, descubrirá el proceso de configuración, exportación, monitorización, cancelación y resolución de problemas para tareas de exportación de clústeres de base de datos.

Temas

- Observaciones sobre la exportación de un clúster de base de datos

- Configuración del acceso a un bucket de Amazon S3
- Creación de tareas de exportación del clúster de base de datos
- Supervisión de tareas de exportación del clúster de base de datos
- Cancelación de una tarea de exportación de un clúster de base de datos
- Resolución de problemas en las exportaciones de clústeres de base de datos

Observaciones sobre la exportación de un clúster de base de datos

Utilice las siguientes secciones para obtener información sobre las limitaciones, las convenciones de nomenclatura de archivos y la conversión y el almacenamiento de datos al exportar datos de clústeres de bases de datos a Amazon S3.

Temas

- Limitaciones
- Convención de nomenclatura de archivos
- Formato de almacenamiento y conversión de datos

Limitaciones

La exportación de datos de clústeres de base de datos a Amazon S3 tiene las siguientes limitaciones:

- No puede ejecutar varias tareas de exportación para el mismo clúster de base de datos simultáneamente. Esto es cierto para las exportaciones completas y parciales.
- Puede tener hasta cinco tareas de exportación de instantáneas de base de datos en curso por Cuenta de AWS.
- Los clústeres de base de datos de Aurora Serverless v1 no admiten la exportación a S3.
- Aurora MySQL y Aurora PostgreSQL admiten exportaciones a S3 solo para el modo de motor aprovisionado.
- Las exportaciones a S3 no admiten prefijos S3 que contengan dos puntos (:).
- Los siguientes caracteres en la ruta del archivo S3 se convierten en guiones bajos (_) durante la exportación:

```
\ ` " (space)
```

- Si una base de datos, esquema o tabla tiene caracteres en su nombre distintos del siguiente, no se admite la exportación parcial. Sin embargo, puede exportar todo el clúster de base de datos.
  - Letras latinas (A–Z)
  - Dígitos (0–9)
  - Símbolo de dólar ($)
  - Guion bajo (_)
- No se admiten espacios ( ) ni determinados caracteres en los nombres de columna de las tablas de bases de datos. Las tablas con los siguientes caracteres en los nombres de columna se omiten durante la exportación:

  ```
  , ; { } ( ) \n \t = (space)
  ```

- Las tablas con barras diagonales (/) en el nombre se omiten durante la exportación.
- Las tablas temporales y no registradas de Aurora PostgreSQL se omiten durante la exportación.
- Si los datos contienen un objeto grande, como un BLOB o CLOB, cercano o superior a 500 MB, se producirá un error en la exportación.
- Si una tabla contiene una fila grande cercana o superior a 2 GB, la tabla se omite durante la exportación.
- Para exportaciones parciales, la lista `ExportOnly` tiene un tamaño máximo de 200 KB.
- Es muy recomendable que utilice un nombre exclusivo para cada tarea de exportación. Si no utiliza un nombre de tarea exclusivo, es posible que aparezca el siguiente mensaje de error como el que sigue:

  exportTaskAlreadyExistsFault: Se ha producido un error (exportTaskAlReadyExists) al llamar a la operación StartExportTask: la tarea de exportación con ID *xxxxx* ya existe.
- Dado que es posible que se omitan algunas tablas, le recomendamos que verifique los recuentos de filas y tablas de los datos después de la exportación.

Convención de nomenclatura de archivos

Los datos exportados para tablas específicas se almacenan en el formato *base_prefix*/*files*, donde el prefijo base es el siguiente:

```
export_identifier/database_name/schema_name.table_name/
```

Por ejemplo:

```
export-1234567890123-459/rdststcluster/mycluster.DataInsert_7ADB5D19965123A2/
```

Los archivos de salida utilizan la siguiente convención de nomenclatura, donde *partition_index* es alfanumérico:

```
partition_index/part-00000-random_uuid.format-based_extension
```

Por ejemplo:

```
1/part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
    a/part-00000-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
```

La convención de nomenclatura de archivos está sujeta a cambios. Por lo tanto, cuando lea tablas de destino, recomendamos que lea todo lo que hay dentro del prefijo base de la tabla.

Formato de almacenamiento y conversión de datos

Cuando exporta un clúster de base de datos a un bucket de Amazon S3, Amazon Aurora convierte, exporta y almacena los datos con el formato Parquet. Para obtener más información, consulte Conversión de datos al exportar a un bucket de Amazon S3.

Configuración del acceso a un bucket de Amazon S3

Una vez identificado el bucket de Amazon S3, dé a la tarea de exportación del clúster de base de datos permiso para acceder a él.

Temas

- Identificación del bucket de Amazon S3 para exportación
- Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM
- Uso de un bucket de Amazon S3 en diversas cuentas

Identificación del bucket de Amazon S3 para exportación

Identifique el bucket de Amazon S3 al que se exportará el clúster de base de datos. Utilice un bucket de S3 ya existente, o bien cree un bucket S3 nuevo.

> **Note**
>
> El bucket de S3; debe estar en la misma región de AWS que el clúster de base de datos.

Para obtener más información acerca de cómo trabajar con buckets de Amazon S3, consulte lo siguiente en Guía del usuario de Amazon Simple Storage Service:

- Cómo se consultan las propiedades de un bucket de S?
- Cómo puedo habilitar el cifrado predeterminado para un bucket de Amazon S?
- Cómo se puede crear un bucket de S?

Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM

Antes de exportar datos de clústeres de bases de datos a Amazon S3, conceda a las tareas de exportación permiso de acceso de escritura al bucket de Amazon S3.

Para conceder este permiso, cree una política de IAM que proporcione acceso al bucket y cree un rol de IAM y adjunte la política al rol. Más adelante, puede asignar el rol de IAM a la tarea de exportación del clúster de base de datos.

> **Important**
>
> Si prevé utilizar la Consola de administración de AWS para exportar el clúster de base de datos, puede elegir crear la política de IAM y el rol automáticamente al exportar el clúster de base de datos. Para obtener instrucciones, consulte Creación de tareas de exportación del clúster de base de datos.

Para dar a las tareas acceso a Amazon S3

1. Cree una política de IAM. Esta política proporciona los permisos de bucket y objeto que permiten a la tarea de exportación de clústeres de base de datos obtener acceso a Amazon S3.

   En la política, incluya las siguientes acciones obligatorias para permitir transferir archivos desde Amazon Aurora a un bucket de S3:

   - `s3:PutObject*`

- `s3:GetObject*`
- `s3:ListBucket`
- `s3:DeleteObject*`
- `s3:GetBucketLocation`

En la política, incluya los siguientes recursos para identificar el bucket de S3 y los objetos incluidos en él. En la siguiente lista de recursos se muestra el formato de nombre de recurso de Amazon (ARN) para obtener acceso a Amazon S3.

- `arn:aws:s3:::amzn-s3-demo-bucket`
- `arn:aws:s3:::amzn-s3-demo-bucket/*`

Para obtener más información sobre cómo crear una política de IAM para Amazon Aurora, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM. Consulte también el Tutorial: Crear y asociar su primera política administrada por el cliente en la Guía del usuario de IAM.

El siguiente comando de la AWS CLI crea una política de IAM denominada `ExportPolicy` con estas opciones. Otorga acceso a un bucket denominado *amzn-s3-demo-bucket*.

> **Note**
>
> Después de crear la política, apunte el ARN de esta. Cuando asocia la política a un rol de IAM, necesita el ARN para realizar un paso posterior.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
```

```
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Cree un rol de IAM que Aurora pueda asumir en su nombre para acceder a sus buckets de Amazon S3. Para obtener más información, vea Crear un rol para delegar permisos a un IAM usuario en Guía del usuario de IAM.

   En el siguiente ejemplo se muestra cómo se usa el comando de la AWS CLI para crear un rol denominado `rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Asocie la política de IAM que creó al rol de IAM creado.

   El siguiente comando de la AWS CLI asocia la política creada anteriormente al rol denominado `rds-s3-export-role`. Sustituya *`your-policy-arn`* por el ARN de la política que ha apuntado en el paso anterior.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Uso de un bucket de Amazon S3 en diversas cuentas

Puede utilizar buckets de S3 en cuentas de AWS. Para obtener más información, consulte Uso de un bucket de Amazon S3 en diversas cuentas.

Creación de tareas de exportación del clúster de base de datos

Cree tareas de exportación para exportar datos desde su clúster de base de datos de Aurora a un bucket de Amazon S3. Puede tener hasta cinco tareas de exportación de clústeres de base de datos en curso por Cuenta de AWS.

> **Note**
>
> La exportación de datos de clústeres de base de datos puede tardar un tiempo en función del tipo y tamaño de la base de datos. La tarea de exportación primero clona y escala toda la base de datos antes de extraer los datos a Amazon S3. El progreso de la tarea durante esta fase se muestra como Starting (Iniciándose). Cuando la tarea cambia a exportar datos a S3, el progreso se muestra como In progress (En curso).
>
> El tiempo que tarda la exportación en completarse depende de los datos almacenados en la base de datos. Por ejemplo, las tablas con columnas de índice o claves primarias numéricas bien distribuidas se exportarán más rápido. Las tablas que no contienen una columna adecuada para la partición y las tablas con un solo índice en una columna basada en cadenas tardarán más tiempo porque la exportación utiliza un proceso más lento que tiene un único subproceso.

Puede exportar datos de clúster esde base de datos a Amazon S3 mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Si utiliza una función de Lambda para exportar los datos de clústeres de base de datos, añada la acción `kms:DescribeKey` a la política de la función de Lambda. Para obtener más información, consulte Permisos de AWS Lambda.

Consola

La opción de la consola Export to Amazon S3 (Exportar a Amazon S3) solo aparece para los clústeres de base de datos que se pueden exportar a Amazon S3. Es posible que un clúster de base de datos no esté disponible para la exportación debido a las siguientes razones:

- El motor de base de datos no es compatible con la exportación de S3.

- La versión del clúster de base de datoss no es compatible con la exportación de S3.
- La exportación de S3 no se admite en la región de AWS donde se creó el clúster de base de datos.

Para exportar datos del clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos cuyos datos desea exportar.
4. En Actions (Acciones), seleccione Export to Amazon S3 (Exportar a Amazon S3).

   Se visualizará la ventana Export to Amazon S3 (Exportar a Amazon S3).
5. En Export Identifier (Identificador de exportación), escriba un nombre para identificar la tarea de exportación. Este valor también se utiliza para el nombre del archivo creado en el bucket de S3.
6. Elija los datos que desea exportar:

   - Seleccione All (Todo) para exportar todos los datos del clúster de base de datos.
   - Seleccione Partial (Parcial) para exportar partes específicas del clúster de base de datos. Para identificar qué partes del clúster exportar, introduzca una o más bases de datos, esquemas o tablas para Identifiers (Identificadores) separadas por espacios.

     Use el siguiente formato:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.scheman]
     [.tablen]
     ```

     Por ejemplo:

     ```
     mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
      mydatabase2.myschema2.mytable2
     ```
7. Para el S3 bucket (Bucket de S3), elija el bucket al que desee realizar la exportación.

   Para asignar los datos exportados a la ruta de una carpeta en el bucket de S3, escriba la ruta opcional para el S3 prefix (Prefijo de S3).
8. Para el rol de IAM, elija un rol que le conceda acceso de escritura al bucket de S3 elegido o cree un nuevo rol.

- Si ha creado un rol siguiendo los pasos indicados en Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM, elija dicho rol.
- Si no ha creado un rol que le conceda acceso de escritura al bucket de S3 elegido, elija Create a new role (Crear un nuevo rol) para crear el rol automáticamente. A continuación, escriba un nombre para el rol en el IAM role name (Nombre del rol de IAM).

9. En KMS key (Clave KMS), introduzca el ARN de la clave que debe utilizarse para cifrar los datos exportados.
10. Elija Export to Amazon S3 (Exportar a Amazon S3).

AWS CLI

Para exportar un clúster de base de datos a Amazon S3 mediante la AWS CLI, ejecute el comando start-export-task con las siguientes opciones obligatorias:

- `--export-task-identifier`
- `--source-arn`: el nombre de recurso de Amazon (ARN) del clúster de base de datos.
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

En los siguientes ejemplos, la tarea de exportación se denomina *`my-clúster-export`* y exporta los datos a un bucket de S3 denominado *`amzn-s3-demo-destination-bucket`*.

Example

Para Linux, macOS o:Unix

```
aws rds start-export-task \
    --export-task-identifier my-cluster-export \
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

En:Windows

```
aws rds start-export-task ^
    --export-task-identifier my-DB-cluster-export ^
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

A continuación, se muestra un resultado de ejemplo.

```
{
    "ExportTaskIdentifier": "my-cluster-export",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "IamRoleArn": "arn:aws:iam:123456789012:role/ExportTest",
    "KmsKeyId": "my-key",
    "Status": "STARTING",
    "PercentProgress": 0,
    "TotalExtractedDataInGB": 0,
}
```

Para proporcionar la ruta de una carpeta del bucket S3 para la exportación del clúster de base de datos, incluya la opción `--s3-prefix` en el comando start-export-task.

API de RDS

Para exportar un clúster de base de datos a Amazon S3 con la API de Amazon RDS, ejecute la operación StartExportTask con los siguientes parámetros obligatorios:

- `ExportTaskIdentifier`
- `SourceArn`: el ARN del clúster de base de datos.
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Supervisión de tareas de exportación del clúster de base de datos

Puede supervisar las exportaciones de clústeres de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para supervisar las exportaciones del clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Exports in Amazon S3 (Exportaciones en Amazon S3).

   Las exportaciones de clústeres de base de datos se indican en la columna Source type (Tipo de fuente). El estado de exportación se muestra en la columna Status (Estado).
3. Para ver información detallada acerca de la exportación de un clúster de base de datos específico, elija la tarea de exportación.

AWS CLI

Para supervisar exportaciones de clústeres de base de datos mediante la AWS CLI, ejecute el comando describe-export-tasks .

En el ejemplo siguiente, se indica cómo mostrar la información actual acerca de todas las exportaciones de clústeres de base de datos.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2022-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:parameter-groups-
test"
        },
```

```
{
            "Status": "COMPLETE",
            "TaskStartTime": "2022-10-31T20:58:06.998Z",
            "TaskEndTime": "2022-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2022-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket amzn-s3-demo-bucket2 isn't located in the
 current AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-30-06-45"
        }
    ]
}
```

Para mostrar información sobre una exportación de clústeres de base de datos específica, incluya la opción `--export-task-identifier` con el comando `describe-export-tasks`. Para filtrar la salida, incluya la opción `--Filters`. Para obtener más opciones, consulte el comando describe-export-tasks.

API de RDS

Para mostrar información sobre las exportaciones de clústeres de base de datos mediante la API de Amazon RDS, ejecute la operación DescribeExportTasks.

Para realizar un seguimiento del flujo de trabajo de exportación o para iniciar otro flujo de trabajo, puede suscribirse a temas de Amazon Simple Notification Service. Para obtener más información sobre Amazon SNS, consulte Uso de notificaciones de eventos de Amazon RDS.

Cancelación de una tarea de exportación de un clúster de base de datos

Puede cancelar una tarea de exportación de clústeres de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

> **Note**
>
> La cancelación de una tarea de exportación no elimina los datos exportados a Amazon S3. Para obtener información acerca de cómo eliminar los datos mediante la consola, consulte ¿Cómo se eliminan objetos de un bucket de S3? Para eliminar los datos mediante la CLI, ejecute el comando delete-object.

Consola

Para cancelar una tarea de exportación de un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Exports in Amazon S3 (Exportaciones en Amazon S3).

   Las exportaciones de clústeres de base de datos se indican en la columna Source type (Tipo de fuente). El estado de exportación se muestra en la columna Status (Estado).
3. Elija la tarea de exportación que desee cancelar.
4. Elija Cancel.
5. Seleccione Cancel export task (Cancelar tarea de exportación) en la página de confirmación.

AWS CLI

Para cancelar una tarea de exportación mediante la AWS CLI, ejecute el comando cancel-export-task. El comando requiere la opción `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my-export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my-export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:export-example-1"
}
```

API de RDS

Para cancelar una tarea de exportación mediante la API de Amazon RDS, ejecute la operación CancelExportTask con el parámetro `ExportTaskIdentifier`.

Resolución de problemas en las exportaciones de clústeres de base de datos

Utilice las siguientes secciones para solucionar los mensajes de error y los errores de permisos de PostgreSQL para las tareas de exportación de clústeres de bases de datos a Amazon S3.

Mensajes de error para tareas de exportación de Amazon S3

En la tabla siguiente se describen los mensajes que se devuelven cuando se producen errores en las tareas de exportación de Amazon S3.

Mensaje de error	Descripción
No se pudo encontrar el clúster de base de datos de origen ni acceder a él: [nombre del clúster]	El clúster de base de datos de origen no se puede clonar.

Mensaje de error	Descripción
Se ha producido un error interno desconocido.	La tarea no se pudo completar debido a un error, excepción o falla desconocidos.
Ocurrió un error interno desconocido al escribir los metadatos de la tarea de exportación en el bucket de S3 [nombre del bucket].	La tarea no se pudo completar debido a un error, excepción o falla desconocidos.
La exportación de RDS no pudo escribir los metadatos de la tarea de exportaci ón porque no puede asumir el rol de IAM [ARN de rol].	La tarea de exportación asume el rol de IAM para validar si está permitido escribir metadatos en el bucket de S3. Si la tarea no puede asumir su rol de IAM, muestra un error.
La exportación de RDS no pudo escribir los metadatos de la tarea de exportaci ón en el bucket de S3 [nombre del bucket] que utiliza el rol de IAM [ARN de rol] con la clave KMS [ID de clave]. Código de error: [código de error]	Faltan uno o más permisos, por lo que la tarea de exportación no puede acceder al bucket de S3. Este mensaje de error aparece cuando se recibe uno de los siguientes códigos de error: • `AWSSecurityTokenServiceException` con el código de error `AccessDenied` • `AmazonS3Exception` con el código de error `NoSuchBucket` , `AccessDenied` , `KMS.KMSInvalidStateException` , `403 Forbidden` , o `KMS.DisabledException` Estos códigos de error indican que la configuración del rol de IAM, el bucket de S3 o la clave KMS es incorrecta.
El rol de IAM [ARN de rol] no está autorizado para llamar a [acción de S3] en el bucket de S3 [nombre del bucket]. Revise sus permisos y vuelva a intentar la exportación.	La política de IAM está mal configurada. Falta el permiso para la acción específica de S3 en el bucket de S3, que provoca que falle la tarea de exportación.

Mensaje de error	Descripción
Error en la verificación de claves KMS Verifique las credenciales de la clave KMS e inténtelo de nuevo.	Error en la verificación de credenciales de la clave KMS.
Error en la verificación de credencia les de S3. Verifique los permisos de su bucket de S3 y de la política de IAM.	Error en la verificación de credenciales de S3.
El bucket de S3 [nombre del bucket] no es válido. O no se encuentra en la Región de AWS actual o no existe. Revise el nombre del bucket de S3 e intente hacer la exportación de nuevo.	El bucket de S3 no es válido.
El bucket de S3 [nombre del bucket] no se encuentra en la Región de AWS actual. Revise el nombre del bucket de S3 e intente hacer la exportación de nuevo.	El bucket de S3 está en la Región de AWS equivocad a.

Solución de problemas de errores de permisos de PostgreSQL

Al exportar bases de datos PostgreSQL a Amazon S3, es posible que vea un error `PERMISSIONS_DO_NOT_EXIST` que indica que se omitieron ciertas tablas. Esto suele deberse a que el superusuario, que se especifica al crear el clúster de base de datos, no tiene permisos para acceder a dichas tablas.

Para corregir este error, ejecute el siguiente comando:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Para obtener más información sobre los privilegios de superusuario, consulte Privilegios de la cuenta de usuario maestro.

Exportación de datos de instantánea del clúster de bases de datos a Amazon S3

Puede exportar datos de instantánea de clúster de bases de datos a un bucket de Amazon S3. El proceso de exportación se ejecuta en segundo plano y no afecta al rendimiento del clúster de la base de datos activa.

Al exportar una instantánea del clúster de base de datos, Amazon Aurora extrae los datos de la instantánea y los almacena en un bucket de Amazon S3. Puede exportar instantáneas manuales e instantáneas del sistema automatizadas. De forma predeterminada, se exportan todos los datos de la instantánea. Sin embargo, también puede optar por exportar conjuntos específicos de bases de datos, esquemas o tablas.

> **Note**
>
> La exportación de datos de una instantánea de clúster de base de datos requiere restaurar la instantánea. Los tiempos de restauración se ven afectados por varios factores, como la cantidad de tráfico de red que recibe una Región de AWS en relación con el ancho de banda disponible. Cuando se produce un aumento repentino del tráfico, es posible que los tiempos de finalización sean más largos de lo esperado.
> Una alternativa para reducir los tiempos de exportación a S3 para las bases de datos de Aurora es la exportación de clústeres de base de datos en tiempo real a S3. La exportación de clústeres de base de datos tiene tiempos de inicio más cortos que la exportación de instantáneas de base de datos, ya que no es necesario restaurar una instantánea. Para obtener más información, consulte Exportación de datos del clúster de base de datos a Amazon S3.

Los datos se almacenan en formato Apache Parquet comprimido y consistente. Los archivos individuales de Parquet suelen tener un tamaño de entre 1 y 10 MB.

Después de exportar los datos, puede analizar los datos exportados directamente con herramientas como Amazon Athena o Amazon Redshift Spectrum. Para obtener más información sobre cómo utilizar Athena para leer los datos de Parquet, consulte Parquet SerDe en Guía del usuario de Amazon Athena. Para obtener más información sobre cómo utilizar Redshift Spectrum para leer datos de Parquet, vea Uso de COPY con formatos de datos de columnas en la Guía para desarrolladores de bases de datos Amazon Redshift.

La disponibilidad de características varía según las versiones específicas de cada motor de base de datos y entre Regiones de AWS. Para obtener más información sobre la disponibilidad en versiones y regiones de la exportación de datos de instantáneas de clústeres de base de datos a S3, consulte Regiones y motores de base de datos Aurora admitidos para exportar datos de instantáneas a Amazon S3.

Utilice el siguiente proceso para exportar datos de instantáneas de base de datos a un bucket de Amazon S3. Para obtener más detalles, consulte las siguientes secciones.

Información general acerca de la exportación de datos de instantáneas

1. Identifique la instantánea que desee exportar.

   Utilice una instantánea automática o manual ya existente, o bien cree una instantánea manual de una instancia de base de datos.

2. Configure el acceso al bucket de Amazon S3.

   Un bucket es un contenedor de objetos o archivos de Amazon S3. Para proporcionar la información necesario para obtener acceso a un bucket, siga los siguientes pasos:

   a. Identifique el bucket de S3 al que se va a exportar la instantánea. El bucket de S3 debe estar en la misma región de AWS que la instantánea. Para obtener más información, consulte Identificación del bucket de Amazon S3 para exportación.

   b. Cree un rol de AWS Identity and Access Management (IAM) que conceda a la tarea de exportación de instantáneas acceso al bucket de S3. Para obtener más información, consulte Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM.

3. Cree una AWS KMS key de cifrado simétrica para el cifrado del lado del servidor. La tarea de exportación de instantáneas utiliza la clave de KMS para configurar el cifrado del lado del servidor de AWS KMS al escribir los datos de exportación en S3.

   La política de clave KMS debe incluir los permisos `kms:CreateGrant` y `kms:DescribeKey`. Para obtener más información acerca del uso de claves KMS en Amazon Aurora, consulte Administración de AWS KMS key.

   Además, si tiene una instrucción deny en la política de claves KMS, asegúrese de excluir explícitamente la entidad principal del servicio de AWS `export.rds.amazonaws.com`.

Puede utilizar una clave de KMS en su cuenta de AWS o puede utilizar una clave KMS en diversas cuentas. Para obtener más información, consulte Uso de un AWS KMS key en diversas cuentas.

4. Exporte la instantánea a Amazon S3 mediante la consola o el comando `start-export-task` de la CLI. Para obtener más información, consulte Creación de tareas de exportación de instantáneas.
5. Para obtener acceso a los datos exportados al bucket de Amazon S3, consulte Carga, descarga y administración de objetos en la Guía del usuario de Amazon Simple Storage Service.

En las siguientes secciones, descubrirá el proceso de configuración, exportación, monitorización, cancelación y resolución de problemas para tareas de exportación de instantáneas de clúster de base de datos.

Temas

- Comentarios sobre la exportación de instantáneas de clústeres de base de datos
- Configuración del acceso a un bucket de Amazon S3
- Creación de tareas de exportación de instantáneas
- Monitoreo de las exportaciones de instantáneas
- Cancelación de una tarea de exportación de instantáneas
- Rendimiento de exportación en Aurora MySQL
- Resolución de problemas en la exportación de instantáneas

Comentarios sobre la exportación de instantáneas de clústeres de base de datos

Limitaciones

Exportar datos de instantáneas de base de datos a Amazon S3 tiene las siguientes limitaciones:

- No puede ejecutar varias tareas de exportación para la misma instantánea del clúster de base de datos simultáneamente. Esto es cierto para las exportaciones completas y parciales.
- Puede tener hasta cinco tareas de exportación de instantáneas de base de datos en curso por Cuenta de AWS.

- No puede exportar los datos de instantáneos desde clústeres de bases de datos de Aurora Serverless v1 a S3.
- Las exportaciones a S3 no admiten prefijos S3 que contengan dos puntos (:).
- Los siguientes caracteres en la ruta del archivo S3 se convierten en guiones bajos (_) durante la exportación:

```
\ ` " (space)
```

- Si una base de datos, esquema o tabla tiene caracteres en su nombre distintos del siguiente, no se admite la exportación parcial. Sin embargo, puede exportar toda la instantánea de base de datos.
  - Letras latinas (A–Z)
  - Dígitos (0–9)
  - Símbolo de dólar ($)
  - Guion bajo (_)
- No se admiten espacios ( ) ni determinados caracteres en los nombres de columna de las tablas de bases de datos. Las tablas con los siguientes caracteres en los nombres de columna se omiten durante la exportación:

```
, ; { } ( ) \n \t = (space)
```

- Las tablas con barras diagonales (/) en el nombre se omiten durante la exportación.
- Las tablas temporales y no registradas de Aurora PostgreSQL se omiten durante la exportación.
- Si los datos contienen un objeto grande, como un BLOB o CLOB, cercano o superior a 500 MB, se producirá un error en la exportación.
- Si una tabla contiene una fila grande cercana o superior a 2 GB, la tabla se omite durante la exportación.
- Para exportaciones parciales, la lista `ExportOnly` tiene un tamaño máximo de 200 KB.
- Es muy recomendable que utilice un nombre exclusivo para cada tarea de exportación. Si no utiliza un nombre de tarea exclusivo, es posible que aparezca el siguiente mensaje de error como el que sigue:

  exportTaskAlreadyExistsFault: Se ha producido un error (exportTaskAlReadyExists) al llamar a la operación StartExportTask: la tarea de exportación con ID *xxxxx* ya existe.
- Puede eliminar una instantánea mientras exporta los datos a S3, pero se le seguirán cobrando los costos de almacenamiento de esa instantánea hasta que se complete la tarea de exportación.

- No puede restaurar los datos de instantáneas exportados de S3 a un nuevo clúster de base de datos.

Convención de nomenclatura de archivos

Los datos exportados para tablas específicas se almacenan en el formato *base_prefix*/*files*, donde el prefijo base es el siguiente:

```
export_identifier/database_name/schema_name.table_name/
```

Por ejemplo:

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

Hay dos convenciones para la forma en que se denominan los archivos.

- Convención actual:

  ```
  batch_index/part-partition_index-random_uuid.format-based_extension
  ```

  El índice de lote es un número secuencial que representa un lote de datos leídos desde la tabla. Si no podemos dividir su tabla en pequeños fragmentos para exportarlos en paralelo, habrá varios índices de lote. Lo mismo ocurre si la tabla está dividida en varias tablas. Habrá varios índices de lote, uno para cada una de las particiones de la tabla principal.

  Si podemos dividir su tabla en pequeños fragmentos para que se lean en paralelo, solo estará la carpeta de índices de lote 1.

  Dentro de la carpeta de índices de lote, habrá uno o varios archivos Parquet que contienen los datos de la tabla. El prefijo del nombre de archivo Parquet es `part-`*partition_index*. Si la tabla está particionada, habrá varios archivos que comiencen por el índice de partición `00000`.

  Puede haber huecos en la secuencia del índice de partición. Esto sucede porque cada partición se obtiene de una consulta por rangos de la tabla. Si no hay datos en el rango de esa partición, se omite ese número secuencial.

  Por ejemplo, supongamos que la columna `id` es la clave principal de la tabla y que sus valores mínimo y máximo son `100` y `1000`. Al intentar exportar esta tabla con nueve particiones, la leemos con consultas paralelas como las siguientes:

```
SELECT * FROM table WHERE id <= 100 AND id < 200
 SELECT * FROM table WHERE id <= 200 AND id < 300
```

Esto debería generar nueve archivos, del `part-00000-`*`random_uuid`*`.gz.parquet` al `part-00008-`*`random_uuid`*`.gz.parquet`. Sin embargo, si no hay filas con ID entre `200` y `350`, una de las particiones completadas estará vacía y no se creará ningún archivo para ella. En el ejemplo anterior, no se crea `part-00001-`*`random_uuid`*`.gz.parquet`.

- Convención anterior:

```
part-partition_index-random_uuid.format-based_extension
```

Es igual a la convención actual, pero sin el prefijo *`batch_index`*, por ejemplo:

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
 part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
 part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

La convención de nomenclatura de archivos está sujeta a cambios. Por lo tanto, cuando lea tablas de destino, recomendamos que lea todo lo que hay dentro del prefijo base de la tabla.

Conversión de datos al exportar a un bucket de Amazon S3

Cuando exporta una instantánea de base de datos a un bucket de Amazon S3, Amazon Aurora convierte los datos al formato Parquet, y exporta y almacena los datos en dicho formato. Para obtener más información sobre Parquet, consulte el sitio web de Apache Parquet.

Parquet almacena todos los datos como uno de los siguientes tipos primitivos:

- BOOLEANO
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY: matriz de bytes de longitud variable, también conocida como binario.

- FIXED_LEN_BYTE_ARRAY. matriz de bytes de longitud fija utilizada cuando los valores tienen un tamaño constante.

Los tipos de datos Parquet son pocos para reducir la complejidad de leer y escribir el formato. Parquet proporciona tipos lógicos para ampliar los tipos primitivos. Un tipo lógico se implementa como una anotación con los datos en un campo de metadatos `LogicalType`. La anotación de tipo lógico explica cómo interpretar el tipo primitivo.

Cuando el tipo lógico `STRING` anota un tipo `BYTE_ARRAY`, indica que la matriz de bytes debe interpretarse como una cadena de caracteres codificada UTF-8. Cuando se complete la tarea de exportación, Amazon Aurora le notificará si se ha producido alguna conversión de cadena. Los datos subyacentes exportados siempre son los mismos que los datos del origen. Sin embargo, debido a la diferencia de codificación en UTF-8, algunos caracteres pueden parecer diferentes a los del origen cuando se leen en herramientas como Athena.

Para obtener más información, consulte Definiciones de tipos lógicos de Parquet en la documentación de Parquet.

Temas

- Mapeo del tipo de datos MySQL con Parquet
- Mapeo de tipos de datos PostgreSQL con Parquet

Mapeo del tipo de datos MySQL con Parquet

En la siguiente tabla se muestra el mapeo de los tipos de datos MySQL con los tipos de datos Parquet cuando los datos se convierten y se exportan a Amazon S3.

Tipo de datos de origen	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de conversión
Tipos de datos numéricos			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN _BYTE_ARRAY(9)	DECIMAL(20,0)	Parquet solo admite tipos firmados, por lo que el mapeo requiere un byte

Tipo de datos de origen	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de conversión
			adicional (8 más 1) para almacenar el tipo BIGINT_UNSIGNED.
BIT	BYTE_ARRAY		
DECIMAL	INT32	DECIMAL (p,s)	Si el valor de origen es inferior a 2^{31}, se almacena como INT32.
	INT64	DECIMAL (p,s)	Si el valor de origen es 2^{31} o superior, pero inferior a 2^{63}, se almacena como INT64.
	FIXED_LEN _BYTE_ARRAY(N)	DECIMAL (p,s)	Si el valor de origen es 2^{63} o superior, se almacena como FIXED_LEN _BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	Parquet no admite una precisión decimal superior a 38. El valor decimal se convierte en una cadena en un tipo BYTE_ARRAY y se codifica como UTF8.
DOUBLE	DOUBLE		
FLOAT	DOUBLE		

Tipo de datos de origen	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de conversión
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		
NUMERIC	INT32	DECIMAL (p,s)	Si el valor de origen es inferior a 2^{31}, se almacena como INT32.
	INT64	DECIMAL (p,s)	Si el valor de origen es 2^{31} o superior, pero inferior a 2^{63}, se almacena como INT64.
	FIXED_LEN _ARRAY(N)	DECIMAL (p,s)	Si el valor de origen es 2^{63} o superior, se almacena como FIXED_LEN _BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	Parquet no admite una precisión numérica superior a 38. Este valor numérico se convierte en una cadena en un tipo BYTE_ARRAY y se codifica como UTF8.

Tipo de datos de origen	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de conversión
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		
TINYINT UNSIGNED	INT32	INT(16, true)	
Tipos de datos de cadena			
BINARY	BYTE_ARRAY		
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	

Tipo de datos de origen	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de conversión
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
Tipos de datos de fecha y hora			
FECHA	BYTE_ARRAY	STRING	Una fecha se convierte en una cadena en un tipo BYTE_ARRAY y se codifica como UTF8.
DATETIME	INT64	TIMESTAMP_MICROS	
TIME	BYTE_ARRAY	STRING	Un tipo TIME se convierte en una cadena en un BYTE_ARRAY y se codifica como UTF8.
TIMESTAMP	INT64	TIMESTAMP_MICROS	
YEAR	INT32		
Tipos de datos geométricos			
GEOMETRY	BYTE_ARRAY		
GEOMETRYCOLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		

Tipo de datos de origen	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de conversión
POINT	BYTE_ARRAY		
POLYGON	BYTE_ARRAY		
Tipos de datos de JSON			
JSON	BYTE_ARRAY	STRING	

Mapeo de tipos de datos PostgreSQL con Parquet

En la tabla siguiente se muestra el mapeo de los tipos de datos PostgreSQL con los tipos de datos Parquet cuando los datos se convierten y se exportan a Amazon S3.

Tipos de datos de PostgreSQL	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de mapeo
Tipos de datos numéricos			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	Un tipo DECIMAL se convierte en una cadena en un tipo BYTE_ARRAY y se codifica como UTF8. Esta conversión se realiza para evitar complicaciones debidas a la precisión de los datos y los valores de datos que

Tipos de datos de PostgreSQL	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de mapeo
			no son un número (NaN).
DOUBLE PRECISION	DOUBLE		
INTEGER	INT32		
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT(16, true)	
SMALLSERIAL	INT32	INT(16, true)	
Tipos de datos de cadena y relacionados			
ARRAY	BYTE_ARRAY	STRING	Una matriz se convierte en una cadena y se codifica como BINARY (UTF8). Esta conversión se realiza para evitar complicaciones debido a la precisión de los datos, valores de datos que no son un número (NaN) y valores de datos de tiempo.
BIT	BYTE_ARRAY	STRING	

Tipos de datos de PostgreSQL	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de mapeo
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	
XML	BYTE_ARRAY	STRING	
Tipos de datos de fecha y hora			
FECHA	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP WITH TIME ZONE	BYTE_ARRAY	STRING	
Tipos de datos geométricos			
BOX	BYTE_ARRAY	STRING	

Tipos de datos de PostgreSQL	Tipo primitivo de Parquet	Anotación de tipo lógico	Notas de mapeo
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	
Tipos de datos JSON			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	
Otros tipos de datos			
BOOLEANO	BOOLEANO		
CIDR	BYTE_ARRAY	STRING	Tipo de datos de red
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	Tipo de datos de red
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	N/A		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

Configuración del acceso a un bucket de Amazon S3

Una vez identificado el bucket de Amazon S3, dé a la instantánea permiso para acceder a él.

Temas

- Identificación del bucket de Amazon S3 para exportación
- Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM
- Uso de un bucket de Amazon S3 en diversas cuentas
- Uso de un AWS KMS key en diversas cuentas

Identificación del bucket de Amazon S3 para exportación

Identifique el bucket de Amazon S3 al que se exportará la instantánea de base de datos. Utilice un bucket de S3 ya existente, o bien cree un bucket S3 nuevo.

> **Note**
>
> El bucket de S3 al que se realizará la exportación debe estar en la misma región de AWS que la instantánea.

Para obtener más información acerca de cómo trabajar con buckets de Amazon S3, consulte lo siguiente en Guía del usuario de Amazon Simple Storage Service:

- Cómo se consultan las propiedades de un bucket de S?
- Cómo puedo habilitar el cifrado predeterminado para un bucket de Amazon S?
- Cómo se puede crear un bucket de S?

Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM

Antes de exportar datos de instantáneas de bases de datos a Amazon S3, conceda a las tareas de exportación de instantáneas permiso de acceso de escritura al bucket de Amazon S3.

Para conceder este permiso, cree una política de IAM que proporcione acceso al bucket y cree un rol de IAM y adjunte la política al rol. Más adelante, puede asignar el rol de IAM a la tarea de exportación de instantáneas.

> **Important**
>
> Si prevé utilizar la Consola de administración de AWS para exportar la instantánea, puede elegir crear la política de IAM y el rol automáticamente al exportar la instantánea. Para obtener instrucciones, consulte Creación de tareas de exportación de instantáneas.

Para dar a las tareas de instantáneas de base de datos acceso a Amazon S3

1. Cree una política de IAM. Esta política proporciona los permisos de bucket y objeto que permiten a la tarea de exportación de instantáneas obtener acceso a Amazon S3.

   En la política, incluya las siguientes acciones obligatorias para permitir transferir archivos desde Amazon Aurora a un bucket de S3:

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

   En la política, incluya los siguientes recursos para identificar el bucket de S3 y los objetos incluidos en él. En la siguiente lista de recursos se muestra el formato de nombre de recurso de Amazon (ARN) para obtener acceso a Amazon S3.

   - `arn:aws:s3:::amzn-s3-demo-bucket`
   - `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Para obtener más información sobre cómo crear una política de IAM para Amazon Aurora, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM. Consulte también el Tutorial: Crear y asociar su primera política administrada por el cliente en la Guía del usuario de IAM.

   El siguiente comando de la AWS CLI crea una política de IAM denominada `ExportPolicy` con estas opciones. Otorga acceso a un bucket denominado *amzn-s3-demo-bucket*.

> **Note**
>
> Después de crear la política, apunte el ARN de esta. Cuando asocia la política a un rol de IAM, necesita el ARN para realizar un paso posterior.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Cree un rol de IAM que Aurora pueda asumir en su nombre para acceder a sus buckets de Amazon S3. Para obtener más información, vea Crear un rol para delegar permisos a un IAM usuario en Guía del usuario de IAM.

   En el siguiente ejemplo se muestra cómo se usa el comando de la AWS CLI para crear un rol denominado `rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
```

```
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Asocie la política de IAM que creó al rol de IAM creado.

   El siguiente comando de la AWS CLI asocia la política creada anteriormente al rol denominado `rds-s3-export-role`. Sustituya *`your-policy-arn`* por el ARN de la política que ha apuntado en el paso anterior.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Uso de un bucket de Amazon S3 en diversas cuentas

Puede utilizar buckets de Amazon S3 en cuentas de AWS. Para utilizar un bucket en diversas cuentas, agregue una política de bucket para permitir el acceso al rol de IAM que está utilizando para las exportaciones de S3. Para obtener más información, consulte el Ejemplo 2: Propietario del bucket que concede permisos de bucket en diversas cuentas.

- Adjunte una política de bucket a su bucket, como se muestra en el siguiente ejemplo.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
```

```
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

Uso de un AWS KMS key en diversas cuentas

Puede utilizar una AWS KMS key en diversas cuentas para cifrar las exportaciones de Amazon S3. En primer lugar, agregue una política de claves a la cuenta local y, a continuación, agregue las políticas de IAM en la cuenta externa. Para obtener más información, consulte Allowing users in other accounts to use a KMS key (Permitir que los usuarios de otras cuentas utilicen una clave KMS).

Para utilizar una clave KMS en diversas cuentas

1. Agregue una política de claves a la cuenta local.

   El siguiente ejemplo proporciona `ExampleRole` y `ExampleUser` en la cuenta externa 444455556666 permisos en la cuenta local 123456789012.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/ExampleRole",
            "arn:aws:iam::444455556666:user/ExampleUser"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
```

```
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "*"
}
```

2. Agregar políticas de IAM a la cuenta externa.

   La siguiente política de IAM de ejemplo permite a la entidad principal utilizar la clave KMS en la cuenta 123456789012 para operaciones criptográficas. Para conceder este permiso a `ExampleRole` y `ExampleUser` de la cuenta 444455556666, adjunte la política en esa cuenta.

```
{
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Creación de tareas de exportación de instantáneas

Cree tareas de exportación de instantáneas para exportar datos desde su instantánea a un bucket de Amazon S3. Puede tener hasta cinco tareas de exportación de instantáneas de base de datos en curso por Cuenta de AWS.

> **Note**
>
> La exportación de instantáneas de RDS puede tardar un tiempo en función del tipo y tamaño de la base de datos. La tarea de exportación primero restaura y escala toda la base de datos antes de extraer los datos a Amazon S3. El progreso de la tarea durante esta fase

se muestra como Starting (Iniciándose). Cuando la tarea cambia a exportar datos a S3, el progreso se muestra como In progress (En curso).
El tiempo que tarda la exportación en completarse depende de los datos almacenados en la base de datos. Por ejemplo, las tablas con columnas de índice o claves primarias numéricas bien distribuidas se exportarán más rápido. Las tablas que no contienen una columna adecuada para la partición y las tablas con un solo índice en una columna basada en cadenas tardarán más tiempo. Este tiempo de exportación más prolongado se produce porque la exportación utiliza un proceso de subproceso único más lento.

Puede exportar una instantánea de base de datos a Amazon S3 mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Si utiliza una función Lambda para exportar una instantánea, agregue la acción `kms:DescribeKey` a la política de la función Lambda. Para obtener más información, consulte Permisos de AWS Lambda.

Consola

La opción de la consola Export to Amazon S3 (Exportar a Amazon S3) solo aparece para las instantáneas que se pueden exportar a Amazon S3. Es posible que una instantánea no esté disponible para la exportación debido a las siguientes razones:

- El motor de base de datos no es compatible con la exportación de S3.
- La versión de la instancia de base de datos no es compatible con la exportación de S3.
- La exportación de S3 no se admite en la región de AWS donde se creó la instantánea.

Para exportar una instantánea de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. En las pestañas, elija el tipo de instantánea que desee exportar.
4. En la lista de instantáneas, elija la instantánea que desee exportar.
5. En Actions (Acciones), seleccione Export to Amazon S3 (Exportar a Amazon S3).

   Se visualizará la ventana Export to Amazon S3 (Exportar a Amazon S3).

6. En Export Identifier (Identificador de exportación), escriba un nombre para identificar la tarea de exportación. Este valor también se utiliza para el nombre del archivo creado en el bucket de S3.
7. Elija los datos que desea exportar:

   - Seleccione All (Todo) para exportar todos los datos de la instantánea.
   - Seleccione Partial (Parcial) para exportar partes específicas de la instantánea. Para identificar qué partes de la instantánea exportar, introduzca una o más bases de datos, esquemas o tablas para Identifiers (Identificadores), separadas por espacios.

     Use el siguiente formato:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
     [.tablen]
     ```

     Por ejemplo:

     ```
     mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
      mydatabase2.myschema2.mytable2
     ```
8. Para el S3 bucket (Bucket de S3), elija el bucket al que desee realizar la exportación.

   Para asignar los datos exportados a la ruta de una carpeta en el bucket de S3, escriba la ruta opcional para el S3 prefix (Prefijo de S3).
9. Para el rol de IAM, elija un rol que le conceda acceso de escritura al bucket de S3 elegido o cree un nuevo rol.

   - Si ha creado un rol siguiendo los pasos indicados en Proporcionar acceso a un bucket de Amazon S3 mediante un rol de IAM, elija dicho rol.
   - Si no ha creado un rol que le conceda acceso de escritura al bucket de S3 elegido, elija Create a new role (Crear un nuevo rol) para crear el rol automáticamente. A continuación, escriba un nombre para el rol en el IAM role name (Nombre del rol de IAM).
10. En AWS KMS key, ingrese el ARN de la clave que debe utilizarse para cifrar los datos exportados.
11. Elija Export to Amazon S3 (Exportar a Amazon S3).

AWS CLI

Para exportar una instantánea de base de datos a Amazon S3 mediante la AWS CLI, ejecute el comando start-export-task con las siguientes opciones obligatorias:

- `--export-task-identifier`
- `--source-arn`
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

En los siguientes ejemplos, la tarea de exportación de instantáneas se denomina *my-snapshot-export* y exporta una instantánea a un bucket de S3 denominado *amzn-s3-demo-destination-bucket*.

Example

Para Linux, macOS o:Unix

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

En:Windows

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

A continuación, se muestra un resultado de ejemplo.

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
```

```
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

Para proporcionar la ruta de una carpeta del bucket S3 para la exportación de instantáneas, incluya la opción `--s3-prefix` en el comando start-export-task.

API de RDS

Para exportar una instantánea de base de datos a Amazon S3 con la API de Amazon RDS, ejecute la operación StartExportTask con los siguientes parámetros obligatorios:

- `ExportTaskIdentifier`
- `SourceArn`
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Monitoreo de las exportaciones de instantáneas

Puede monitorear las exportaciones de instantáneas de bases de datos mediante Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para monitorear las exportaciones de instantáneas de bases de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Exports in Amazon S3 (Exportaciones en Amazon S3).

   Las exportaciones de instantáneas de base de datos se indican en la columna Source type (Tipo de fuente). El estado de exportación se muestra en la columna Status (Estado).

3. Para ver información detallada acerca de la exportación de una instantánea específica, elija la tarea de exportación.

AWS CLI

Para monitorear exportaciones de instantáneas de bases de datos mediante la AWS CLI, ejecute el comando describe-export-tasks .

En el ejemplo siguiente se muestra cómo mostrar la información actual acerca de todas las exportaciones de instantáneas.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
        },
{
            "Status": "COMPLETE",
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
```

```
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket my-exports isn't located in the current AWS
 Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

Para mostrar información sobre una exportación de instantáneas específica, incluya la opción `--export-task-identifier` con el comando `describe-export-tasks`. Para filtrar la salida, incluya la opción `--Filters`. Para obtener más opciones, consulte el comando describe-export-tasks.

API de RDS

Para mostrar información sobre las exportaciones de instantáneas de bases de datos mediante la API de Amazon RDS, ejecute la operación DescribeExportTasks.

Para realizar un seguimiento del flujo de trabajo de exportación o para iniciar otro flujo de trabajo, puede suscribirse a temas de Amazon Simple Notification Service. Para obtener más información sobre Amazon SNS, consulte Uso de notificaciones de eventos de Amazon RDS.

Cancelación de una tarea de exportación de instantáneas

Puede cancelar una tarea de exportación de instantáneas de bases de datos mediante Consola de administración de AWS, la AWS CLI o la API de RDS.

> **Note**
>
> La cancelación de una tarea de exportación de instantáneas no elimina los datos exportados a Amazon S3. Para obtener información acerca de cómo eliminar los datos mediante la consola, consulte ¿Cómo se eliminan objetos de un bucket de S3? Para eliminar los datos mediante la CLI, ejecute el comando delete-object.

Consola

Para cancelar una tarea de exportación de una instantánea

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Exports in Amazon S3 (Exportaciones en Amazon S3).

   Las exportaciones de instantáneas de base de datos se indican en la columna Source type (Tipo de fuente). El estado de exportación se muestra en la columna Status (Estado).
3. Elija la tarea de exportación de instantáneas que desee cancelar.
4. Elija Cancel.
5. Seleccione Cancel export task (Cancelar tarea de exportación) en la página de confirmación.

AWS CLI

Para cancelar una tarea de exportación de instantáneas mediante la AWS CLI, ejecute el comando cancel-export-task . El comando requiere la opción `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
```

```
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
}
```

API de RDS

Para cancelar una tarea de exportación de instantáneas mediante la API de Amazon RDS, ejecute la operación CancelExportTask con el parámetro `ExportTaskIdentifier`.

Rendimiento de exportación en Aurora MySQL

Las instantáneas de clústeres de base de datos de Aurora MySQL versión 2 y versión 3 utilizan un mecanismo de exportación avanzado para mejorar el rendimiento y reducir el tiempo de exportación. El mecanismo incluye optimizaciones, como varios subprocesos de exportación y consultas paralelas de Aurora MySQL, para aprovechar la arquitectura de almacenamiento compartido de Aurora. Las optimizaciones se aplican de forma adaptativa, según el tamaño y la estructura del conjunto de datos.

No es necesario activar la consulta paralela para utilizar el proceso de exportación más rápido, pero el proceso tiene las mismas limitaciones que la consulta paralela. Además, no se admiten algunos valores de datos, como las fechas en las que el día del mes es `0` o el año es `0000`. Para obtener más información, consulte Consulta paralela para Amazon Aurora MySQL.

Al aplicar optimizaciones de rendimiento, es posible que también vea archivos Parquet mucho más grandes (aproximadamente 200 GB) para las exportaciones de Aurora MySQL versiones 2 y 3.

Si no se puede utilizar el proceso de exportación más rápido, por ejemplo, debido a tipos de datos o valores incompatibles, Aurora cambia automáticamente a un modo de exportación de un solo subproceso sin consultas paralelas. Según el proceso que se utilice y la cantidad de datos que se exporten, el rendimiento de la exportación puede variar.

Resolución de problemas en la exportación de instantáneas

Utilice las siguientes secciones para solucionar los mensajes de error y los errores de permisos de PostgreSQL para las tareas de exportación de clústeres de bases de datos a Amazon S3.

Mensajes de error para tareas de exportación de Amazon S3

En la tabla siguiente se describen los mensajes que se devuelven cuando se producen errores en las tareas de exportación de Amazon S3.

Mensaje de error	Descripción
Se ha producido un error interno desconocido.	La tarea no se pudo completar debido a un error, excepción o falla desconocidos.
Ocurrió un error interno desconocido al escribir los metadatos de la tarea de exportación en el bucket de S3 [nombre del bucket].	La tarea no se pudo completar debido a un error, excepción o falla desconocidos.
La exportación de RDS no pudo escribir los metadatos de la tarea de exportación porque no puede asumir el rol de IAM [ARN de rol].	La tarea de exportación asume el rol de IAM para validar si está permitido escribir metadatos en el bucket de S3. Si la tarea no puede asumir su rol de IAM, muestra un error.
La exportación de RDS no pudo escribir los metadatos de la tarea de exportación en el bucket de S3 [nombre del bucket] que utiliza el rol de IAM [ARN de rol] con la clave KMS [ID de clave]. Código de error: [código de error]	Faltan uno o más permisos, por lo que la tarea de exportación no puede acceder al bucket de S3. Este mensaje de error aparece cuando se recibe uno de los siguientes códigos de error: • `AWSSecurityTokenServiceException` con el código de error `AccessDenied` • `AmazonS3Exception` con el código de error `NoSuchBucket`, `AccessDenied`, `KMS.KMSInvalidStateException`, `403 Forbidden`, o `KMS.DisabledException`

Mensaje de error	Descripción
	Estos códigos de error indican que la configuración del rol de IAM, el bucket de S3 o la clave KMS es incorrecta.
El rol de IAM [ARN de rol] no está autorizado para llamar a [acción de S3] en el bucket de S3 [nombre del bucket]. Revise sus permisos y vuelva a intentar la exportación.	La política de IAM está mal configurada. Falta el permiso para la acción específica de S3 en el bucket de S3, que provoca que falle la tarea de exportación.
Error en la verificación de claves KMS Verifique las credenciales de la clave KMS e inténtelo de nuevo.	Error en la verificación de credenciales de la clave KMS.
Error en la verificación de credenciales de S3. Verifique los permisos de su bucket de S3 y de la política de IAM.	Error en la verificación de credenciales de S3.
El bucket de S3 [nombre del bucket] no es válido. O no se encuentra en la Región de AWS actual o no existe. Revise el nombre del bucket de S3 e intente hacer la exportación de nuevo.	El bucket de S3 no es válido.
El bucket de S3 [nombre del bucket] no se encuentra en la Región de AWS actual. Revise el nombre del bucket de S3 e intente hacer la exportación de nuevo.	El bucket de S3 está en la Región de AWS equivocada.

Solución de problemas de errores de permisos de PostgreSQL

Al exportar bases de datos PostgreSQL a Amazon S3, es posible que vea un error `PERMISSIONS_DO_NOT_EXIST` que indica que se omitieron ciertas tablas. Esto suele deberse a que el superusuario, que se especifica al crear la instancia de base de datos, no tiene permisos para acceder a dichas tablas.

Para corregir este error, ejecute el siguiente comando:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Para obtener más información sobre los privilegios de superusuario, consulte Privilegios de la cuenta de usuario maestro.

Restauración de un clúster de base de dato a un momento indicado

Para restaurar un clúster de bases de datos a un momento específico en el tiempo, cree un nuevo clúster de bases de datos.

Cuando restaure un clúster de bases de datos a un momento específico en el tiempo, puede elegir el grupo de seguridad de nube virtual privada (VPC) predeterminado. O bien, puede aplicar un grupo de seguridad de VPC personalizado al clúster de bases de datos.

Las clústeres de base de datos restaurados se asocian automáticamente con los grupos de opciones y parámetros de base de datos predeterminados. Sin embargo, puede aplicar grupos de parámetros personalizados especificándolos durante una restauración.

Amazon RDS carga los registros de los clústeres de bases de datos en Amazon S3 de forma continua. Para ver la última hora restaurable para un clúster de bases de datos, utilice el comando describe-db-clústers de la AWS CLI y observe el valor devuelto en el campo `LatestRestorableTime` para el clúster de bases de datos.

Puede restaurar a cualquier punto en el tiempo dentro del periodo de retención de copia de seguridad. Para ver la hora restaurable más temprana para un clúster de bases de datos, utilice el comando describe-db-clústers de la AWS CLI y observe el valor devuelto en el campo `EarliestRestorableTime` para el clúster de bases de datos.

El periodo de retención de las copias de seguridad del clúster de base de datos restaurado es el mismo que el del clúster de la base de datos de origen.

> **Note**
>
> La información de este tema se aplica a Amazon Aurora. Para obtener información sobre la restauración de una instancia de base de datos de Amazon RDS, consulte Restauración de una instancia de base de datos a un momento especificado.
> Para obtener más información sobre la realización de copia de seguridad y restauración de un clúster de bases de datos de Aurora, consulte Información general de copias de seguridad y restauración de un clúster de base de datos Aurora.
> Para Aurora MySQL, puede restaurar un clúster de bases de datos aprovisionado en un clúster de bases de datos de Aurora Serverless. Para obtener más información, consulte Restauración de un clúster de bases de datos de Aurora Serverless v1.

También puede utilizar AWS Backup para administrar copias de seguridad de clústeres de base de datos de Amazon Aurora. Si su clúster de base de datos está asociado a un plan de copia de seguridad en AWS Backup, ese plan se utiliza para la recuperación en un momento dado. Para obtener más información, consulte Restauración de un clúster de base de datos a un momento especificado mediante AWS Backup.

Para obtener información sobre la restauración de un clúster de base de datos Aurora o un clúster global con una versión del Soporte extendido de RDS, consulte Restauración de un clúster de base de datos de Aurora con Soporte extendido de Amazon RDS.

Restaure un clúster de base de datos a un momento especificado a partir de una copia de seguridad automatizada, una copia de seguridad automatizada retenida o mediante la AWS Backup.

Temas

- Restauración de un clúster de base de datos a un momento determinado
- Restauración de un clúster de base de datos a un momento especificado a partir de una copia de seguridad automatizada retenida
- Restauración de un clúster de base de datos a un momento especificado mediante AWS Backup

Restauración de un clúster de base de datos a un momento determinado

Puede restaurar un clúster de bases de datos a un momento dado mediante Consola de administración de AWS, AWS CLI o la API de RDS.

Consola

Para restaurar un clúster de bases de datos a un momento indicado

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Automated backups (Copias de seguridad automatizadas).

   Las copias de seguridad automatizadas se muestran en la pestaña Current Region (Región actual).

3. Elija el clúster de bases de datos de que desea restaurar.
4. Para Actions (Acciones), elija Restore to point in time (Restaurar a un momento dado).

   Aparecerá la ventana Restore to point in time (Restaurar a un momento dado).
5. Elija Latest restorable time (Última hora de restauración) para restaurar a la última hora posible o elija Custom (Personalizar) para elegir una hora.

   Si elige Custom (Personalizar), ingrese la fecha y hora a la que quiere restaurar el clúster.

   > **Note**
   >
   > Las horas se muestran en su zona horaria local, que se indica mediante una diferencia de la hora universal coordinada (UTC). Por ejemplo, UTC-5 es la hora estándar del Este/horario de verano central.
6. En Identificador de clúster de bases de datos, ingrese el nombre del clúster de base de datos restaurado de destino. El nombre debe ser único.
7. Elija otras opciones según sea necesario, como la clase de instancia de base de datos y la configuración de almacenamiento del clúster de base de datos.

   Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.
8. Elija Restore to point in time (Restaurar a un momento dado).

AWS CLI

Para restaurar un clúster de base de datos a un momento especificado, use el comando restore-db-clúster-to-point-in-time para crear un nuevo clúster de base de datos.

Puede especificar otras opciones. Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.

Si se proporcionan etiquetas en la solicitud, las etiquetas proporcionadas se aplican al clúster de base de datos restaurado. Si no se proporcionan etiquetas en la solicitud y si el clúster de base de datos de origen está activo en la región y las tiene, Aurora agrega las etiquetas más recientes del clúster de base de datos de origen al clúster de base de datos restaurado.

Example

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
       --source-db-cluster-identifier mysourcedbcluster \
       --db-cluster-identifier mytargetdbcluster \
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Para Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
       --source-db-cluster-identifier mysourcedbcluster ^
       --db-cluster-identifier mytargetdbcluster ^
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Si usa la consola para restaurar un clúster de bases de datos, Amazon RDS crea automáticamente la instancia primaria (de escritura) del clúster de bases de datos. Si usa la AWS CLI para restaurar un clúster de bases de datos, debe crear expresamente la instancia primaria del clúster de bases de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos.

Para crear la instancia primaria del clúster de bases de datos, llame al comando create-db-instance de AWS CLI. Incluya el nombre del clúster de bases de datos como valor de la opción `--db-cluster-identifier`.

API de RDS

Para restaurar un clúster de bases de datos a un momento especificado, llame a la operación `RestoreDBClusterToPointInTime` de la API de Amazon RDS con los siguientes parámetros:

- `SourceDBClusterIdentifier`

- `DBClusterIdentifier`
- `RestoreToTime`

> **Important**
>
> Si usa la consola para restaurar un clúster de bases de datos, Amazon RDS crea automáticamente la instancia primaria (de escritura) del clúster de bases de datos. Si usa la API de RDS para restaurar un clúster de bases de datos a un momento especificado, debe crear expresamente la instancia primaria del clúster de bases de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos.
> Para crear la instancia primaria del clúster de bases de datos, llame a la operación de la API de RDS CreateDBInstance. Incluya el nombre del clúster de bases de datos como valor del parámetro `DBClusterIdentifier`.

Restauración de un clúster de base de datos a un momento especificado a partir de una copia de seguridad automatizada retenida

Puede restaurar un clúster de base de datos a partir de una copia de seguridad automatizada retenida después de eliminar el clúster de base de datos de origen, si la copia de seguridad se encuentra dentro del período de retención del clúster de origen. El proceso es similar al de la restauración de un clúster de base de datos a partir de una copia de seguridad automatizada.

> **Note**
>
> No puede restaurar un clúster de base de datos de Aurora Serverless v1 mediante este procedimiento, ya que no se conservan las copias de seguridad automáticas de los clústeres de Aurora Serverless v1.

Consola

Para restaurar un clúster de bases de datos a un momento indicado

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Automated backups (Copias de seguridad automatizadas).

3. Elija la pestaña Retenidas.

4. Elija el clúster de bases de datos de que desea restaurar.
5. Para Actions (Acciones), elija Restore to point in time (Restaurar a un momento dado).

   Aparecerá la ventana Restore to point in time (Restaurar a un momento dado).
6. Elija Latest restorable time (Última hora de restauración) para restaurar a la última hora posible o elija Custom (Personalizar) para elegir una hora.

   Si elige Custom (Personalizar), ingrese la fecha y hora a la que quiere restaurar el clúster.

   > **Note**
   >
   > Las horas se muestran en su zona horaria local, que se indica mediante una diferencia de la hora universal coordinada (UTC). Por ejemplo, UTC-5 es la hora estándar del Este/horario de verano central.

7. En Identificador de clúster de bases de datos, ingrese el nombre del clúster de base de datos restaurado de destino. El nombre debe ser único.
8. Elija otras opciones según sea necesario, como la clase de instancia de base de datos.

   Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.
9. Elija Restore to point in time (Restaurar a un momento dado).

AWS CLI

Para restaurar un clúster de base de datos a un momento especificado, use el comando restore-db-clúster-to-point-in-time para crear un nuevo clúster de base de datos.

Puede especificar otras opciones. Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.

Se admite el etiquetado de recursos para esta operación. Al utilizar la opción `--tags`, se ignoran las etiquetas del clúster de base de datos de origen y se utilizan las proporcionadas. De lo contrario, se utilizan las etiquetas más recientes del clúster de origen.

Example

Para Linux, macOS o:Unix

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE \
    --db-cluster-identifier mytargetdbcluster \
    --restore-to-time 2017-10-14T23:45:00.000Z
```

En:Windows

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE ^
    --db-cluster-identifier mytargetdbcluster ^
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Si usa la consola para restaurar un clúster de bases de datos, Amazon RDS crea automáticamente la instancia primaria (de escritura) del clúster de bases de datos. Si usa la AWS CLI para restaurar un clúster de bases de datos, debe crear expresamente la instancia primaria del clúster de bases de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos.

Para crear la instancia primaria del clúster de bases de datos, llame al comando create-db-instance de AWS CLI. Incluya el nombre del clúster de bases de datos como valor de la opción `--db-cluster-identifier`.

API de RDS

Para restaurar un clúster de bases de datos a un momento especificado, llame a la operación `RestoreDBClusterToPointInTime` de la API de Amazon RDS con los siguientes parámetros:

- `SourceDbClusterResourceId`
- `DBClusterIdentifier`

- `RestoreToTime`

> **Important**
>
> Si usa la consola para restaurar un clúster de bases de datos, Amazon RDS crea automáticamente la instancia primaria (de escritura) del clúster de bases de datos. Si usa la API de RDS para restaurar un clúster de bases de datos a un momento especificado, debe crear expresamente la instancia primaria del clúster de bases de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos.
> Para crear la instancia primaria del clúster de bases de datos, llame a la operación de la API de RDS CreateDBInstance. Incluya el nombre del clúster de bases de datos como valor del parámetro `DBClusterIdentifier`.

Restauración de un clúster de base de datos a un momento especificado mediante AWS Backup

Puede utilizar AWS Backup para gestionar sus copias de seguridad automatizadas y, a continuación, restaurarlas a un momento especificado. Para ello, debe crear un plan de copias de seguridad en AWS Backup y asignar su clúster de base de datos como recurso. A continuación, habilita las copias de seguridad continuas para la PITR en la regla de copia de seguridad. Para obtener más información sobre los planes de copia de seguridad y las reglas de copia de seguridad, consulte AWS Backup Developer Guide.

Habilitación de copias de seguridad continuas en AWS Backup

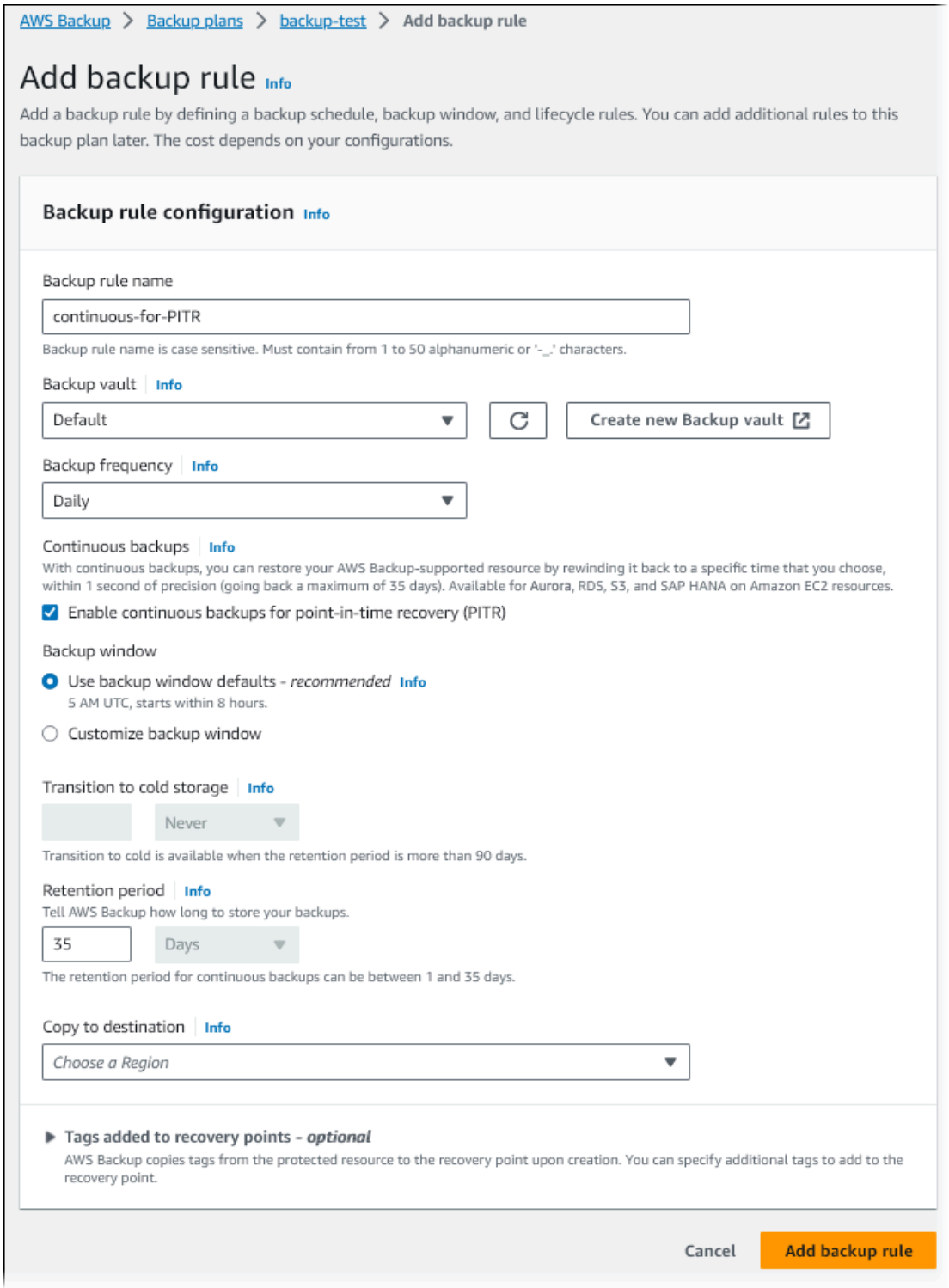
A continuación, habilita las copias de seguridad continuas en las reglas de copia de seguridad.

Para habilitar las copias de seguridad continuas para la PITR

1. Inicie sesión en la Consola de administración de AWS y abra la consola AWS Backup en https://console.aws.amazon.com/backup.
2. En el panel de navegación, seleccione Backup plans (Planes de copias de seguridad).
3. En Nombre del plan de copia de seguridad, seleccione el plan de copia de seguridad que utilizará para hacer una copia de seguridad del clúster de base de datos.
4. En la sección Reglas de copia de seguridad, elija Agregar regla de copia de seguridad.

Se muestra la página Agregar regla de copia de seguridad.

5. Seleccione la casilla de verificación Habilite las copias de seguridad continuas para la recuperación en un momento dado (PITR).



6. Elija otros ajustes según sea necesario y, a continuación, elija Agregar regla de copia de seguridad.

Restauración a partir de una copia de seguridad continua enAWS Backup

La restauración se realiza a un momento especificado desde un almacén de copias de seguridad.

Consola

Puede utilizar la Consola de administración de AWS para restaurar un clúster de bases de datos a un momento especificado.

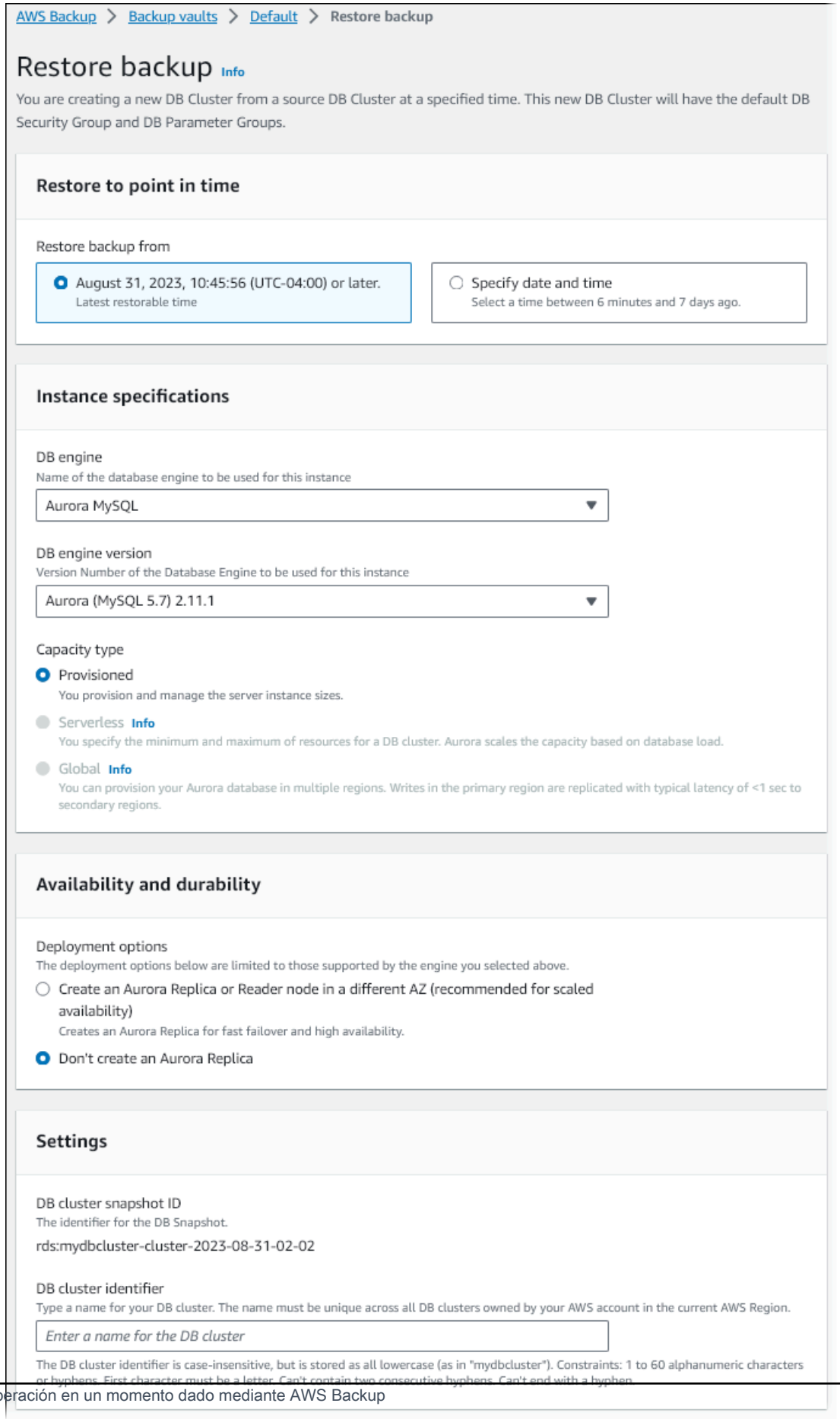
Para restaurar a partir de una copia de seguridad continua enAWS Backup

1. Inicie sesión en la Consola de administración de AWS y abra la consola AWS Backup en https://console.aws.amazon.com/backup.
2. En el panel de navegación, elija Backup vaults (Almacenes de copia de seguridad).
3. Elija el almacén de copias de seguridad que contenga la copia de seguridad continua, por ejemplo, Predeterminado.

   Se muestra la página de detalles del almacén de copias de seguridad.
4. En Puntos de recuperación, seleccione el punto de recuperación para la copia de seguridad automatizada.

   Tiene un tipo de copia de seguridad Continuo y un nombre con `continuous:cluster-`*`AWS-Backup-job-number`*.
5. En Acciones, elija Restaurar.

   Se muestra la página Restaurar copia de seguridad.


AWS Backup > Backup vaults > Default > Restore backup
Restore backup Info
You are creating a new DB Cluster from a source DB Cluster at a specified time. This new DB Cluster will have the default DB Security Group and DB Parameter Groups.
Restore to point in time
Restore backup from
August 31, 2023, 10:45:56 (UTC-04:00) or later.
Latest restorable time
Specify date and time
Select a time between 6 minutes and 7 days ago.
Instance specifications
DB engine
Name of the database engine to be used for this instance
Aurora MySQL
DB engine version
Version Number of the Database Engine to be used for this instance
Aurora (MySQL 5.7) 2.11.1
Capacity type
Provisioned
You provision and manage the server instance sizes.
Serverless Info
You specify the minimum and maximum of resources for a DB cluster. Aurora scales the capacity based on database load.
Global Info
You can provision your Aurora database in multiple regions. Writes in the primary region are replicated with typical latency of <1 sec to secondary regions.
Availability and durability
Deployment options
The deployment options below are limited to those supported by the engine you selected above.
Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.
Don't create an Aurora Replica
Settings
DB cluster snapshot ID
The identifier for the DB Snapshot.
rds:mydbcluster-cluster-2023-08-31-02-02
DB cluster identifier
Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.
Enter a name for the DB cluster
The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

6. En Restaurar a un momento dado, seleccione Especificar fecha y hora para restaurar a un momento específico en el tiempo.
7. Elija los demás ajustes necesarios para restaurar el clúster de base de datos y, a continuación, elija Restaurar copia de seguridad.

   Se muestra la página Trabajos, que muestra el panel Trabajos de restauración. En la parte superior de la página, aparecerá un mensaje con información sobre el trabajo de restauración.

Una vez restaurado el clúster de base de datos, tiene que agregar la instancia de base de datos principal (de escritura). Para crear la instancia primaria del clúster de bases de datos, llame al comando create-db-instance de AWS CLI. Incluya el nombre del clúster de bases de datos como valor del parámetro `--db-cluster-identifier`.

CLI

Utilice el comando de la AWS CLI start-restore-job para restaurar el clúster de base de datos a un momento especificado. Se requieren los siguientes parámetros:

- `--recovery-point-arn`: el nombre de recurso de Amazon (ARN) del punto de recuperación desde el que se realiza la restauración.
- `--resource-type`: utilice `Aurora`.
- `--iam-role-arn`: el ARN del rol de IAM que utiliza para las operaciones de AWS Backup.
- `--metadata`: los metadatos que se utilizan para restaurar el clúster de base de datos. Se requieren los siguientes parámetros:
  - `DBClusterIdentifier`
  - `Engine`
  - `RestoreToTime` o `UseLatestRestorableTime`

El siguiente ejemplo muestra cómo restaurar un clúster de base de datos a un momento especificado.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
```

```
--metadata '{"DBClusterIdentifier":"backup-pitr-test","Engine":"aurora-
mysql","RestoreToTime":"2023-09-01T17:00:00.000Z"}'
```

El siguiente ejemplo muestra cómo restaurar un clúster de base de datos al último momento restaurable.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
--metadata '{"DBClusterIdentifier":"backup-pitr-latest","Engine":"aurora-
mysql","UseLatestRestorableTime":"true"}'
```

Una vez restaurado el clúster de base de datos, tiene que agregar la instancia de base de datos principal (de escritura). Para crear la instancia primaria del clúster de bases de datos, llame al comando create-db-instance de AWS CLI. Incluya el nombre del clúster de bases de datos como valor del parámetro `--db-cluster-identifier`.

Eliminación de una instantánea de clúster de base de datos

Puede eliminar las instantáneas de clúster de base de datos administradas por Amazon RDS cuando ya no las necesite.

> **Note**
>
> Para eliminar copias de seguridad administradas por AWS Backup, utilice la consola de AWS Backup. Para obtener más información sobre AWS Backup, consulte la Guía para desarrolladores de AWS Backup.

Eliminación de una instantánea de clúster de base de datos

Puede eliminar una instantánea de clúster de base de datos con la consola, la AWS CLI, o la API de RDS.

Para eliminar una instantánea compartida o pública, debe iniciar sesión en la cuenta de AWS propietaria de la instantánea.

Consola

Para eliminar una instantánea de clúster de base de datos, realice el siguiente procedimiento:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea de clúster de base de datos que desee eliminar.
4. En Actions (Acciones), elija Delete Snapshot (Eliminar instantánea).
5. En la página de confirmación, elija Delete (Eliminar).

AWS CLI

Puede eliminar una instantánea de clúster de base de datos con el comando de la AWS CLI delete-db-snapshot.

Las siguientes opciones se usan para eliminar una instantánea de clúster de base de datos.

- `--db-cluster-snapshot-identifier`: el identificador de la instantánea de clúster de base de datos.

Example

El código siguiente elimina la instantánea de clúster de base de datos `mydbclustersnapshot`.

Para Linux, macOS o:Unix

```
aws rds delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

En:Windows

```
aws rds delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API de RDS

Puede eliminar una instantánea de clúster de base de datos mediante la operación de la API de Amazon RDS DeleteDBclústerSnapshot.

Los siguientes parámetros se usan para eliminar una instantánea de clúster de base de datos.

- `DBClusterSnapshotIdentifier`: el identificador de la instantánea de clúster de base de datos.

Tutorial: Restaurar un clúster de base de datos de Amazon Aurora desde una instantánea de clúster de base de datos

Una situación habitual con Amazon Aurora es tener una instancia de base de datos con la que se trabaja de vez en cuando pero que no se necesita todo el tiempo. Por ejemplo, puede usar un clúster de base de datos para almacenar los datos de un informe que se ejecuta solo trimestralmente. Una forma de ahorrar dinero en este caso es tomar una instantánea de clúster de base de datos una vez completado el informe. A continuación, elimine el clúster de base de datos y restáurelo cuando necesite cargar datos nuevos y ejecutar el informe en el siguiente trimestre.

Al restaurar un clúster de base de datos, debe indicar el nombre de la instantánea del clúster de base de datos desde la que se restaura. A continuación, proporcione un nombre para el nuevo clúster de base de datos que se cree a partir de la restauración. Para obtener información detallada sobre cómo restaurar clústeres de base de datos desde instantáneas, consulte Restauración de una instantánea de clúster de base de datos.

En este tutorial, también actualizaremos el clúster de base de datos restaurado de Aurora MySQL versión 2 (compatible con MySQL 5.7) a la versión 3 de Aurora MySQL (compatible con MySQL 8.0).

Restaure un clúster de base de datos a un momento específico a partir de una instantánea de clúster de base de datos mediante la consola de Amazon RDS o la AWS CLI.

Para obtener más información sobre la administración de claves de AWS KMS para Amazon RDS, consulte Administración de AWS KMS key.

Temas

- Tutorial: Restauración de un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante la consola de Amazon RDS
- Tutorial: Restauración de un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante AWS CLI

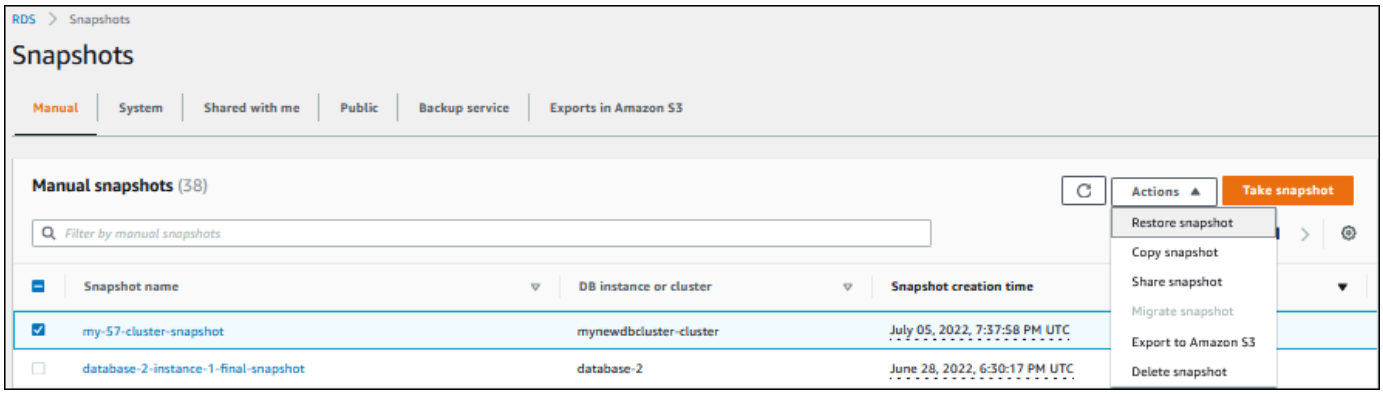
Tutorial: Restauración de un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante la consola de Amazon RDS

En este tutorial, restaurará un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante la consola de Amazon RDS. Al restaurar un clúster de base de datos desde una instantánea mediante la Consola de administración de AWS, también se crea la instancia de base de datos principal (escritor).

> **Note**
>
> Mientras se crea la instancia de base de datos principal, aparece como instancia de lector, pero tras su creación es una instancia de escritor.

Para restaurar un clúster de base de datos desde una instantánea de clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Snapshots (Instantáneas).
3. Elija la instantánea de clúster de base de datos desde la que desea restaurar.
4. En Acciones, elija Restaurar instantánea.

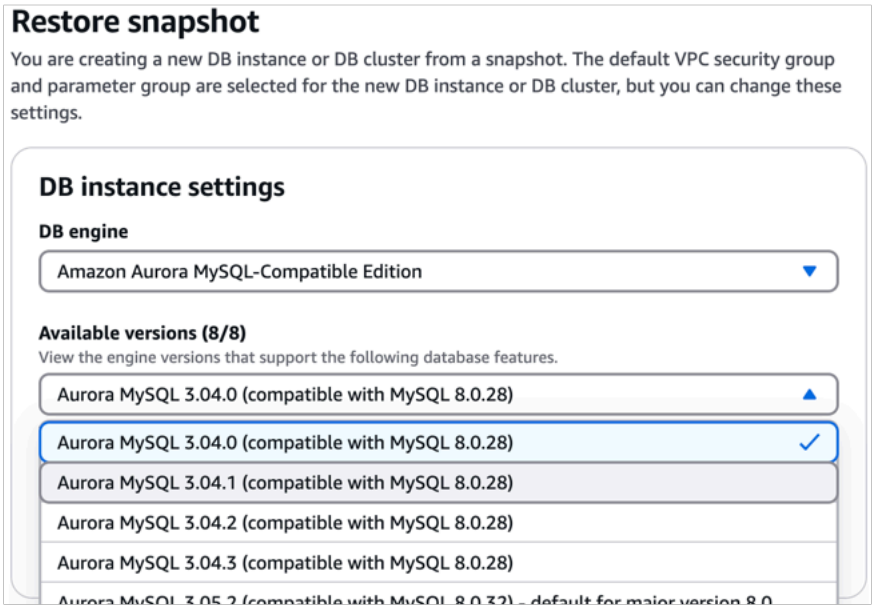


Aparecerá la página Restaurar instantánea.

5. En DB instance settings (Configuración de la instancia de base de datos), realice lo siguiente:

   a. Usar la configuración predeterminada para Motor de base de datos.

b. Para Versiones disponibles, elija una versión compatible con MySQL 8.0, como Aurora MySQL 3.04.0 (compatible con MySQL 8.0.28).



6. En Configuración, en Identificador de instancias de bases de datos, ingrese el nombre único que quiere usar para la instancia de base de datos restaurada; por ejemplo, **`my-80`**.

> **Note**
>
> Para crear el identificador del clúster de base de datos, Amazon RDS adjunta `-cluster` al identificador de instancia de base de datos que especifique.

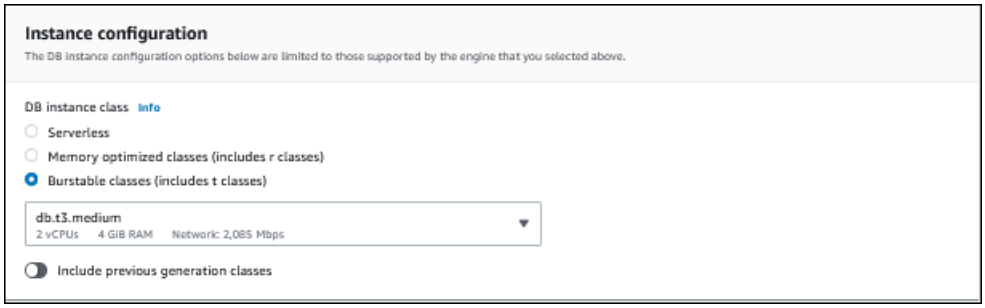
7. En Conectividad, utilice la configuración predeterminada en los siguientes casos:

   - Nube privada virtual (VPC)
   - Grupo de subredes de base de datos
   - Acceso público
   - Grupo de seguridad de VPC (firewall)

8. Elija la clase de instancia de base de datos.

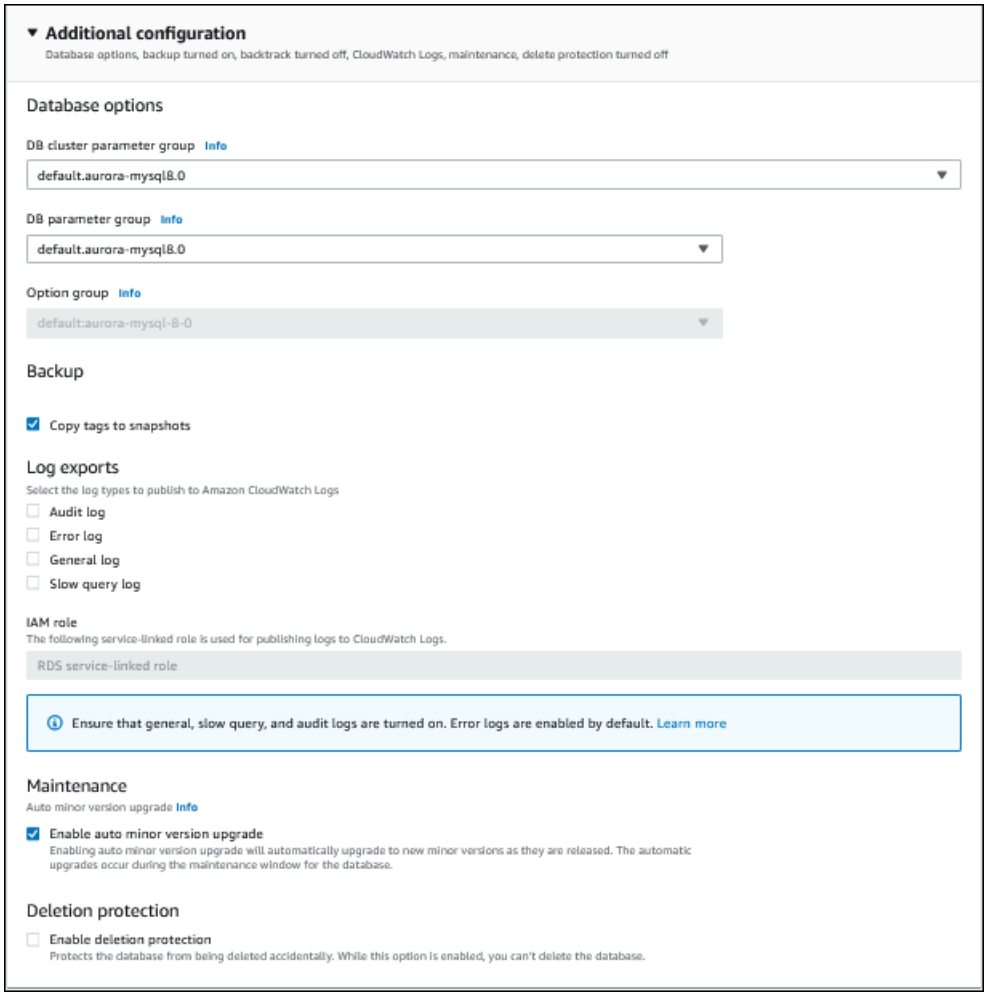
   En este tutorial, elija Clases por ráfagas (incluye clases t) y, a continuación, elija db.t3.medium.

> **Note**
>
> Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.



9. En Database authentication (Autenticación de bases de datos), utilice la configuración predeterminada.
10. En Cifrado, use la configuración predeterminada.

    Si el clúster de base de datos de origen de la instantánea se cifró, el clúster de base de datos restaurado también se cifra. No puede hacerlo sin cifrar.
11. Amplíe Additional configuration (Configuración adicional) en la parte inferior de la página.



12. Realice los siguientes cambios:

    a. En este tutorial, utilice el valor predeterminado para DB clúster parameter group (Grupo de parámetros del clúster de base de datos).

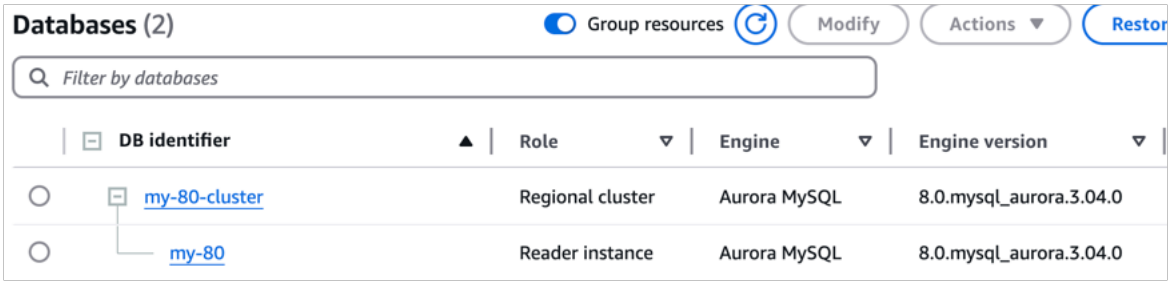
    b. En este tutorial, utilice el valor predeterminado para DB parameter group (Grupo de parámetros de base de datos).

    c. En Log exports (Exportaciones de registros), seleccione todas las casillas de verificación.

    d. En Deletion protection (Protección contra eliminación), marque la casilla de verificación Enable deletion protection (Habilitar la protección contra eliminación).

13. Elija Restaurar instancia de base de datos.

La página Databases (Bases de datos) muestra la instancia de base de datos restaurada, con el estado `Creating`.



Mientras se crea la instancia de base de datos principal, aparece como instancia de lector, pero tras su creación es una instancia de escritor.

Tutorial: Restauración de un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante AWS CLI

En este tutorial, restaurará un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante la AWS CLI. Restauración de un clúster de base de datos a partir de una instantánea de clúster de base de datos mediante AWS CLI

1. Restauración de un clúster de bases de datos utilizando el comando restore-db-clúster-from-snapshot
2. Creación de la instancia de base de datos principal (escritora) utilizando el comando create-db-instance

Restauración de un clúster de bases de datos

Utilice el comando `restore-db-cluster-from-snapshot`. Se requieren las siguientes opciones:

- `--db-cluster-identifier`: nombre del clúster de base de datos restaurado.
- `--snapshot-identifier`: nombre de la instantánea de base de datos desde la que se va a restaurar.
- `--engine`: motor de base de datos del clúster de base de datos restaurado. Debe ser compatible con el motor de base de datos del clúster de base de datos de origen.

  Las opciones son las siguientes:

  - `aurora-mysql`: Aurora compatible con MySQL 5.7 y 8.0
  - `aurora-postgresql`: Aurora compatible con PostgreSQL.

  En este ejemplo, usaremos `aurora-mysql`.

- `--engine-version`: versión del clúster de base de datos restaurado. En este ejemplo, utilizamos una versión compatible con MySQL 8.0.

El siguiente ejemplo restaura un clúster de base de datos de Aurora compatible con MySQL 8.0 llamado `my-new-80-cluster` desde una instantánea de clúster de base de datos denominada `my-57-cluster-snapshot`.

Para restaurar un clúster de bases de datos

- Utilice uno de los siguientes comandos.

  Para Linux, macOS o:Unix

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-80-cluster \
    --snapshot-identifier my-57-cluster-snapshot \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.02.0
```

  En:Windows

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier my-new-80-cluster ^
    --snapshot-identifier my-57-cluster-snapshot ^
    --engine aurora-mysql ^
    --engine-version 8.0.mysql_aurora.3.02.0
```

La salida se parece a la siguiente.

```
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "eu-central-1b",
            "eu-central-1c",
            "eu-central-1a"
        ],
        "BackupRetentionPeriod": 14,
        "DatabaseName": "",
        "DBClusterIdentifier": "my-new-80-cluster",
```

```
        "DBClusterParameterGroup": "default.aurora-mysql8.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "my-new-80-cluster.cluster-############.eu-
central-1.rds.amazonaws.com",
        "ReaderEndpoint": "my-new-80-cluster.cluster-ro-############.eu-
central-1.rds.amazonaws.com",
        "MultiAZ": false,
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "Port": 3306,
        "MasterUsername": "admin",
        "PreferredBackupWindow": "01:55-02:25",
        "PreferredMaintenanceWindow": "thu:21:14-thu:21:44",
        "ReadReplicaIdentifiers": [],
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "HostedZoneId": "Z1RLNU0EXAMPLE",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:123456789012:key/#######-5ccc-49cc-8aaa-
############",
        "DbClusterResourceId": "cluster-ZZ12345678ITSJUSTANEXAMPLE",
        "DBClusterArn": "arn:aws:rds:eu-central-1:123456789012:cluster:my-new-80-
cluster",
        "AssociatedRoles": [],
        "IAMDatabaseAuthenticationEnabled": false,
        "ClusterCreateTime": "2022-07-05T20:45:42.171000+00:00",
        "EngineMode": "provisioned",
        "DeletionProtection": false,
        "HttpEndpointEnabled": false,
        "CopyTagsToSnapshot": false,
        "CrossAccountClone": false,
        "DomainMemberships": [],
        "TagList": []
    }
}
```

Creación de la instancia de base de datos principal (escritora)

Para crear la instancia de base de datos principal (escritora), utilice el comando `create-db-instance`. Se requieren las siguientes opciones:

- `--db-cluster-identifier`: nombre del clúster de base de datos restaurado.
- `--db-instance-identifier`: nombre de la instancia de base de datos principal.
- `--db-instance-class`: clase de instancia de la instancia de base de datos principal. En este ejemplo, usaremos `db.t3.medium`.

  > **Note**
  >
  > Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.

- `--engine`: motor de base de datos de la instancia de base de datos principal. Debe ser el mismo motor de base de datos que utiliza el clúster de base de datos restaurado.

  Las opciones son las siguientes:

  - `aurora-mysql`: Aurora compatible con MySQL 5.7 y 8.0
  - `aurora-postgresql`: Aurora compatible con PostgreSQL.

  En este ejemplo, usaremos `aurora-mysql`.

En el siguiente ejemplo se crea una instancia de base de datos principal (escritora) llamada `my-new-80-cluster-instance` en el clúster de base de datos de Aurora compatible con MySQL 8.0 denominado `my-new-80-cluster`.

Para crear la instancia de base de datos principal

- Utilice uno de los siguientes comandos.

  Para Linux, macOS o:Unix

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-80-cluster \
```

```
    --db-instance-identifier my-new-80-cluster-instance \
    --db-instance-class db.t3.medium \
    --engine aurora-mysql
```

En:Windows

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-80-cluster ^
    --db-instance-identifier my-new-80-cluster-instance ^
    --db-instance-class db.t3.medium ^
    --engine aurora-mysql
```

La salida se parece a la siguiente.


```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-new-80-cluster-instance",
        "DBInstanceClass": "db.t3.medium",
        "Engine": "aurora-mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 1,
        "PreferredBackupWindow": "01:55-02:25",
        "BackupRetentionPeriod": 14,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.aurora-mysql8.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2305ca49",
            "SubnetGroupStatus": "Complete",
```

```
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1a"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1b"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1c"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sat:02:41-sat:03:11",
        "PendingModifiedValues": {},
        "MultiAZ": false,
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "general-public-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:aurora-mysql-8-0",
                "Status": "in-sync"
            }
        ],
        "PubliclyAccessible": false,
        "StorageType": "aurora",
        "DbInstancePort": 0,
        "DBClusterIdentifier": "my-new-80-cluster",
```

```
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:534026745191:key/#######-5ccc-49cc-8aaa-
############",
        "DbiResourceId": "db-5C6UT5PU0YETANOTHEREXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:eu-central-1:123456789012:db:my-new-80-cluster-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": [],
        "TagList": []
    }
}
```

Supervisión de métricas en un clúster de Amazon Aurora

Amazon Aurora utiliza un clúster de servidores de base de datos replicados. Monitorear un clúster de Aurora suele precisar de una comprobación del estado de varias instancias de base de datos. Es posible que las instancias tengan roles especializados, controlen, en su mayoría, operaciones de escritura, de solo lectura o una combinación. También supervisa el estado general del clúster al medir el retraso de replicación. Es la cantidad de tiempo para que los cambios realizados por una instancia de base de datos estén disponibles para las otras instancias.

Temas

- Plan de monitoreo
- Referencia de rendimiento
- Directrices de rendimiento
- Supervisión de herramientas de Amazon Aurora
- Visualización del estado del clúster
- Recomendaciones para Amazon Aurora
- Consulta de métricas en la consola de Amazon RDS
- Visualización de las métricas combinadas en la consola de Amazon RDS
- Supervisión de métricas de Amazon Aurora con Amazon CloudWatch
- Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch
- Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora
- Análisis de anomalías de rendimiento de Aurora con Amazon DevOps Guru para Amazon RDS
- Supervisión de las métricas del sistema operativo con Supervisión mejorada
- Referencia de métricas para Amazon Aurora

Plan de monitoreo

Antes de comenzar la monitorización Amazon Aurora, cree un plan de monitorización. El plan debe responder a las siguientes preguntas:

- ¿Cuáles son los objetivos de la monitorización?

- ¿Qué recursos va a monitorizar?
- ¿Con qué frecuencia va a monitorizar estos recursos?
- ¿Qué herramientas de monitorización va a utilizar?
- ¿Quién se encargará de realizar las tareas de monitorización?
- ¿Quién debe recibir una notificación cuando surjan problemas?

Referencia de rendimiento

Para lograr sus objetivos de monitoreo, debe establecer una referencia. Para ello, mida el rendimiento bajo distintas condiciones de carga en diferentes momentos en su entorno de Amazon Aurora. Puede monitorear métricas como las siguientes:

- Network throughput
- Conexiones de clientes
- E/S para operaciones de lectura, escritura o metadatos
- Saldos de crédito de ráfagas para sus instancias de base de datos

Le recomendamos que almacene datos históricos de rendimiento para Amazon Aurora. Utilizando los datos almacenados, puede comparar el rendimiento actual frente a las tendencias anteriores. También puede distinguir los patrones de rendimiento normales de las anomalías y diseñar técnicas para solucionar problemas.

Directrices de rendimiento

En general, los valores aceptables para las métricas de rendimiento dependen de lo que hace la aplicación respecto a la referencia. Investigue las variaciones coherentes o de las tendencias con respecto a la referencia. Las siguientes métricas suelen ser la fuente de problemas de rendimiento:

- Consumo elevado de CPU o RAM: unos valores elevados de consumo de CPU o RAM es posible que sean si se ajustan a los objetivos de su aplicación (de rendimiento o simultaneidad, por ejemplo) y son los esperados.
- Consumo de espacio en disco: investigue el consumo de espacio en el disco si el espacio utilizado está por sistema alrededor o por encima del 85 % del espacio total disponible en el disco. Compruebe si es posible eliminar datos de la instancia o archivar los datos en un sistema diferente para liberar espacio.

- Tráfico de red: para el tráfico de red, hable con el administrador de su sistema para saber cuál es el rendimiento esperado para la red de su dominio y para su conexión a Internet. Investigue el tráfico de red si el rendimiento es por sistema inferior al esperado.
- Conexiones a bases de datos: si ve que hay un alto número de conexiones de usuarios además de una reducción en el rendimiento y el tiempo de respuesta de la instancia, valore la posibilidad de restringir las conexiones a las bases de datos. El mejor número de conexiones de usuarios para su instancia de base de datos varía en función de la clase de instancia y de la complejidad de las operaciones que se estén llevando a cabo. Para determinar el número de conexiones a bases de datos, asocie la instancia de base de datos con un grupo de parámetros en el que el parámetro `User Connections` se haya establecido en un valor distinto de 0 (ilimitado). Puede utilizar un grupo de parámetros existente o crear uno nuevo. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.
- Métricas de IOPS: los valores esperados para las métricas de IOPS dependen de la especificación del disco y la configuración del servidor, así que debe usar su referencia para conocer los valores típicos. Investigue si los valores son por sistema diferentes de los de la referencia. Para un rendimiento óptimo de IOPS, asegúrese de que el conjunto de trabajo típico se ajuste a la memoria para minimizar las operaciones de lectura y escritura.

Cuando el rendimiento está fuera del punto de referencia establecido, es posible que tenga que realizar cambios para optimizar la disponibilidad de la base de datos para la carga de trabajo. Por ejemplo, es posible que necesite cambiar la clase de instancia de su instancia de base de datos. O es posible que necesite cambiar el número de instancias de base de datos y réplicas de lectura disponibles para los clientes.

Supervisión de herramientas de Amazon Aurora

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon Aurora y de otras soluciones de AWS. AWS ofrece diversas herramientas de supervisión para vigilar a Amazon Aurora, informar cuando algo no funciona y tomar medidas de manera automática cuando corresponda.

Temas

- Herramientas de monitoreo automatizadas
- Herramientas de monitoreo manuales

Herramientas de monitoreo automatizadas

Le recomendamos que automatice las tareas de supervisión en la medida de lo posible.

Temas

- Estado y recomendaciones del clúster de Amazon Aurora
- Métricas de Amazon CloudWatch para Amazon Aurora
- Supervisión del sistema operativo e Información de rendimiento de Amazon RDS
- Servicios integrados

Estado y recomendaciones del clúster de Amazon Aurora

Puede utilizar las siguientes herramientas automatizadas para vigilar a Amazon Aurora e informar cuando haya algún problema:

- Estado de la instancia de Amazon Aurora: vea los detalles sobre el estado actual de la instancia mediante la consola de Amazon RDS, la AWS CLI o la API de RDS.
- Las Recomendaciones para Amazon Aurora responden a recomendaciones automatizadas para recursos de base de datos, como instancias de base de datos, clústeres de base de datos, y grupos de parámetros de clúster de base de datos. Para obtener más información, consulte Recomendaciones para Amazon Aurora.

Métricas de Amazon CloudWatch para Amazon Aurora

Amazon Aurora se integra con Amazon CloudWatch para proporcionar funciones de supervisión adicionales.

- Amazon CloudWatch: este servicio monitorea sus recursos de AWS y las aplicaciones que ejecuta en AWS en tiempo real. Puede utilizar las siguientes características de Amazon CloudWatch con Amazon Aurora:
  - Métricas de Amazon CloudWatch–Amazon Aurora envía métricas automáticamente a CloudWatch cada minuto para cada base de datos activos. No se cobran cargos adicionales por métricas de Amazon RDS en CloudWatch. Para obtener más información, consulte Métricas de Amazon CloudWatch para Amazon Aurora.
  - Alarmas de Amazon CloudWatch–: puede ver una sola Amazon Auroramétrica durante un periodo de tiempo específico. A continuación, puede realizar una o varias acciones en función del valor de la métrica en relación al umbral establecido.

Supervisión del sistema operativo e Información de rendimiento de Amazon RDS

Puede utilizar las siguientes herramientas automatizadas para supervisar el rendimiento de Amazon Aurora:

- Información sobre rendimiento de Amazon RDS: evalúa la carga en su base de datos y determina cuándo y dónde realizar acciones. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.
- Supervisión mejorada de Amazon RDS: examine métricas en tiempo real para el sistema operativo. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.

Servicios integrados

Los siguientes servicios de AWS se integran con Amazon Aurora:

- Amazon EventBridge: es un bus de eventos sin servidor que facilita la conexión de sus aplicaciones con datos de varios orígenes. Para obtener más información, consulte Supervisión de eventos de Amazon Aurora.

- Registros de Amazon Cloudwatch le ayuda a supervisar, almacenar y acceder a los archivos de registro desde instancias de Amazon Aurora, CloudTrail y otros orígenes. Para obtener más información, consulte Supervisión de archivos de registro de Amazon Aurora.
- AWS CloudTrail captura las llamadas a la API y otros eventos relacionados que realiza la Cuenta de AWS o que se realizan en nombre de esta. Además, entrega los archivos de registro a un bucket de Amazon S3 especificado. Para obtener más información, consulte Supervisión de llamadas a la API de Amazon Aurora en AWS CloudTrail.
- Los Flujos de actividad de la base de datos son una característica de Amazon Aurora que proporciona un flujo casi en tiempo real de la actividad en su de clúster de base de datos de . Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.
- DevOps Guru for RDS es una función de Amazon DevOps Guru que aplica el machine learning a las métricas de Información sobre rendimiento para las bases de datos de Amazon Aurora. Para obtener más información, consulte Análisis de anomalías de rendimiento de Aurora con Amazon DevOps Guru para Amazon RDS.

Herramientas de monitoreo manuales

Tiene que monitorear manualmente aquellos elementos que las alarmas de CloudWatch no cubren. Los paneles de las consolas de Amazon RDS, CloudWatch, AWS Trusted Advisor y otras consolas de AWS proporcionan una vista rápida del entorno de AWS. Es recomendable que también compruebe los archivos de registro de su instancia de base de datos.

- En la consola de Amazon RDS, puede monitorizar los siguientes elementos para sus recursos:
  - Número de conexiones a una instancia de base de datos
  - La cantidad de operaciones de lectura y escritura de una instancia de base de datos
  - La cantidad de almacenamiento que utiliza actualmente una instancia de base de datos
  - La cantidad de memoria y de CPU que se utiliza para una instancia de base de datos
  - La cantidad de tráfico de red de entrada y salida de una instancia de base de datos
- Desde el panel de Trusted Advisor, puede revisar las siguientes comprobaciones de optimización del costo, seguridad, tolerancia a errores y mejora del rendimiento:
  - Amazon RDS Idle DB Instances
  - Amazon RDS Security Group Access Risk
  - Copias de seguridad de Amazon RDS

  - Amazon RDS Multi-AZ
  - Accesibilidad de instancias de base de datos de Aurora

  Para obtener más información acerca de estas comprobaciones, consulte Prácticas recomendadas de Trusted Advisor (verificaciones).
- La página de inicio de CloudWatch muestra:
  - Alarmas y estado actual
  - Gráficos de alarmas y recursos
  - Estado de los servicios

  Además, puede utilizar CloudWatch para hacer lo siguiente:
  - Crear paneles personalizados para supervisar los servicios que le importan.
  - Realizar un gráfico con los datos de las métricas para resolver problemas y descubrir tendencias.
  - Buscar y examinar todas sus métricas de recursos de AWS.
  - Crear y editar las alarmas de notificación de problemas.

Visualización del estado del clúster

Con la consola de Amazon RDS, puede acceder rápidamente al estado de su clúster de base de datos.

Temas

- Visualización de un clúster de base de datos de Amazon Aurora
- Ver el estado del clúster de base de datos
- Visualización del

Visualización de un clúster de base de datos de Amazon Aurora

Dispone de varias opciones para ver información acerca de los clústeres de base de datos de Amazon Aurora y de las instancias de bases de datos que contienen.

- Puede ver clústeres e instancias de base de datos en la consola de Amazon RDS eligiendo Databases (Bases de datos) en el panel de navegación.
- Puede obtener información de los clústeres e instancias de base de datos con la AWS Command Line Interface (AWS CLI).
- Puede obtener información de los clústeres e instancias de base de datos con la API de Amazon RDS.

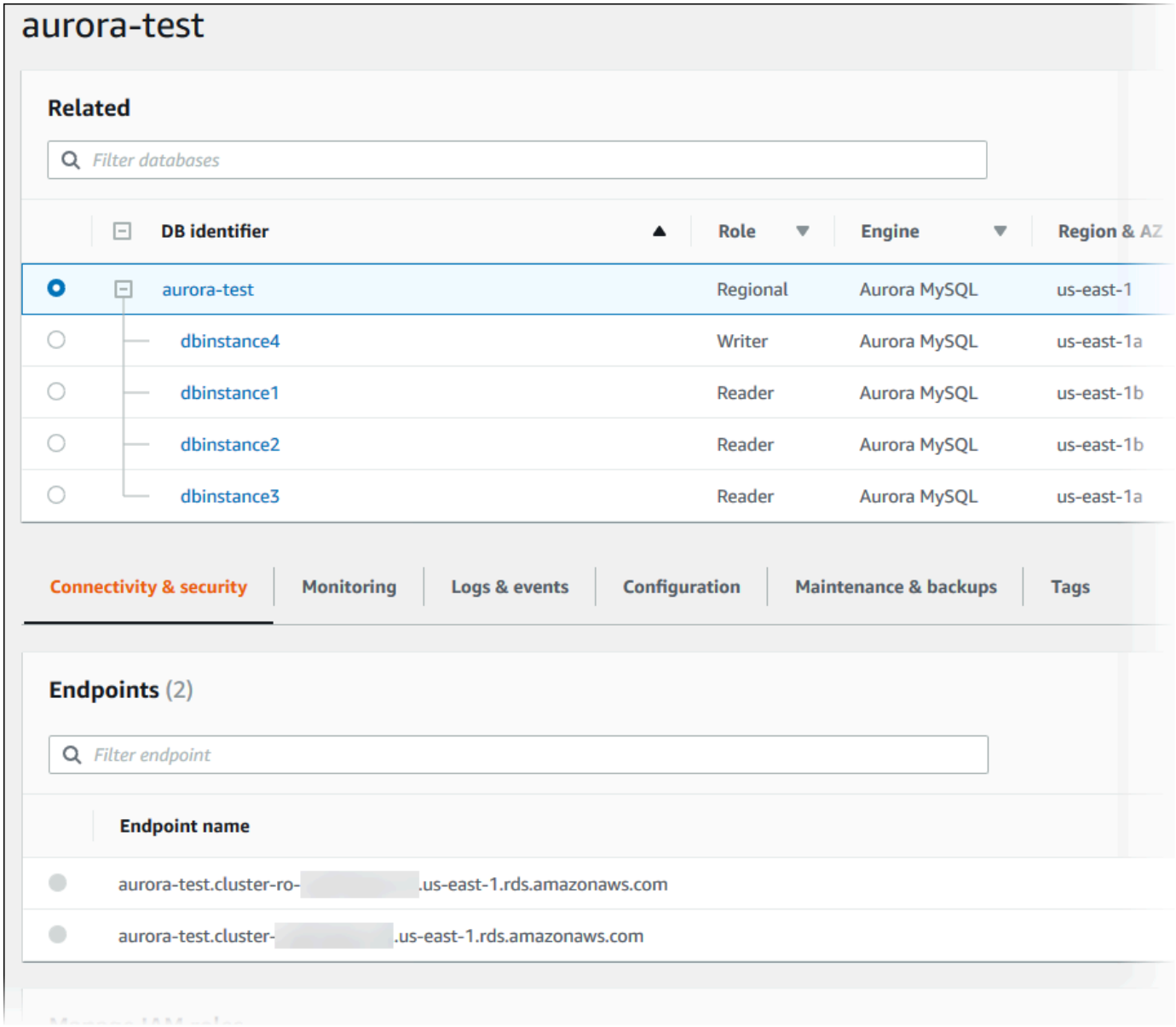
Consola

En la consola de Amazon RDS, puede ver la información sobre un clúster de base de datos si elige Bases de datos desde el panel de navegación de la consola. También puede ver los detalles sobre las instancias de base de datos que son miembros de un clúster de base de datos de Amazon Aurora.

Para ver o modificar clústeres de bases de datos en la consola de Amazon RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Bases de datos.
3. Elija el nombre del clúster de base de datos de Aurora que desea ver en la lista.

   Por ejemplo, la siguiente imagen muestra la página de detalles para el clúster llamado `aurora-test`. El clúster de base de datos tiene cuatro instancias de base de datos mostradas en la lista DB identifier (identificador de base de datos). La instancia de base de datos del escritor, `dbinstance4`, es la instancia de base de datos principal para el clúster de base de datos.



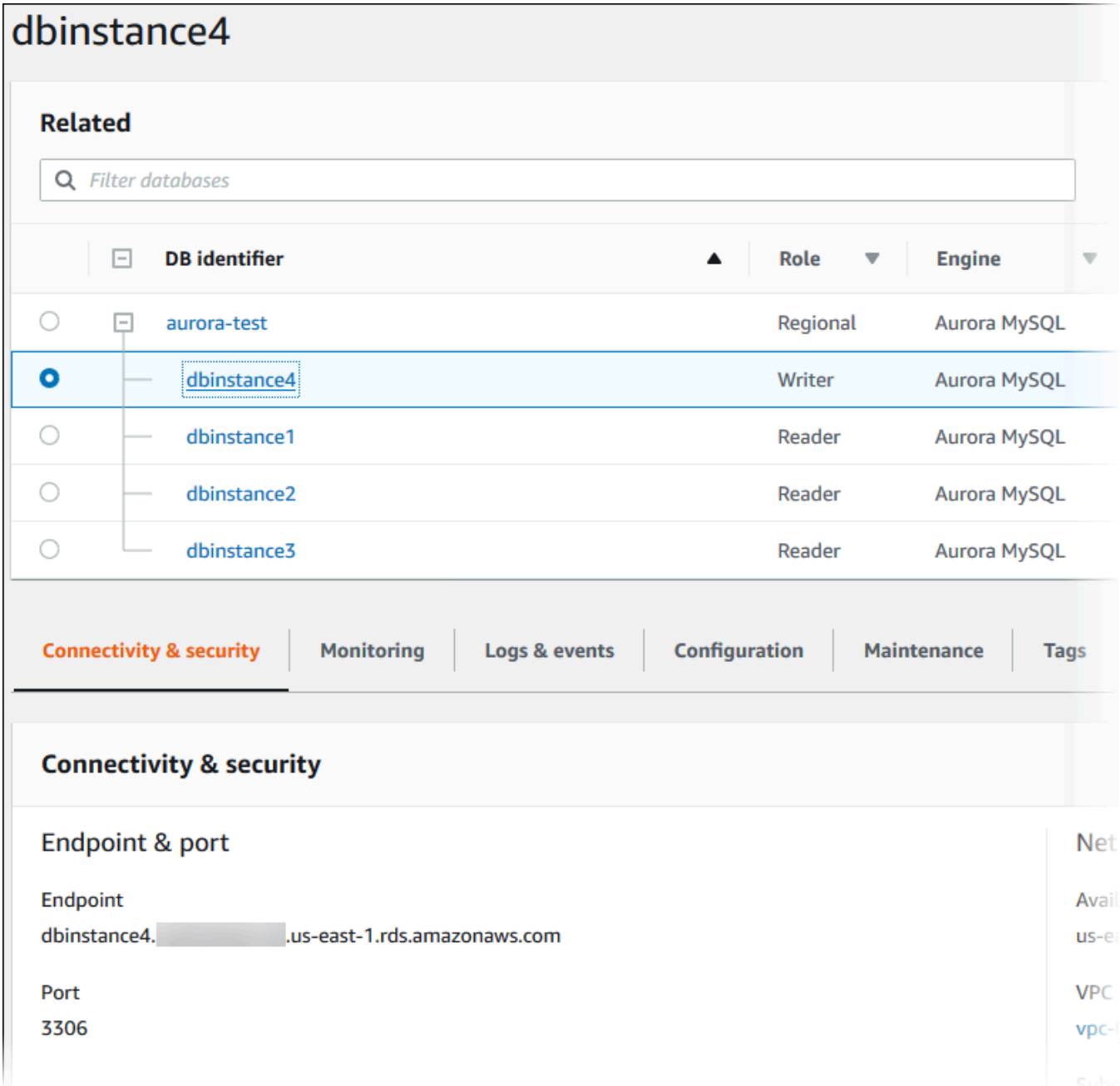
4. Para modificar un clúster de base de datos, selecciónelo en la lista y elija Modify (Modificar).

Para ver o modificar instancias de base de datos de un clúster de base de datos en la consola de Amazon RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Realice una de las siguientes acciones:

- Para ver una instancia de base de datos, elija una en la lista que sea miembro del clúster de base de datos de Aurora.

  Por ejemplo, si elige el identificador de instancias de bases de datos `dbinstance4`, la consola muestra la página de detalles de la instancia de base de datos `dbinstance4`, como se muestra en la imagen siguiente.



- Para modificar una instancia de base de datos, elíjala en la lista y seleccione Modify (Modificar). Para obtener más información sobre la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

AWS CLI

Para ver información de un clúster de base de datos con la AWS CLI utilice el comando describe-db-clusters. Por ejemplo, el comando de la AWS CLI siguiente muestra información del clúster de base de datos para todos los clústeres de base de datos de la región `us-east-1` de modificación para la cuenta de AWS configurada.

```
aws rds describe-db-clusters --region us-east-1
```

Si la AWS CLI está configurada para salida JSON, el comando devuelve lo siguiente.

```
{
    "DBClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "sample-cluster1.cluster-123456789012.us-
east-1.rds.amazonaws.com"
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "sample-cluster1",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T03:35:42.563Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-replica"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-primary"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "03:34-04:04",
            "VpcSecurityGroups": [
```

```
                {
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-ddb65fec"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:06:08.903Z",
            "PreferredMaintenanceWindow": "wed:08:15-wed:08:45"
        },
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "aurora-sample.cluster-123456789012.us-
east-1.rds.amazonaws.com",
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "aurora-sample-cluster",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T10:21:34.826Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-replica-sample"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-sample"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "10:20-10:50",
            "VpcSecurityGroups": [
                {
```

```
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-55da224b"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:00:11.491Z",
            "PreferredMaintenanceWindow": "sun:03:53-sun:04:23"
        }
    ]
}
```

API de RDS

Para ver información de un clúster de base de datos con la API de Amazon RDS, utilice la operación DescribeDBClusters.

Ver el estado del clúster de base de datos

El estado de un clúster de base de datos indica su estado de funcionamiento actual. Puede ver el estado de un clúster de base de datos y las instancias del clúster en la consola de Amazon RDS, la AWS CLI o la API.

> **Note**
>
> Aurora también usa otro estado llamado estado de mantenimiento, que se muestra en la columna Mantenimiento de la consola de Amazon RDS. Este valor indica el estado de los parches de mantenimiento que se deben aplicar a un clúster de base de datos. El estado de mantenimiento es independiente del estado del clúster de base de datos. Para obtener más información sobre el estado de mantenimiento, consulte Aplicación de actualizaciones a un clúster de base de datos.

Encuentre los valores de estado posibles para clústeres de base de datos en la siguiente tabla.

Estado del clúster de base de datos	Facturado	Descripción
disponibles	Facturado	El clúster de base de datos está disponible para su modificación. Cuando un clúster Aurora Serverless está disponible y en pausa, solo se factura el almacenamiento.
Backing-up	Facturado	Se está creando una copia de seguridad del clúster de base de datos.
Backtracking	Facturado	Se están realizando actualmente búsquedas de datos anteriores en el clúster de base de datos. Este estado solo se aplica a Aurora MySQL.
Cloning-failed	No facturado	La clonación de un clúster de base de datos no se ha realizado correctamente.

Estado del clúster de base de datos	Facturado	Descripción
Creación	No facturado	Se está creando el clúster de base de datos. No se puede obtener acceso al clúster de base de datos mientras se está creando.
Eliminación	No facturado	Se está eliminando el clúster de base de datos.
Failing-over	Facturado	Se está realizando una conmutación por error de la instancia principal a una réplica de Aurora.
Inaccessible-encryption-credentials	No facturado	No se puede obtener acceso ni recuperar la AWS KMS key utilizada para cifrar o descifrar el clúster de base de datos.
Inaccessible-encryption-credentials-recoverable	Facturado para almacenamiento	No se puede obtener acceso a la clave KMS utilizada para cifrar o descifrar el clúster de base de datos. Sin embargo, si la clave de KMS está activada, reiniciar el clúster de base de datos puede ayudar a recuperarla. Para obtener más información, consulte Cifrar un clúster de bases de datos de Amazon Aurora.
Mantenimiento	Facturado	Amazon RDS está aplicando una actualización de mantenimiento al clúster de base de datos. Este estado se usa para el mantenimiento de nivel de clúster de base de datos que RDS programa con mucha antelación.
Migrating	Facturado	Se está restaurando una instantánea de clúster de base de datos en un clúster de base de datos.
Migration-failed	No facturado	Una migración no se ha realizado correctamente.

Estado del clúster de base de datos	Facturado	Descripción
Modificar	Facturado	El clúster de base de datos se está modificando porque un cliente ha solicitado su modificación.
Promoting	Facturado	Una réplica de lectura se está promocionando a un clúster de base de datos independiente.
Preparing-data-migration	Facturado	Amazon RDS se está preparando para migrar los datos a Aurora.
Cambio de nombre	Facturado	El nombre del clúster de base de datos se está cambiando porque un cliente lo ha solicitado.
Resetting-master-credentials	Facturado	Las credenciales maestras del clúster de base de datos se están restableciendo porque un cliente lo ha solicitado.
Inicio	Facturado para almacenamiento	El clúster de base de datos se está iniciando.
Stopped	Facturado para almacenamiento	El clúster de base de datos se detiene.
Detención	Facturado para almacenamiento	Se está deteniendo el clúster de base de datos.
Storage-optimization	Facturado	La instancia de base de datos se está modificando para cambiar el tamaño o el tipo de almacenamiento. La instancia de base de datos está totalmente operativa. Sin embargo, mientras su estado sea storage-optimization (Optimización de almacenamiento), no podrá solicitar ningún cambio del almacenamiento de dicha instancia. El proceso de optimización del almacenamiento suele durar poco, pero a veces puede tardar 24 horas o más.

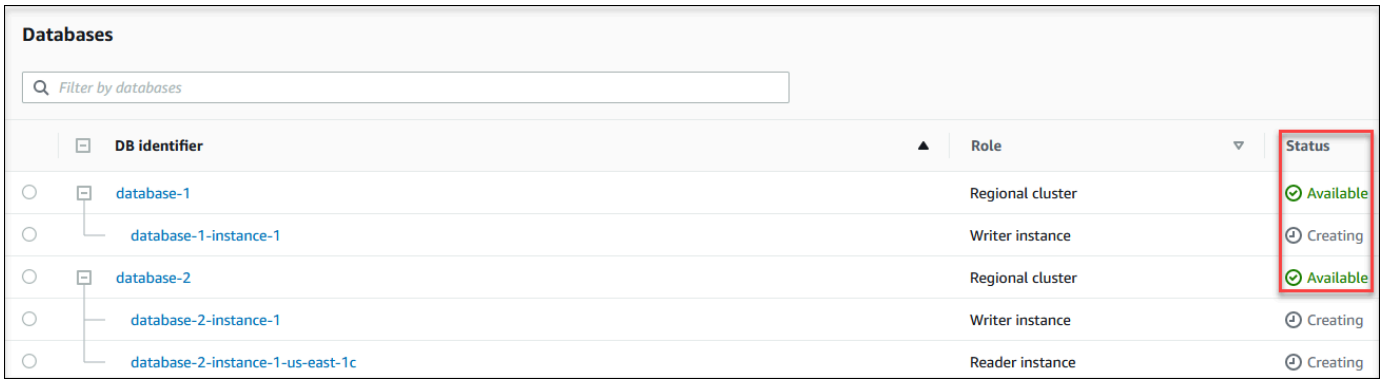
Estado del clúster de base de datos	Facturado	Descripción
Update-iam-db-auth	Facturado	Se está actualizando la autorización de IAM para el clúster de base de datos.
Upgrading	Facturado	Se está actualizando la versión del motor de clúster de base de datos o del sistema operativo.
upgrade_failed	No facturado	Se ha producido un error en el clúster de la base de datos al actualizar a una versión compatible. Aurora crea una instantánea final con el prefijo `rds-final` .

Consola

Para ver el estado de un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).

   Se abre la página Databases (Bases de datos) con la lista de clústeres de base de datos. Para cada clúster de base de datos, se muestra el valor del estado.



CLI

Para ver solo el estado de los clústeres de bases de datos, utilice la siguiente consulta en AWS CLI.

```
aws rds describe-db-clusters --query 'DBClusters[*].[DBClusterIdentifier,Status]' --
output table
```

Visualización del

El estado de una instancia de base de datos en un clúster de Aurora indica su estado de funcionamiento actual. Puede utilizar los siguientes procedimientos para ver el estado de la instancia de base de datos de un clúster en la consola de Amazon RDS, el comando de la AWS CLI o la operación de la API.

> **Note**
>
> Amazon RDS también usa otro estado llamado estado de mantenimiento, que se muestra en la columna Mantenimiento de la consola de Amazon RDS. Este valor indica el estado de los parches de mantenimiento que se deben aplicar a una instancia de base de datos. El estado de mantenimiento es independiente del estado de la instancia de base de datos. Para obtener más información sobre el estado de mantenimiento, consulte Aplicación de actualizaciones a un clúster de base de datos.

Encuentre los valores de estado posibles para instancias de base de datos en la siguiente tabla. Esta tabla le muestra si se le facturará la instancia de base de datos y el almacenamiento, si se le facturará solo el almacenamiento o si no se le facturará. Para todos los estados de instancia de base de datos, se le factura siempre el uso de copia de seguridad.

Estado de la instancia de base de datos	Facturado	Descripción
available	Facturado	La instancia de base de datos está disponible para su modificación.
backing-up	Facturado	Se está creando una copia de seguridad de la instancia de base de datos.
backtracking	Facturado	Se está realizando actualmente búsquedas de datos anteriores en la instancia de base de datos. Este estado solo se aplica a Aurora MySQL.
configuring-enhanced-monitoring	Facturado	La monitorización mejorada se está habilitando o deshabilitando para esta instancia de base de datos.

Estado de la instancia de base de datos	Facturado	Descripción
configuring-iam-database-auth	Facturado	AWS Identity and Access ManagementLa autenticación de base de datos en (IAM) se está habilitando o desactivando para esta instancia de base de datos.
configuring-log-exports	Facturado	La publicación de archivos de registro en Amazon CloudWatch Logs se está habilitando o deshabilitando para esta instancia de base de datos.
converting-to-vpc	Facturado	La instancia de base de datos se está convirtiendo de una instancia de base de datos que no está en una Amazon Virtual Private Cloud (Amazon VPC) a una instancia de base de datos que está en una Amazon VPC.
crear	No facturado (no PITR) Facturado (solo PITR)	La instancia de base de datos se está creando. No se puede obtener acceso a la instancia de base de datos mientras se está creando. Si restaura una base de datos durante una recuperación en un momento dado (PITR), se le facturará mientras la base de datos esté en el estado creación. Este es el único escenario en el que el estado creación incurre en cargos.
delete-precheck	No facturado	Amazon RDS valida que las réplicas de lectura estén seguras para eliminar.
deleting	No facturado	Se está eliminando la instancia de base de datos.
error	No facturado	La instancia de base de datos ha generado un error y Amazon RDS no puede recuperarla. Realice una restauración al último momento restaurable de la instancia de base de datos para recuperar los datos.

Estado de la instancia de base de datos	Facturado	Descripción
inaccessible-encryption-credentials	No facturado	No se puede obtener acceso ni recuperar la AWS KMS key utilizada para cifrar o descifrar la instancia de base de datos.
inaccessible-encryption-credentials-recoverable	Facturado para almacenamiento	No se puede acceder a la clave de KMS utilizada para cifrar o descifrar la instancia de base de datos. Sin embargo, si la clave de KMS está activada, reiniciar la instancia de base de datos puede ayudar a recuperarla. Para obtener más información, consulte Cifrar un clúster de bases de datos de Amazon Aurora.
incompatible-create	No facturado	Amazon RDS está intentando crear una instancia de base de datos, pero no puede hacerlo porque los recursos no son compatibles con la instancia de base de datos. Este estado puede darse si, por ejemplo, el perfil de instancia de su instancia de base de datos no tiene los permisos correctos.
incompatible-network	No facturado	Amazon RDS está intentando realizar una acción de recuperación en una instancia de base de datos, pero no puede hacerlo porque la VPC está en un estado que impide completar la acción. Este estado puede darse si, por ejemplo, todas las direcciones IP disponibles en una subred están en uso y Amazon RDS no puede obtener una dirección IP para la instancia de base de datos.

Estado de la instancia de base de datos	Facturado	Descripción
incompatible-option-group	Facturado	Amazon RDS ha intentado aplicar un cambio en el grupo de opciones, pero no puede hacerlo y no puede revertir al estado anterior del grupo de opciones. Para obtener información, consulte la lista Recent Events (Eventos recientes) para la instancia de base de datos. Este estado puede darse si, por ejemplo, el grupo de opciones contiene una opción como TDE y la instancia de base de datos no contiene información cifrada.
incompatible-parameters	Facturado	Amazon RDS no puede iniciar la instancia de base de datos porque los parámetros especificados en el grupo de parámetros de base de datos de la instancia de base de datos no son compatibles con la instancia de base de datos. Revierta los cambios de los parámetros o hágalos compatibles con la instancia de base de datos para recuperar el acceso a la instancia de base de datos. Para obtener más información acerca de los parámetros incompatibles, consulte la lista Recent Events (Eventos recientes) de la instancia de base de datos.
incompatible-restore	No facturado	Amazon RDS no puede realizar una restauración a un momento dado. Las causas habituales para este estado incluyen el uso de tablas temporales o el uso de tablas de MyISAM con MySQL .

Estado de la instancia de base de datos	Facturado	Descripción
insufficient-capacity	No facturado	Amazon RDS no puede crear la instancia porque actualmente no hay suficiente capacidad disponible. Para crear la instancia de base de datos en la misma zona de disponibilidad con el mismo tipo de instancia , elimine la instancia de base de datos, espere unas horas e intente crear de nuevo. Alternativamente, cree una nueva instancia utilizando una clase de instancia o zona de disponibilidad diferente.
maintenance	Facturado	Amazon RDS está aplicando una actualización de mantenimiento a la instancia de base de datos. Este estado se usa para el mantenimiento de nivel de instancia que RDS programa con mucha antelación.
modifying	Facturado	La instancia de base de datos se está modificando porque un cliente ha solicitado su modificación.
moving-to-vpc	Facturado	La instancia de base de datos se está moviendo a una nueva Amazon Virtual Private Cloud (Amazon VPC).
rebooting	Facturado	La instancia de base de datos se está reiniciando porque un cliente o un proceso de Amazon RDS que requiere el reinicio lo ha solicitado.
resetting-master-credentials	Facturado	Las credenciales maestras de la instancia de base de datos se están restableciendo porque un cliente lo ha solicitado.
renaming	Facturado	El nombre de la instancia de base de datos se está cambiando porque un cliente lo ha solicitado.
restore-error	Facturado	La instancia de base de datos ha registrado un error al intentar restaurar a un momento dado o a partir de una instantánea.

Estado de la instancia de base de datos	Facturado	Descripción
starting	Facturado para almacenamiento	La instancia de base de datos se está iniciando.
stopped	Facturado para almacenamiento	La instancia de base de datos se ha detenido.
deteniendo	Facturado para almacenamiento	La instancia de base de datos se está deteniendo.
storage-config-upgrade	Facturado	Se está actualizando la configuración del sistema de archivos de almacenamiento de la instancia de base de datos. Este estado solo es aplicable a las bases de datos verdes de una implementación azul/verde o a las réplicas de lectura de instancias de base de datos.
storage-full	Facturado	La instancia de base de datos ha alcanzado su asignación de capacidad de almacenamiento. Se trata de un estado crítico y le recomendamos que corrija este problema de inmediato. Para ello, aumente la escala del almacenamiento modificando la instancia de base de datos. Para evitar esta situación, defina las alarmas de Amazon CloudWatch para que se le advierta cuando el espacio de almacenamiento está bajando.
storage-initialization	Facturado	La instancia de base de datos carga bloques de datos de Amazon S3 para optimizar el rendimiento del volumen después de restaurarlo a partir de una instantánea. Sigue disponible para las operaciones, pero es posible que el rendimiento no esté al máximo hasta que se complete la inicialización.

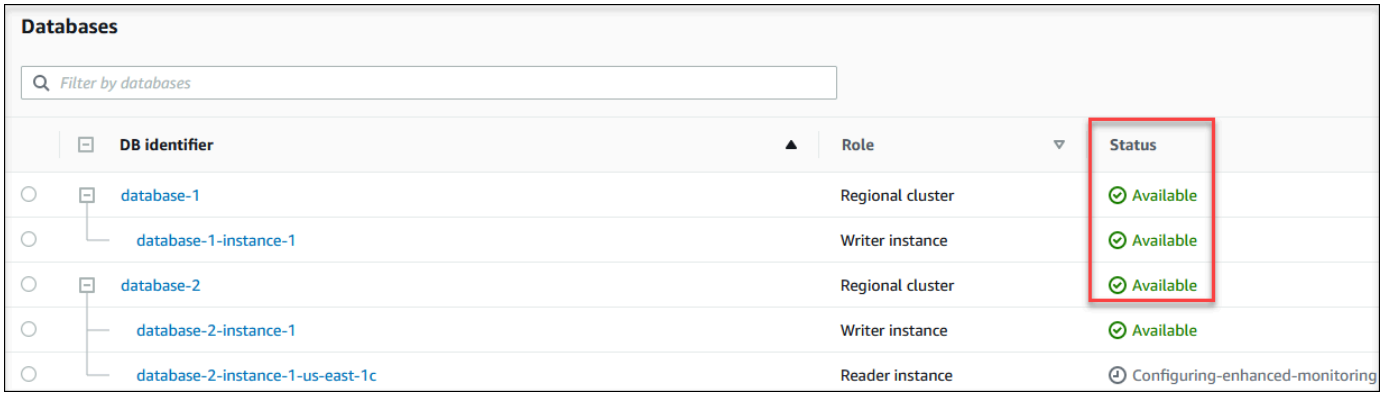
Estado de la instancia de base de datos	Facturado	Descripción
storage-optimization	Facturado	Amazon RDS está optimizando el almacenamiento de su instancia de base de datos. El proceso de optimización del almacenamiento suele durar poco, pero a veces puede tardar 24 horas o más. Durante la optimización del almacenamiento, la instancia de base de datos permanece disponible. La optimización del almacenamiento es un proceso en segundo plano que no afecta a la disponibilidad de la instancia.
upgrading	Facturado	Se está actualizando la versión del motor de base de datos o del sistema operativo.
upgrade_failed	No facturado	Se ha producido un error en la instancia de base de datos al actualizar a una versión compatible. Aurora crea una instantánea final con el prefijo `rds-final` .

Consola

Para ver el estado de una instancia de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).

   Se abre la página Databases (Bases de datos) con la lista de instancias de base de datos. Para cada clúster de base de datos de un clúster, se muestra el valor del estado.



CLI

Para ver la instancia de base de datos y su información de estado usando el AWS CLI, utilice el comando describe-db-instances. Por ejemplo, el siguiente comando AWS CLI enumera toda la información de las instancias de base de datos.

```
aws rds describe-db-instances
```

Para ver una instancia de base de datos específica y su estado, llame al comando describe-db-instances con la siguiente opción:

- `DBInstanceIdentifier`: el nombre de la instancia de base de datos.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

Para ver solo el estado de todas las instancias de bases de datos, utilice la siguiente consulta en AWS CLI.

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

API

Para ver el estado de la instancia de base de datos usando la API de Amazon RDS, llame a la operación DescribeDBInstances.

Recomendaciones para Amazon Aurora

Amazon Aurora ofrece recomendaciones automatizadas para recursos de base de datos, como instancias de base de datos, clústeres de base de datos, y grupos de parámetros de bases de datos. Estas recomendaciones proporcionan instrucciones de las prácticas recomendadas mediante el análisis de la configuración de clúster de base de datos, la configuración de instancia de base de datos, el uso y los datos de rendimiento.

Información de rendimiento de Amazon RDS monitoriza automáticamente métricas específicas y crea umbrales mediante el análisis de qué niveles se consideran potencialmente problemáticos para un recurso específico. Cuando los nuevos valores de las métricas cruzan un umbral predefinido durante un período de tiempo determinado, Información de rendimiento genera una recomendación proactiva. Esta recomendación ayuda a evitar que el rendimiento de la base de datos se vea afectado en el futuro. Por ejemplo, la recomendación “Inactiva en la transacción” se genera para las instancias de Aurora PostgreSQL cuando las sesiones conectadas a la base de datos no están realizando un trabajo activo, pero pueden mantener bloqueados los recursos de la base de datos. Para recibir recomendaciones proactivas, debe activar Información de rendimiento con un período de retención de nivel de pago. Para obtener información acerca de la activación de Información de rendimiento, consulte Activación y desactivación de Información de rendimiento de Aurora. Para obtener información sobre los precios y la retención de datos de Información de rendimiento, consulte Precios y retención de datos de Performance Insights.

DevOps Guru para RDS monitoriza determinadas métricas para detectar cuándo el comportamiento de una métrica se vuelve muy inusual o anómalo. Estas anomalías se presentan como información reactiva con recomendaciones. Por ejemplo, DevOps Guru para RDS podría recomendar que considere aumentar la capacidad de la CPU o investigar los eventos de espera que contribuyen a la carga de la base de datos. DevOps Guru para RDS también proporciona recomendaciones proactivas basadas en umbrales. Para ver estas recomendaciones, debe activar DevOps Guru para RDS. Para obtener información sobre cómo activar DevOps Guru para RDS, consulte Activación de DevOps Guru y especificación de la cobertura de recursos.

Las recomendaciones tendrán uno de los siguientes estados: activas, rechazadas, pendientes o resueltas. Las recomendaciones resueltas están disponibles durante 365 días.

Puede ver o descartar las recomendaciones. Puede aplicar una recomendación activa basada en la configuración de forma inmediata, programarla para el siguiente periodo de mantenimiento o descartarla. Para obtener recomendaciones proactivas basadas en umbrales y reactivas basadas

en machine learning, debe revisar la causa sugerida del problema y, a continuación, realizar las acciones recomendadas para solucionarlo.

Las recomendaciones son compatibles en las siguientes:Regiones de AWS

- Este de EE. UU. (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- Asia-Pacífico (Bombay)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (París)
- Europa (Estocolmo)
- América del Sur (São Paulo)

Aprenda a ver, aplicar, descartar y modificar recomendaciones de Amazon Aurora en las siguientes secciones.

Temas

- Visualización Amazon Aurora de recomendaciones
- Aplicación de recomendaciones para Amazon Aurora
- Descarte de las recomendaciones de Amazon Aurora
- Modificación de las recomendaciones de Amazon Aurora descartadas a recomendaciones activas
- Recomendaciones de la referencia de Amazon Aurora

Visualización Amazon Aurora de recomendaciones

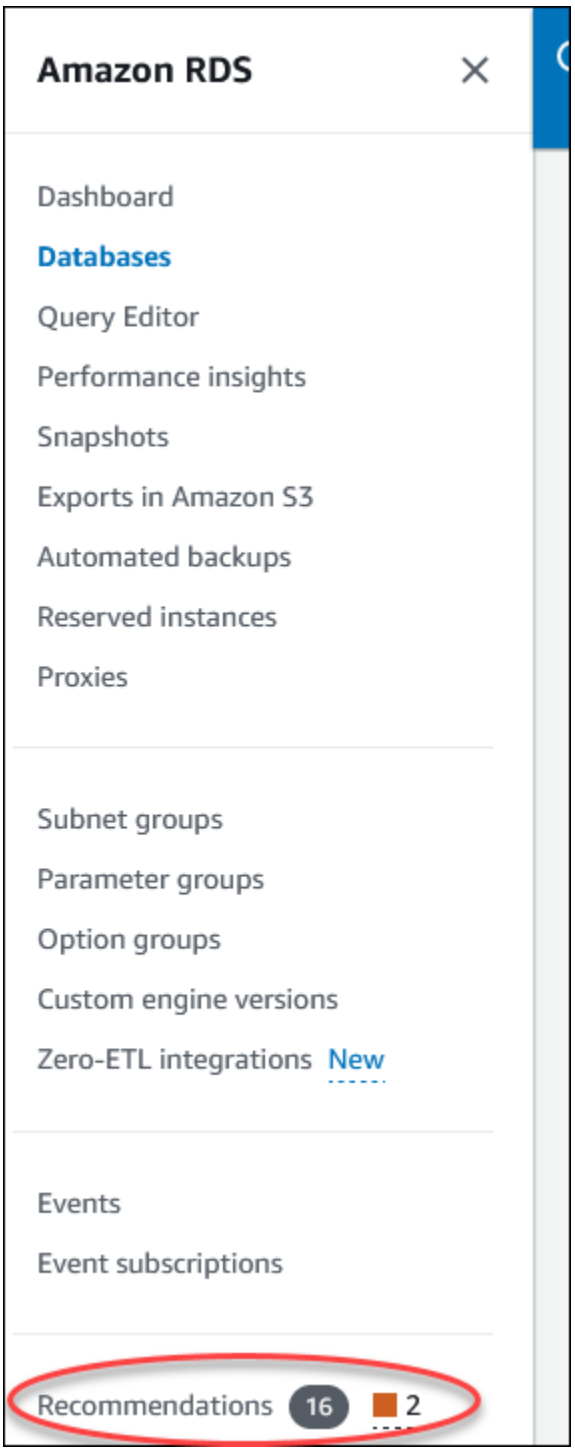
Con la consola de Amazon RDS, puede ver las recomendaciones de Amazon Aurora para los recursos de su base de datos. En el caso de un clúster de base de datos, aparecen las recomendaciones para el clúster de base de datos y sus instancias.

Consola

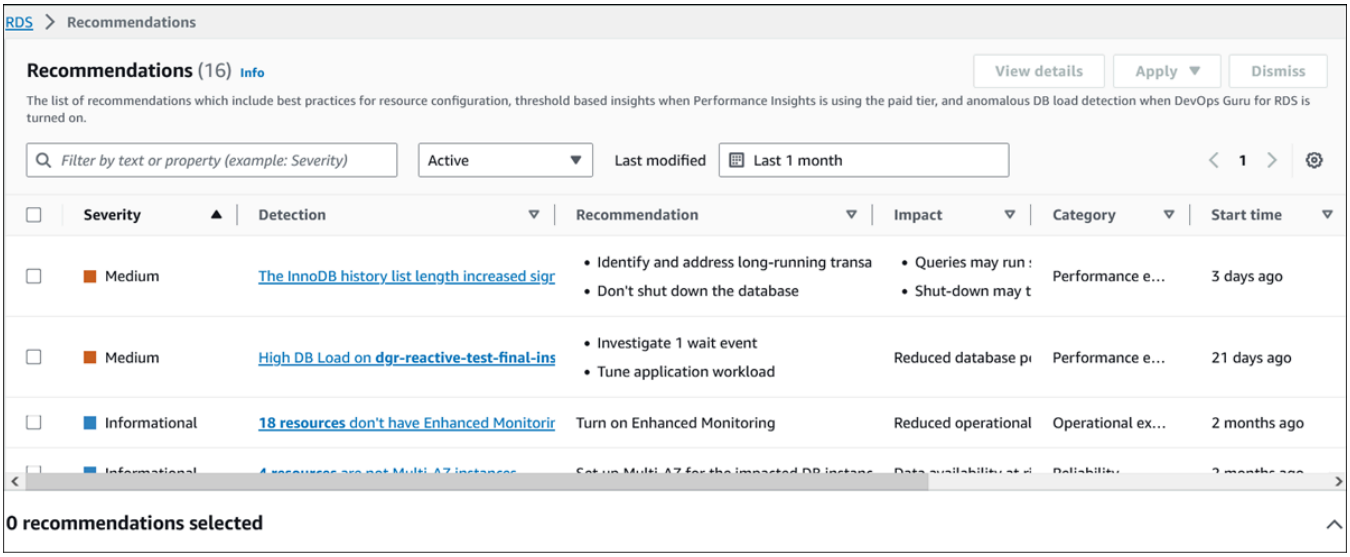
Para ver las recomendaciones de Amazon Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, realice cualquiera de las siguientes acciones:

   - Elija Recomendaciones. El número de recomendaciones activas para sus recursos y el número de recomendaciones con la mayor gravedad generadas en el último mes están disponibles junto a Recomendaciones. Para encontrar el número de recomendaciones activas para cada gravedad, seleccione el número que muestre la gravedad más alta.

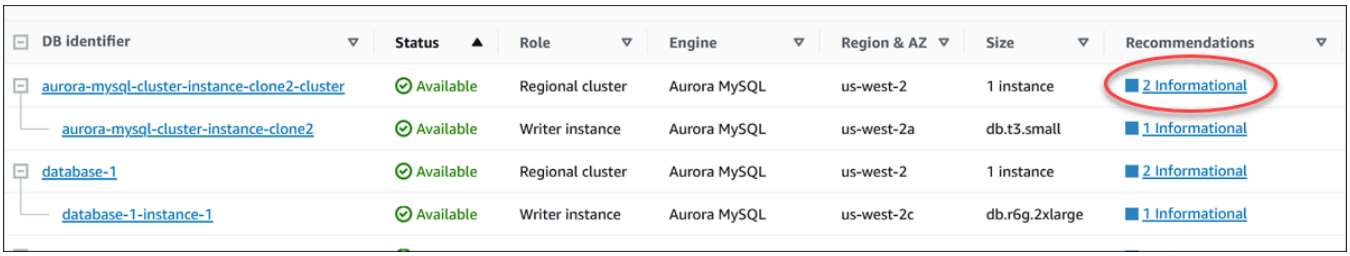


De forma predeterminada, la página de Recomendaciones muestra una lista de las nuevas recomendaciones en el mes pasado. Amazon Aurora ofrece recomendaciones de todos los recursos de su cuenta y las clasifica por gravedad.

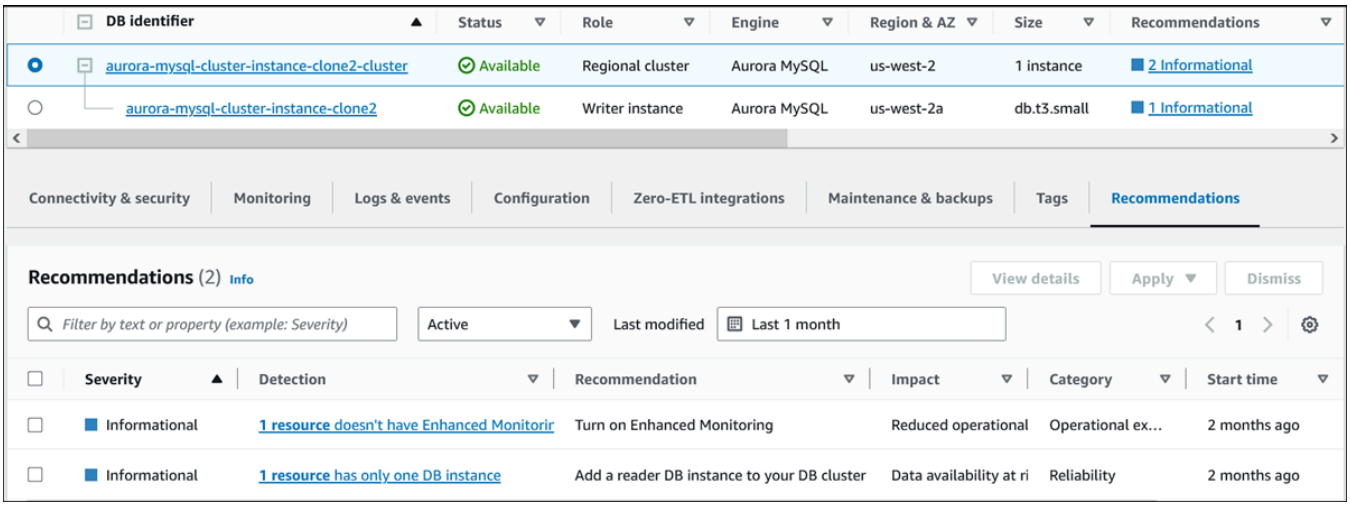


Puede elegir una recomendación para ver una sección en la parte inferior de la página que contiene los recursos afectados y los detalles sobre cómo se aplicará la recomendación.

- En la página Bases de datos, seleccione Recomendaciones para un recurso.



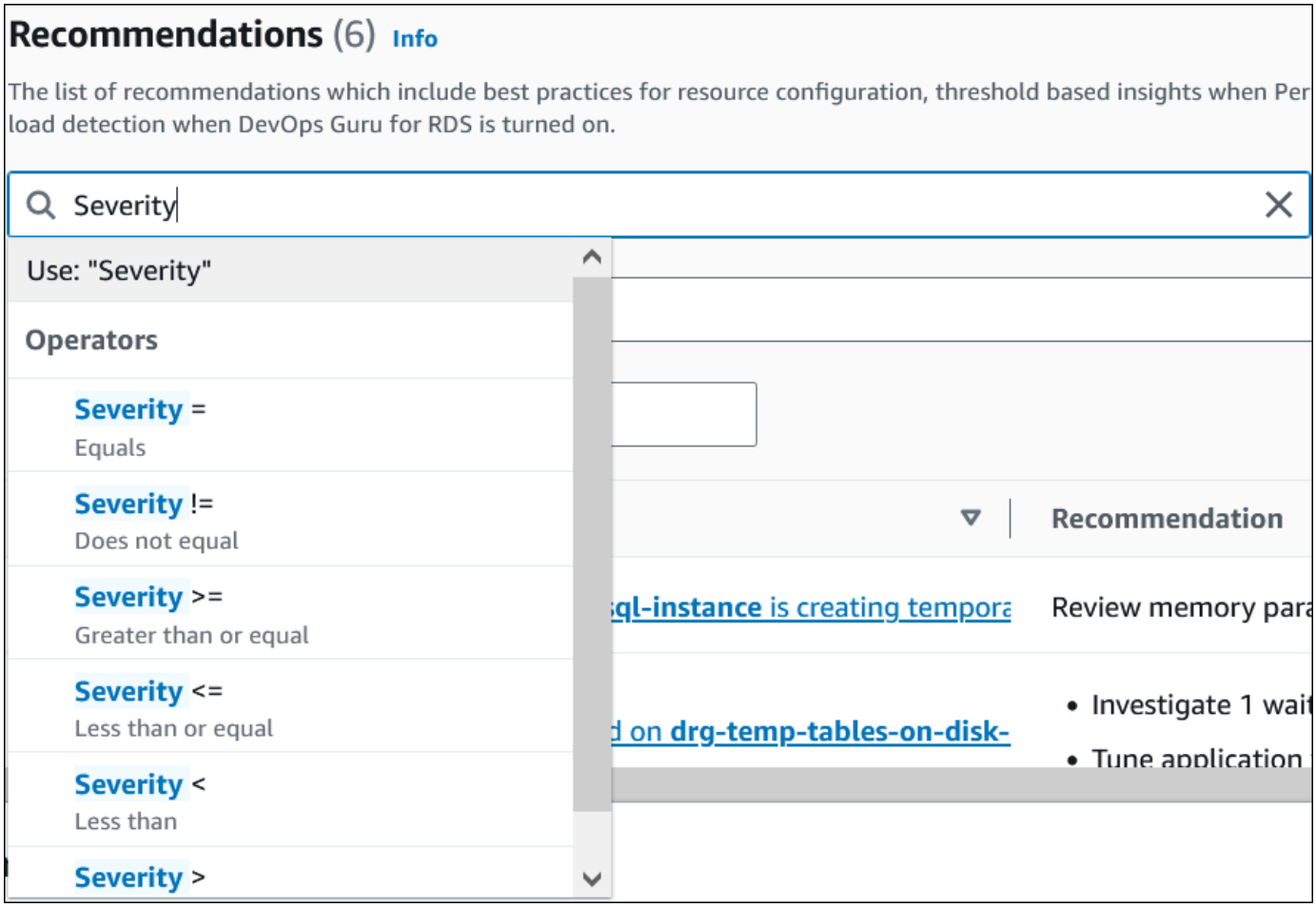
La pestaña Recomendaciones muestra las recomendaciones y sus detalles para el recurso seleccionado.



Están disponibles los siguientes detalles para las recomendaciones:

- Gravedad: el nivel de implicación del problema. Los niveles de gravedad son Alto, Medio, Bajo e Informativo.
- Detección: el número de recursos afectados y una breve descripción del problema. Haga clic en este enlace para ver la recomendación y los detalles del análisis.
- Recomendación: una breve descripción de la acción que se recomienda aplicar.
- Impacto: una breve descripción del posible impacto si no se aplica la recomendación.
- Categoría: el tipo de recomendación. Las categorías son Eficiencia de rendimiento, Seguridad, Fiabilidad, Optimización de costos, Excelencia operativa y Sostenibilidad.
- Estado: el estado actual de la recomendación. Los estados posibles son Todas, Activa, Descartada, Resuelta y Pendiente.
- Hora de inicio: hora a la que comenzó el problema. Por ejemplo, Hace 18 horas.
- Última modificación: la hora en que el sistema actualizó la recomendación por última vez debido a un cambio en la Gravedad, o la hora en que respondiera a la recomendación. Por ejemplo, Hace 10 horas.
- Hora de finalización: hora en la que finalizó el problema. La hora no se mostrará si hay problemas continuos.
- Identificador de recurso: el nombre de uno o más recursos.

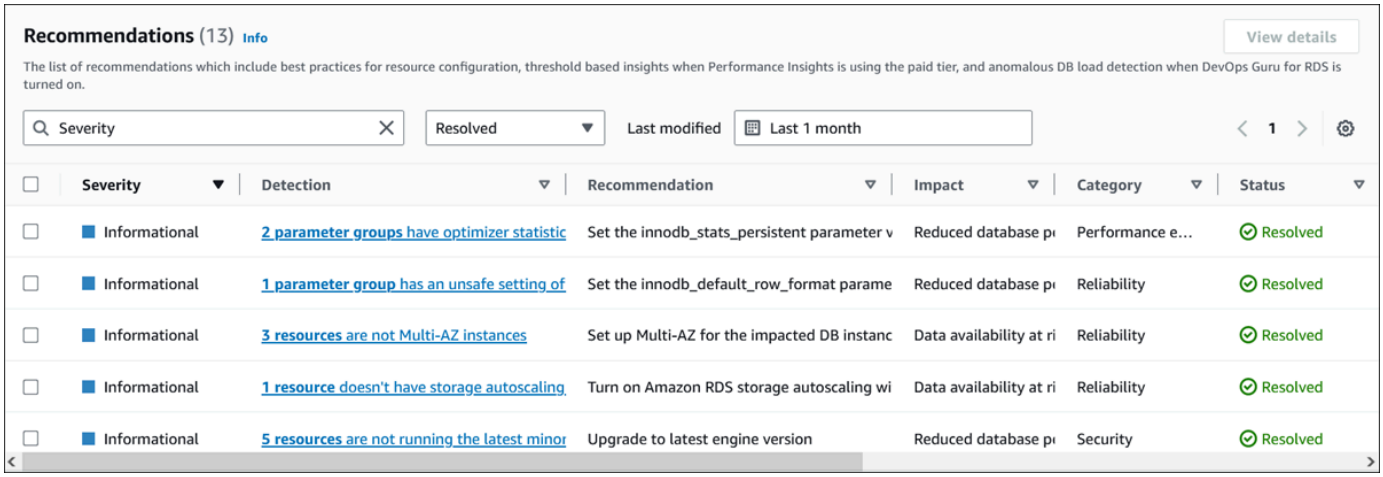
3. (Opcional) Elija los operadores Gravedad o Categoría en el campo para filtrar la lista de recomendaciones.



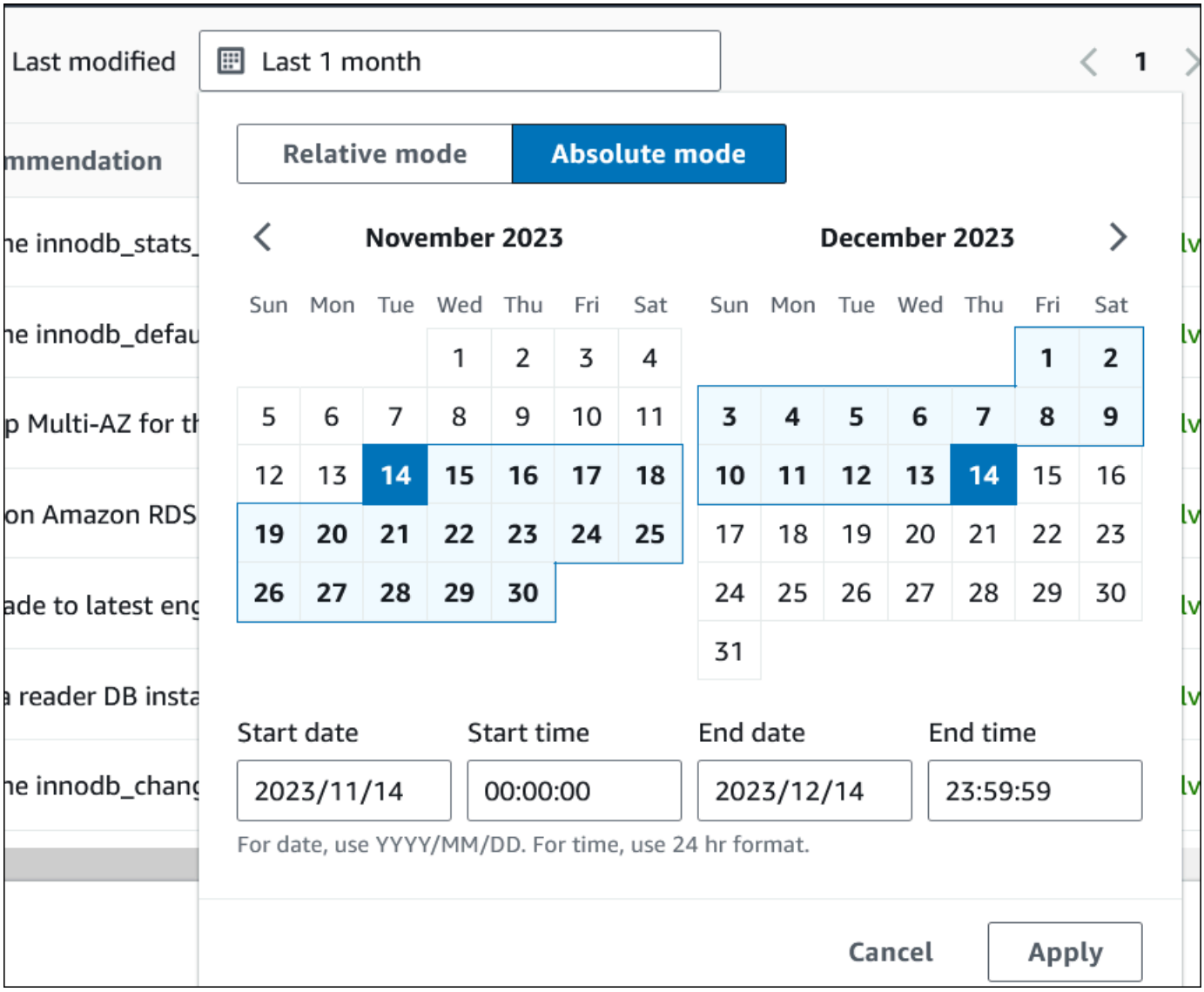
Aparecen las recomendaciones para la operación seleccionada.

4. (Opcional) Elija cualquiera de los siguientes estados de recomendación:

   - Activa: muestra las recomendaciones actuales que puede aplicar, programar para el próximo período de mantenimiento o descartar.
   - Todas: muestra todas las recomendaciones con el estado actual.
   - Descartada: muestra las recomendaciones rechazadas.
   - Resuelta: muestra las recomendaciones que se han resuelto.
   - Pendiente: muestra las recomendaciones cuyas acciones recomendadas están en curso o programadas para el siguiente período de mantenimiento.



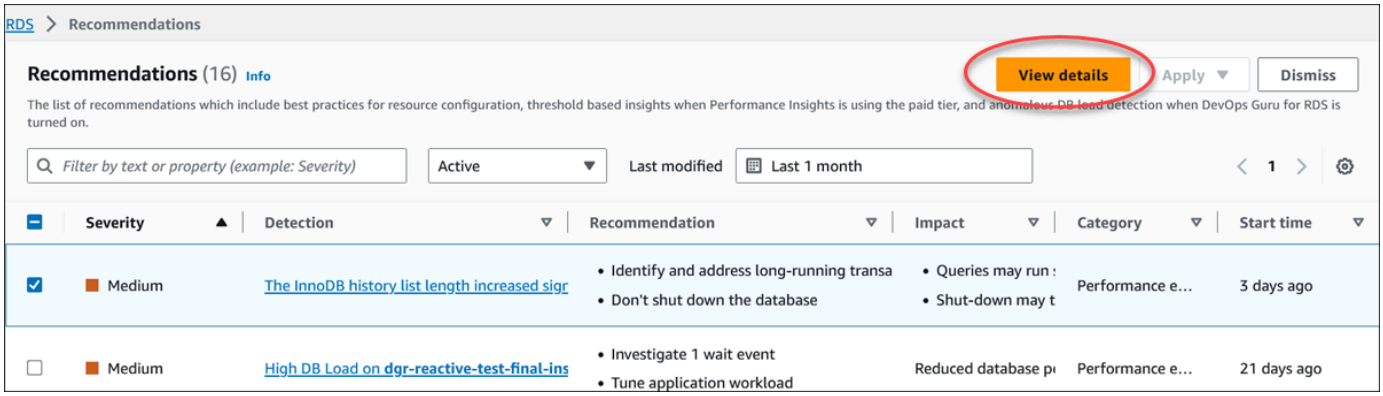
5. (Opcional) Seleccione Modo relativo o Modo absoluto en Última modificación para modificar el periodo de tiempo. La página de Recomendaciones muestra las recomendaciones generadas en el periodo de tiempo. El periodo de tiempo predeterminado es el mes pasado. En el Modo absoluto, puede elegir el período de tiempo o introducir la hora en los campos Fecha de inicio y Fecha de finalización.



Se muestran las recomendaciones para el período de tiempo establecido.

Tenga en cuenta que puede ver todas las recomendaciones de recursos de su cuenta si configura el rango en Todos.

6. (Opcional) Seleccione Preferencias en la parte derecha para personalizar los detalles que se van a mostrar. Puede elegir un tamaño de página, ajustar las líneas del texto y permitir u ocultar las columnas.
7. (Opcional) Elija una recomendación y, a continuación, seleccione Ver detalles.



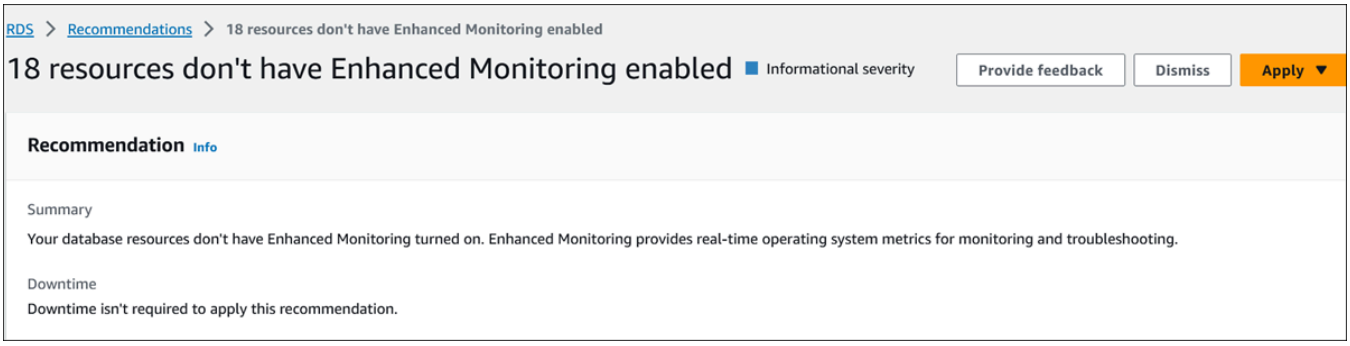
Aparece la página de detalles de la recomendación. En el título se indica el recuento total de los recursos con el problema detectado y su gravedad.

Para obtener información sobre los componentes de la página de detalles de una recomendación reactiva basada en anomalías, consulte Viewing reactive anomalies en la Guía del usuario de Amazon DevOps Guru.

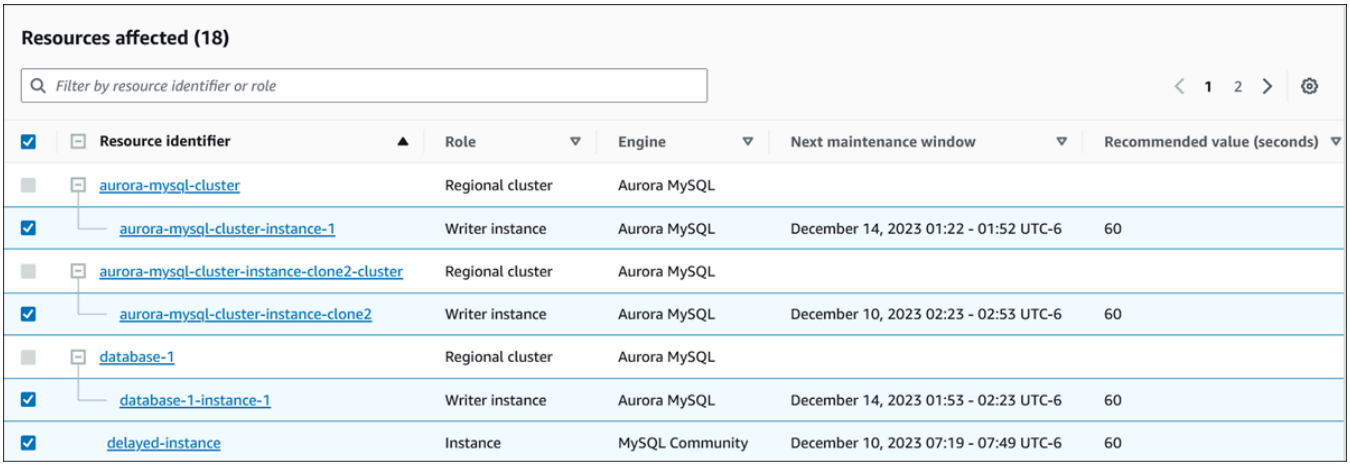
Para obtener información sobre los componentes en la página de detalles de una recomendación proactiva basada en un umbral, consulte Visualización de las recomendaciones proactivas de Información de rendimiento.

Las demás recomendaciones automatizadas muestran los siguientes componentes en la página de detalles de la recomendación:

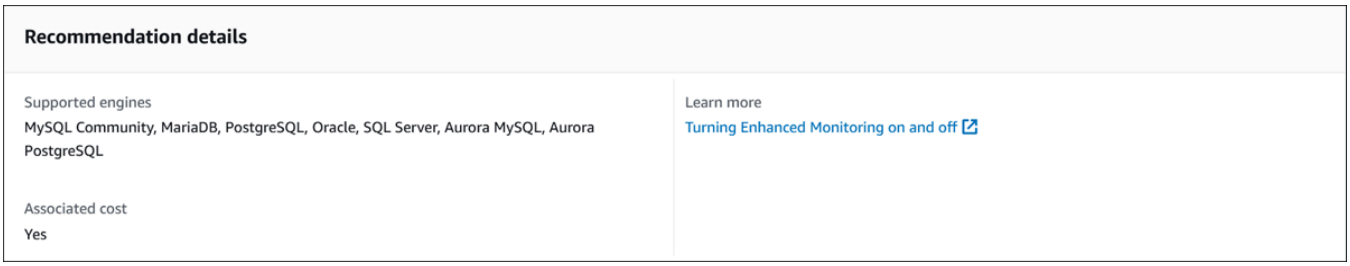
- Recomendación: un resumen de la recomendación y si se requiere un tiempo de inactividad para aplicarla.



- Recursos afectados: detalles de los recursos afectados.



- Detalles de la recomendación: información sobre los motores compatibles, cualquier costo asociado necesario para aplicar la recomendación y enlace a la documentación para obtener más información.



CLI

Para ver las recomendaciones de Amazon RDS sobre las instancias de base de datos o los clústeres de base de datos, utilice el siguiente comando en AWS CLI.

```
aws rds describe-db-recommendations
```

API de RDS

Para ver las recomendaciones de Amazon RDS mediante la API de Amazon RDS, utilice la operación DescribeDBRecommendations.

Aplicación de recomendaciones para Amazon Aurora

Para aplicar las recomendaciones de Amazon Aurora mediante la consola de Amazon RDS, seleccione una recomendación basada en la configuración o un recurso afectado en la página de detalles. A continuación, seleccione si desea aplicar la recomendación inmediatamente o

programarla para el siguiente periodo de mantenimiento. Es posible que el recurso tenga que reiniciarse para que el cambio se aplique. Para obtener algunas recomendaciones sobre grupos de parámetros de bases de datos, es posible que deba reiniciar los recursos.

Las recomendaciones proactivas basadas en umbrales o las reactivas basadas en anomalías no tendrán la opción de aplicarse y es posible que necesiten una revisión adicional.

Consola

Para aplicar una recomendación basada en la configuración

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, realice una de las siguientes acciones:

   - Elija Recomendaciones.

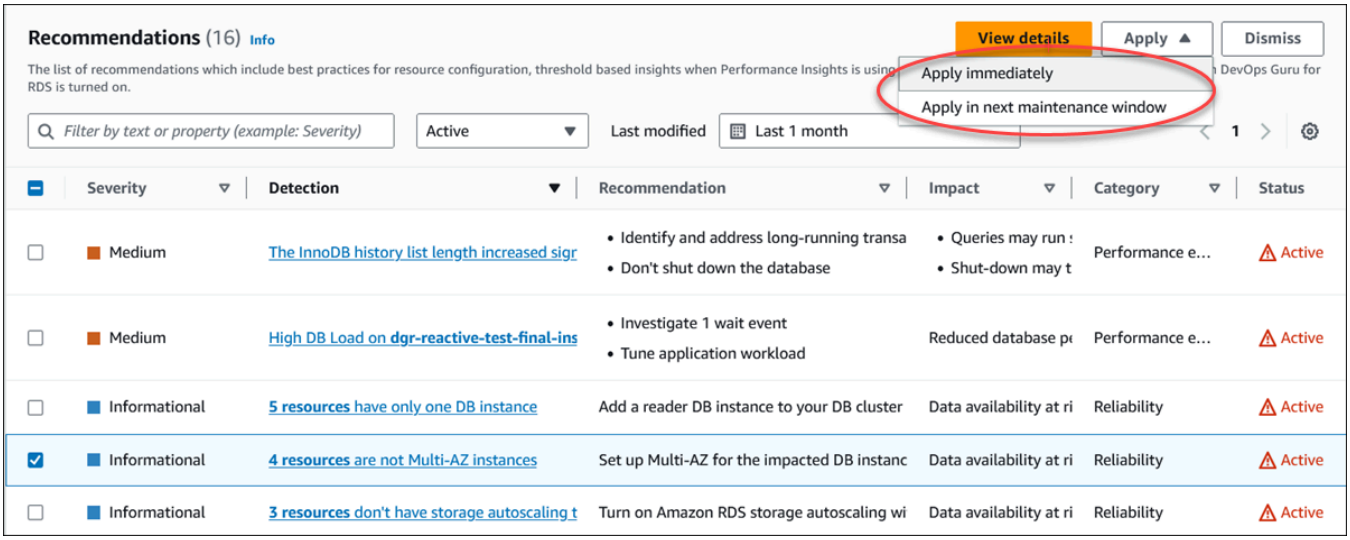
     Aparece la página de Recomendaciones con la lista de todas las recomendaciones.
   - Elija Bases de datos y, a continuación, elija Recomendaciones para un recurso en la página de bases de datos.

     Los detalles aparecen en la pestaña Recomendaciones de la recomendación seleccionada.
   - Seleccione Detección para ver una recomendación activa en la página Recomendaciones o en la pestaña Recomendaciones de la página Bases de datos.

     Aparece la página de detalles de la recomendación.
3. Elija una recomendación o uno o varios recursos afectados en la página de detalles de la recomendación y realice una de las siguientes acciones:

   - Seleccione Aplicar y, a continuación, seleccione Aplicar inmediatamente para aplicar la recomendación inmediatamente.
   - Seleccione Aplicar y elija Aplicar en el siguiente periodo de mantenimiento para programarlo en el siguiente periodo de mantenimiento.

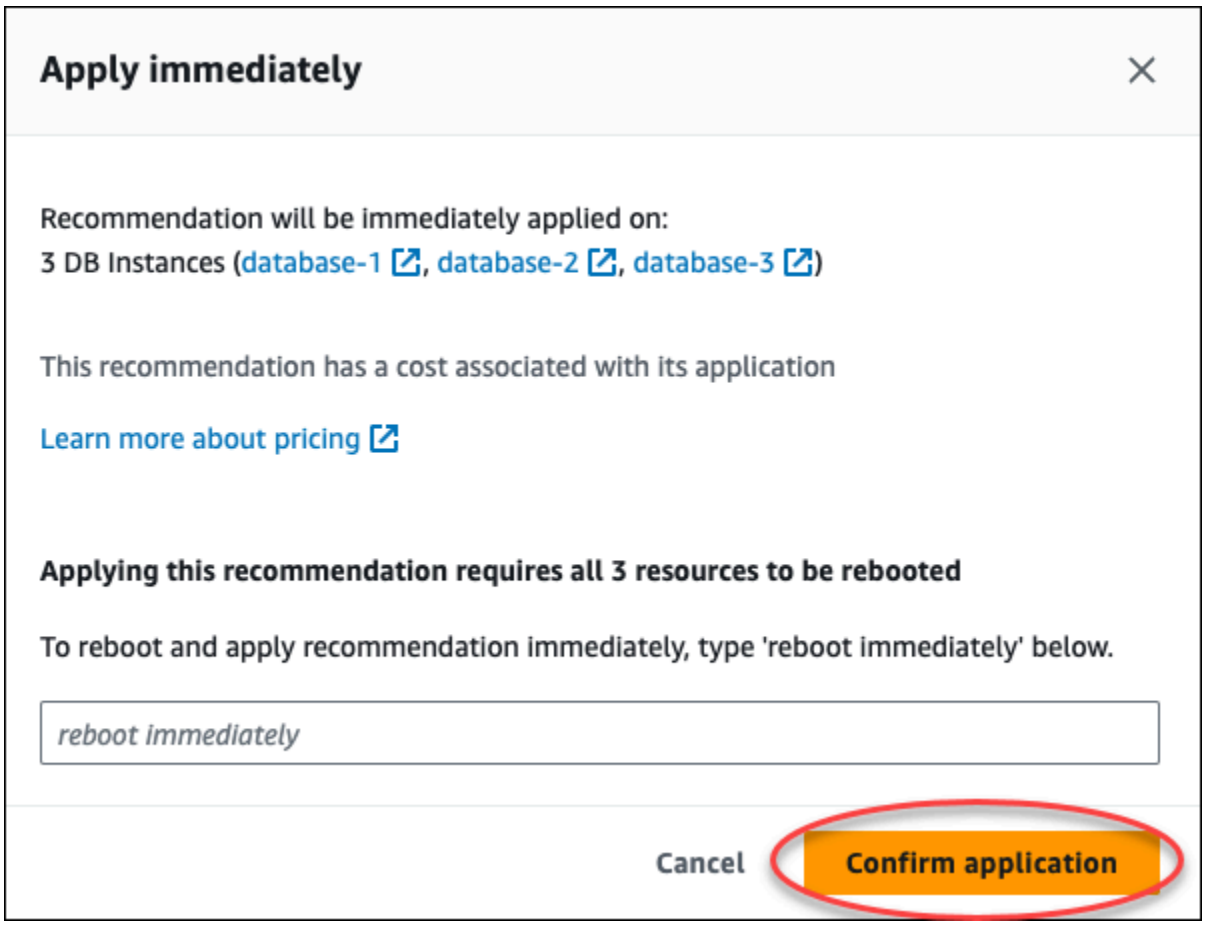
     El estado de la recomendación seleccionada se actualiza a pendiente hasta el siguiente período de mantenimiento.



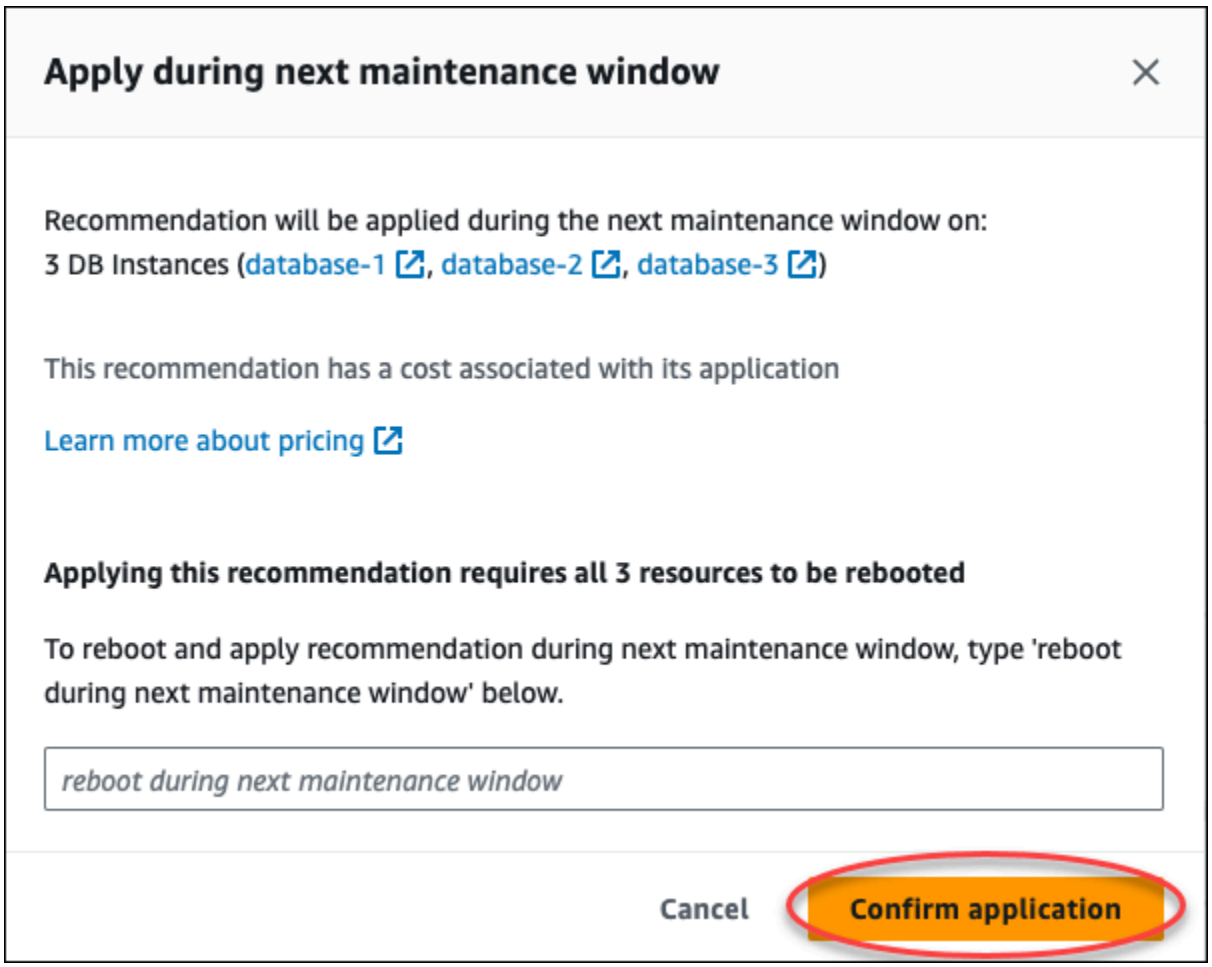
Aparece una ventana de confirmación.

4. Seleccione Confirmar aplicación para aplicar la recomendación. Esta ventana confirma si los recursos necesitan un reinicio automático o manual para que los cambios se apliquen.

   En el siguiente ejemplo, se muestra la ventana de confirmación para aplicar la recomendación inmediatamente.



El siguiente ejemplo muestra la ventana de confirmación para programar la aplicación de la recomendación en el siguiente período de mantenimiento.



Aparecerá un banner con un mensaje cuando la recomendación aplicada se haya aplicado correctamente o si no se ha podido aplicar.

En el siguiente ejemplo, se muestra el banner con el mensaje de aplicación correcta.

En el siguiente ejemplo, se muestra el banner con el mensaje de aplicación incorrecta.

API de RDS

Para aplicar una recomendación de Aurora basada en la configuración mediante la API de Amazon RDS

1. Utilice la operación DescribeDBRecommendations. Las `RecommendedActions` de la salida pueden tener una o varias acciones recomendadas.
2. Use el objeto RecommendedAction para cada acción recomendada del paso 1. La salida contiene `Operation` y `Parameters`.

   En el siguiente ejemplo, se muestra el resultado con una acción recomendada.

```
   "RecommendedActions": [
       {
           "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
           "Title": "Turn on auto backup", // localized
           "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
           "Operation": "ModifyDbInstance",
           "Parameters": [
               {
                   "Key": "DbInstanceIdentifier",
                   "Value": "my-mysql-instance-1"
               },
               {
                   "Key": "BackupRetentionPeriod",
                   "Value": "7"
               }
           ],
           "ApplyModes": ["immediately", "next-maintenance-window"],
           "Status": "applied"
       },
       ... // several others
   ],
```

3. Utilice la `operation` para cada acción recomendada del resultado del paso 2 e introduzca los valores de `Parameters`.
4. Cuando la operación del paso 2 se haya realizado correctamente, utilice la operación ModifyDBRecommendation para modificar el estado de la recomendación.

Descarte de las recomendaciones de Amazon Aurora

Puede descartar una o más recomendaciones de Amazon Aurora mediante la consola de Amazon RDS, la AWS CLI o la API de Amazon RDS.

Consola

Para descartar una o varias recomendaciones

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, realice una de las siguientes acciones:

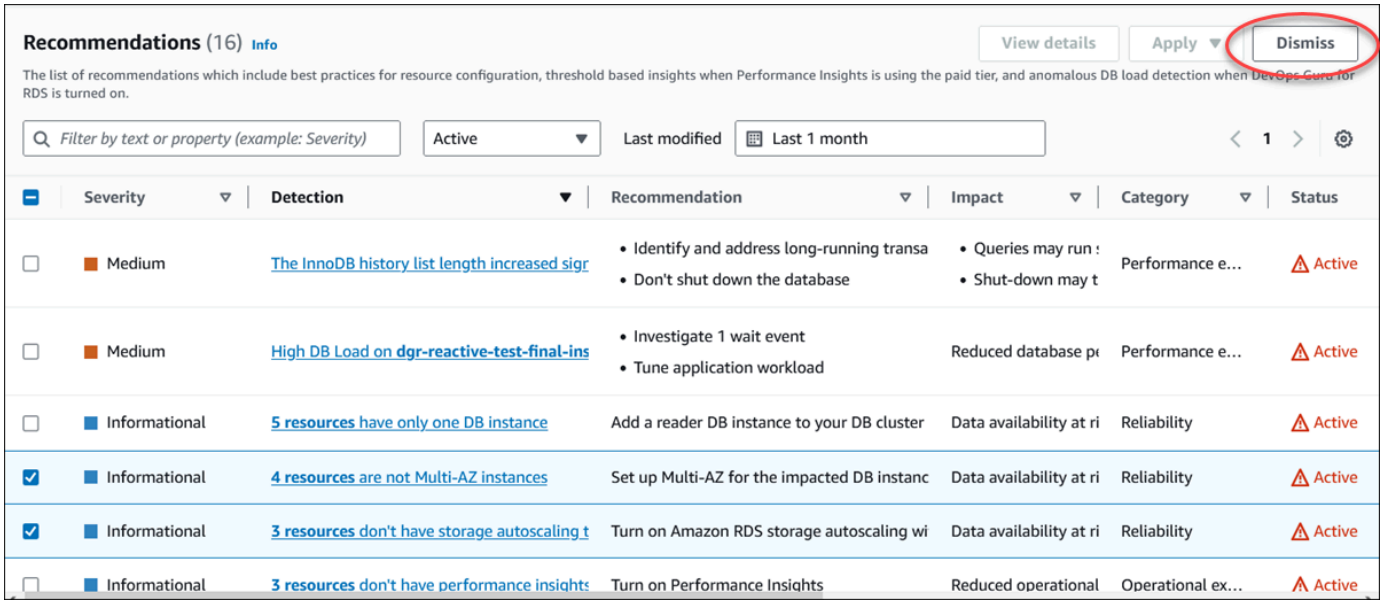
   - Elija Recomendaciones.

     Aparece la página de Recomendaciones con la lista de todas las recomendaciones.
   - Elija Bases de datos y, a continuación, elija Recomendaciones para un recurso en la página de bases de datos.

     Los detalles aparecen en la pestaña Recomendaciones de la recomendación seleccionada.
   - Seleccione Detección para ver una recomendación activa en la página Recomendaciones o en la pestaña Recomendaciones de la página Bases de datos.

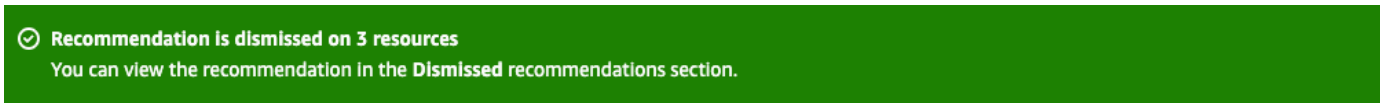
     La página de detalles de la recomendación muestra la lista de los recursos afectados.
3. Elija una o más recomendaciones o uno o más recursos afectados en la página de detalles de la recomendación y, a continuación, elija Descartar.

   En el siguiente ejemplo, se muestra la página Recomendaciones con varias recomendaciones activas seleccionadas para descartarlas.



Un banner muestra un mensaje cuando se descartan una o más recomendaciones seleccionadas.

En el siguiente ejemplo, se muestra el banner con el mensaje de aplicación correcta.



En el siguiente ejemplo, se muestra el banner con el mensaje de aplicación incorrecta.

CLI

Para descartar una recomendación de Aurora utilizando la AWS CLI

1. Ejecute el comando `aws rds describe-db-recommendations --filters "Name=status,Values=active"`.

   En el resultado se proporciona una lista de recomendaciones en el estado `active`.
2. Busque el `recommendationId` para la recomendación que desee descartar en el paso 1.
3. Ejecute el comando `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` con el `recommendationId` del paso 2 para descartar la recomendación.

API de RDS

Para descartar una recomendación de Aurora mediante la API de Amazon RDS, utilice la operación ModifyDBRecommendation.

Modificación de las recomendaciones de Amazon Aurora descartadas a recomendaciones activas

Puede cambiar una o más recomendaciones de Amazon Aurora descartadas a activas mediante la consola de Amazon RDS, la AWS CLI o la API de Amazon RDS.

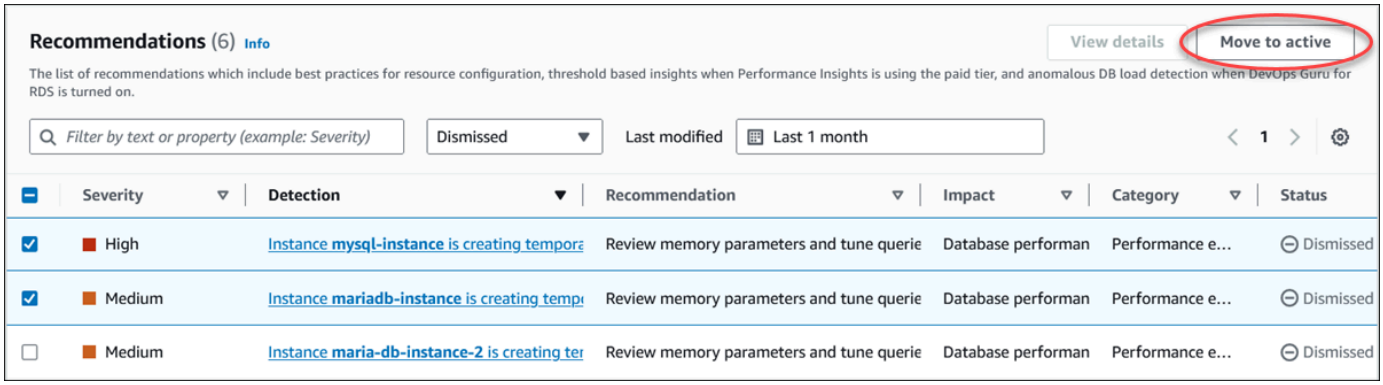
Consola

Para trasladar una o más recomendaciones descartadas a las recomendaciones activas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, realice una de las siguientes acciones:
   - Elija Recomendaciones.

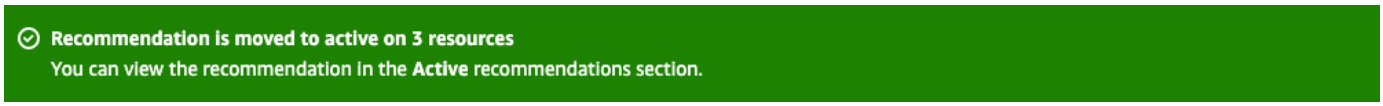
     En la página Recomendaciones, se muestra una lista de recomendaciones ordenadas por su gravedad para todos los recursos de su cuenta.
   - Elija Bases de datos y, a continuación, elija Recomendaciones para un recurso en la página de bases de datos.

     La pestaña Recomendaciones muestra las recomendaciones y sus detalles para el recurso seleccionado.
3. Elija una o más recomendaciones descartadas de la lista y, a continuación, elija Pasar a activas.



Un banner muestra un mensaje de éxito o fracaso al pasar las recomendaciones seleccionadas del estado descartado al estado activo.

En el siguiente ejemplo, se muestra el banner con el mensaje de aplicación correcta.



En el siguiente ejemplo, se muestra el banner con el mensaje de aplicación incorrecta.

CLI

Para convertir una recomendación de Aurora descartada en una recomendación activa mediante la AWS CLI

1. Ejecute el comando `aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"`.

   En el resultado se proporciona una lista de recomendaciones en el estado `dismissed`.
2. Busque el `recommendationId` de la recomendación para la que desee cambiar el estado desde el paso 1.
3. Ejecute el comando `>aws rds modify-db-recommendation --status active --recommendationId <ID>` con el `recommendationId` del paso 2 para cambiar la recomendación al estado activa.

API de RDS

Para convertir una recomendación de Aurora descartada en una recomendación activa mediante la API de Amazon RDS, utilice la operación ModifyDBRecommendation.

Recomendaciones de la referencia de Amazon Aurora

Amazon Aurora genera recomendaciones para un recurso cuando se crea o modifica el recurso. Puede encontrar ejemplos de recomendaciones de Amazon Aurora en la siguiente tabla.

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El recurso de copias de seguridad automatizadas está desactivado	Las copias de seguridad automatizadas no están activadas en sus instancias de base de datos. Se recomienda usar copias de seguridad automatizadas porque permiten la recuperación a un momento dado de su instancia de base de datos.	Active las copias de seguridad automatizadas con un período de retención de hasta 14 días.	Sí	Información general de copias de seguridad y restauración de un clúster de base de datos Aurora Demystifying Amazon RDS backup storage costs en el blog de AWS Database
Debe efectuarse una mejora de la versión secundaria del motor	Los recursos de su base de datos no están ejecutando la última versión secundaria del motor de base de datos. La última versión secundaria contiene las últimas revisiones de seguridad y otras mejoras.	Actualícela a la última versión del motor.	Sí	Mantenimiento de un clúster de base de datos de Amazon Aurora
La monitorización mejorada está desactivada	Los recursos de la base de datos no tienen activada la monitorización mejorada. El	Active la monitorización mejorada	No	Supervisión de las métricas del sistema operativo con Supervisión mejorada

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
	monitoreo mejorado proporciona métricas del sistema operativo en tiempo real para el monitoreo y la solución de problemas.			

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El cifrado de almacenamiento está desactivado	Amazon RDS admite el cifrado en reposo para todos los motores de bases de datos mediante las claves que administra en AWS Key Management Service (AWS KMS). En una instancia de base de datos activa con cifrado de Amazon RDS, los datos almacenados en reposo en el almacenamiento están cifrados, de forma similar a las copias de seguridad, las réplicas de lectura y las instantáneas automatizadas. Si el cifrado no está activado al crear un clúster de base de datos de Aurora, debe restaurar una instantánea descifrad	Active el cifrado de los datos en reposo de su clúster de base de datos.	Sí	Seguridad en Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
	a en un clúster de base de datos cifrado.			
Clústeres de base de datos con todas las instancias en la misma zona de disponibilidad	Actualmente, los clústeres de base de datos se encuentran en una sola zona de disponibilidad. Utilice varias zonas de disponibilidad para mejorar la disponibilidad.	Agregue las instancias de base de datos a varias zonas de disponibilidad de su clúster de base de datos.	No	Alta disponibilidad para Amazon Aurora
Instancias de bases de datos en los clústeres con tamaños de instancia heterogéneos	Le recomendamos que utilice la misma clase de instancia de base de datos para todas las instancias de base de datos en su clúster de base de datos.	Utilice la misma clase y tamaño de instancia de base de datos para todas las instancias de base de datos del clúster de base de datos.	Sí	Replicación con Amazon Aurora
Instancias de bases de datos en los clústeres con clases de instancia heterogéneas	Le recomendamos que utilice la misma clase de instancia de base de datos para todas las instancias de base de datos en su clúster de base de datos.	Utilice la misma clase y tamaño de instancia de base de datos para todas las instancias de base de datos del clúster de base de datos.	Sí	Replicación con Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
Instancias de bases de datos en los clústeres con grupos de parámetros heterogéneos	Recomendamos que todas las instancias de base de datos del clúster de base de datos utilicen el mismo grupo de parámetros de base de datos.	Asocie la instancia de base de datos con el grupo de parámetros de base de datos asociado a la instancia de escritura de su clúster de base de datos.	No	Grupos de parámetros para Amazon Aurora
Los clústeres de base de datos de Amazon RDS tienen una instancia de base de datos	Agregue al menos una instancia de base de datos más a su clúster de base de datos para mejorar la disponibilidad y el rendimiento.	Agregue una instancia de base de datos de lectura a su clúster de base de datos.	No	Alta disponibilidad para Amazon Aurora
Información de rendimiento está desactivado	Información de rendimiento monitoriza la carga de la instancia de base de datos para ayudarle a analizar y solucionar los problemas de rendimiento de la base de datos. Le recomendamos que active Información de rendimiento.	Activar Información de rendimiento.	No	Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
Es necesario actualizar las versiones principales de los recursos de RDS	No se admiten las bases de datos con la versión principal actual del motor de base de datos. Le recomendamos que actualice a la última versión principal, que incluye nuevas funciones y mejoras.	Actualización a la versión principal más reciente del motor de base de datos.	Sí	Actualizaciones de Amazon Aurora Creación de una implementación azul/verde en Amazon Aurora
Tamaño del volumen máximo del clúster de base de datos	Las versiones más recientes del motor admiten volúmenes de almacenamiento más grandes para el clúster de base de datos.	Le recomendamos que actualice la versión del motor del clúster de base de datos a la versión más reciente para beneficiarse del aumento de la capacidad de almacenamiento.	Sí	Límites de tamaño de Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
Clústeres de base de datos con todas las instancias de lector en la misma zona de disponibilidad	Las zonas de disponibilidad (AZ) representan ubicaciones distintas entre sí para proporcionar aislamiento en caso de interrupciones en cada región de AWS. Es recomendable que distribuya la instancia principal y las instancias de lectura del clúster de base de datos entre varias AZ para mejorar la disponibilidad del clúster de base de datos. Puede crear un clúster multi-AZ mediante la Consola de administración de AWS, la CLI de AWS API de Amazon RDS al crear el clúster. También puede modificar el clúster de Aurora ya existente y convertirlo en un clúster multi-AZ agregando una nueva	Su clúster de base de datos dispone de todas las instancias de lectura en la misma zona de disponibilidad. Recomendamos distribuir las instancias de lector entre varias zonas de disponibilidad. La distribución aumenta la disponibilidad y mejora el tiempo de respuesta al reducir la latencia de la red entre los clientes y la base de datos.	No	Alta disponibilidad para Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
	instancia de lector y especificando una AZ distinta.			
Los parámetros de la memoria de la base de datos difieren de los predeterminados	Los parámetros de memoria de las instancias de base de datos difieren considerablemente de los valores predeterminados. Esta configuración puede afectar al rendimiento y provocar errores. Recomendamos restablecer los parámetros de memoria personalizados para la instancia de base de datos a sus valores predeterminados en el grupo de parámetros de la base de datos.	Restablezca los parámetros de memoria a sus valores predeterminados.	No	Grupos de parámetros para Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro del caché de consultas está activado	Cuando los cambios requieran que se purgue la caché de consultas, parecerá que la instancia de base de datos se ha bloqueado. La mayoría de las cargas de trabajo no se benefician de una caché de consultas. La caché de consultas se ha quitado de la versión 8.0 de MySQL y posteriores. Es recomendable que establezca el parámetro query_cache_type en 0.	Establezca el valor del parámetro `query_cache_type` en `0` en el grupo de parámetros de su base de datos.	Sí	Grupos de parámetros para Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `log_output` está establecido en table	Cuando `log_output` se establece en `TABLE`, se utiliza más espacio de almacenamiento que cuando `log_output` se establece en `FILE`. Recomendamos que establezca el parámetro en `FILE` para evitar que se alcance el límite de tamaño de almacenamiento. Está establecido en `FILE` de forma predeterminada en MySQL versión 8.4 y posteriores.	Establezca el valor del parámetro `log_output` en `FILE` en el grupo de parámetros de su base de datos.	No	Archivos de registro de base de datos de Aurora MySQL

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `autovacuum` está desactivado	El parámetro autovacuum está desactivado en sus clústeres de base de datos. Desactivar autovacuum aumenta la sobrecarga de la tabla y del índice y afecta al rendimiento. Le recomendamos que active autovacuum en sus grupos de parámetros de base de datos.	Active el parámetro autovacuum en sus grupos de parámetros de clúster de base de datos.	No	Understanding autovacuum in Amazon RDS for PostgreSQL environments en el blog de AWS Database

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `synchronous_commit` está desactivado	Cuando el parámetro `synchronous_commit` está desactivado, es posible que se pierdan datos si la base de datos se bloquea. La durabilidad de la base de datos está en riesgo. Le recomendamos que active el parámetro `synchronous_commit`.	Active el parámetro `synchronous_commit` en sus grupos de parámetros de la base de datos.	Sí	Amazon Aurora PostgreSQL parameters: Replication, security, and logging en el blog de AWS Database

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `track_counts` está desactivado	Si el parámetro `track_counts` está desactivado, la base de datos no recopila las estadísticas de actividad de la base de datos. Autovacuum necesita estas estadísticas para funcionar correctamente. Es recomendable que establezca el parámetro `track_counts` en 1.	Establezca el parámetro `track_counts` en 1.	No	Estadísticas de tiempo de ejecución para PostgreSQL
El parámetro `enable_indexonlyscan` está desactivado	El planificador u optimizador de consultas no puede usar el plan de análisis de solo índice si está desactivado. Es recomendable que establezca el valor del parámetro `enable_indexonlyscan` en 1.	Establezca el valor del parámetro `enable_indexonlyscan` en 1.	No	Configuración del método del planificador para PostgreSQL

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `enable_indexscan` está desactivado	El planificador u optimizador de consultas no puede usar el plan de análisis de índice si está desactivado. Es recomendable que defina el valor `enable_indexscan` en 1.	Establezca el valor del parámetro `enable_indexscan` en 1.	No	Configuración del método del planificador para PostgreSQL
El parámetro `innodb_flush_log_at_trx` está desactivado	El valor del parámetro `innodb_flush_log_at_trx` de la instancia de base de datos no es un valor seguro. Este parámetro controla la persistencia de las operaciones de confirmación en el disco. Es recomendable que establezca el parámetro `innodb_flush_log_at_trx` en 1.	Establezca el valor del parámetro `innodb_flush_log_at_trx` en 1.	No	Configuración de la frecuencia de vaciado del búfer de registro

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `innodb_stats_persistent` está desactivado	Su instancia de base de datos no está configurada para conservar las estadísticas de InnoDB en el disco. Cuando las estadísticas no están almacenadas, se vuelven a calcular cada vez que la instancia se reinicia y se accede a la tabla. Esto provoca variaciones en el plan de ejecución de las consultas. Puede modificar el valor de este parámetro global a nivel de tabla. Es recomendable que establezca el valor del parámetro `innodb_stats_persistent` en `ON`.	Establezca el valor del parámetro `innodb_stats_persistent` en `ON`.	No	Grupos de parámetros para Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `innodb_open_files` es bajo	El parámetro `innodb_open_files` controla el número de archivos que InnoDB puede abrir a la vez. InnoDB abre todos los archivos de registro y de espacio de tablas del sistema cuando se ejecuta mysqld. Su instancia de base de datos tiene un valor bajo para la cantidad máxima de archivos que InnoDB puede abrir a la vez. Le recomendamos que establezca el parámetro `innodb_open_files` en un valor mínimo de 65.	Establezca el parámetro `innodb_open_files` en un valor mínimo de 65.	Sí	Archivos abiertos de InnoDB para MySQL

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `max_user_connections` es bajo	La instancia de base de datos tiene un valor bajo para el número máximo de conexiones simultáneas para cada cuenta de base de datos. Se recomienda aumentar el parámetro `max_user_connections` a un número superior a `5`.	Aumente el valor del parámetro `max_user_connections` a un número superior a `5`.	Sí	Establecimiento de límites de recursos de la cuenta para MySQL
Las réplicas de lectura están abiertas en modo de escritura	Su instancia de base de datos tiene la réplica de lectura en modo de escritura, lo que permite actualizaciones de los clientes. Se recomienda configurar el parámetro `read_only` en `TrueIfReplica` para que las réplicas de lectura no estén en modo de escritura.	Establezca el valor del parámetro `read_only` en `TrueIfReplica`.	No	Grupos de parámetros para Amazon Aurora

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
La configuración del parámetro `innodb_default_row_format` no es segura	Su instancia de base de datos encuentra un problema conocido: una tabla creada en una versión de MySQL anterior a la 8.0.26 con el valor `row_format` establecido en `COMPACT` o `REDUNDANT` será inaccesible e irrecuperable si el índice supera los 767 bytes. Es recomendable que establezca el valor del parámetro `innodb_default_row_format` en `DYNAMIC`.	Establezca el valor del parámetro `innodb_default_row_format` en `DYNAMIC`.	No	Cambios en MySQL 8.0.26

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El parámetro `general_logging` está activado	El registro general está activado para su instancia de base de datos. Esta configuración es útil para solucionar los problemas de la base de datos. Sin embargo, la activación del registro general aumenta la cantidad de operaciones de E/S y el espacio de almacenamiento asignado, lo que puede provocar problemas de contención y una degradación del rendimiento. Compruebe sus requisitos para el uso del registro general. Es recomendable que establezca el valor del parámetro `general_logging` en `0`.	Compruebe sus requisitos para el uso del registro general. Si no es obligatorio, le recomendamos que establezca el valor del parámetro `general_logging` en `0`.	No	Información general de los registros de bases de datos de Aurora MySQL

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
El clúster de base de datos está insuficientemente aprovisionado para la carga de trabajo de lectura	Le recomendamos que agregue una instancia de base de datos de lectura a su clúster de base de datos con la misma clase y tamaño de instancia que la instancia de base de datos de escritor del clúster. La configuración actual tiene una instancia de base de datos con una carga de base de datos continuamente alta, causada en el mayor número de casos por operaciones de lectura. Distribuya estas operaciones agregando otra instancia de base de datos al clúster y dirigiendo la carga de trabajo de lectura al punto de conexión de solo lectura del clúster de base de datos.	Agregar una instancia de base de datos de lectura al clúster.	No	Adición de réplicas de Aurora a un clúster de base de datos Administración del rendimiento y el escalado para clústeres de base de datos Aurora Precios de Amazon RDS

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
Instancia de RDS con aprovisionamiento insuficiente para la capacidad de memoria del sistema	Se recomienda ajustar las consultas para utilizar menos memoria o utilizar un tipo de instancia de base de datos con una mayor memoria asignada. Cuando la instancia se queda sin memoria, el rendimiento de la base de datos se ve afectado.	Utilice una instancia de base de datos con mayor capacidad de memoria	Sí	Scaling Your Amazon RDS Instance Vertically and Horizontally en el blog de AWS Database Tipos de instancia de Amazon RDS Precios de Amazon RDS
Instancia de RDS con aprovisionamiento insuficiente para la capacidad de CPU del sistema	Le recomendamos que ajuste las consultas para que usen menos CPU o que modifique la instancia de base de datos para que use una clase de instancia de base de datos con una asignación mayor de vCPU. El rendimiento de la base de datos puede disminuir cuando una instancia de base de datos se está quedando sin CPU.	Utilice una instancia de base de datos con mayor capacidad de CPU	Sí	Scaling Your Amazon RDS Instance Vertically and Horizontally en el blog de AWS Database Tipos de instancia de Amazon RDS Precios de Amazon RDS

Tipo	Descripción	Recomendación	Tiempo de inactividad requeri	Información adicional
Los recursos de RDS no utilizan correctamente el grupo de conexiones	Le recomendamos que habilite Amazon RDS Proxy para agrupar y compartir de manera eficiente las conexiones de bases de datos existentes. Si ya utiliza un proxy para su base de datos, configúrelo correctamente para mejorar la agrupación de conexiones y el equilibrio de carga en varias instancias de base de datos. RDS Proxy puede ayudar a reducir el riesgo de agotamiento y tiempo de inactividad de la conexión, al mismo tiempo que mejora la disponibilidad y la escalabilidad.	Habilite RDS Proxy o modifique la configuración de proxy existente	No	Scaling Your Amazon RDS Instance Vertically and Horizontally en el blog de AWS Database Amazon RDS Proxy para Aurora Precios de Amazon RDS Proxy

Consulta de métricas en la consola de Amazon RDS

Amazon RDS se integra con Amazon CloudWatch para mostrar una variedad de métricas de clústeres de base de datos de Aurora en la consola de RDS. Algunas métricas se aplican a nivel del clúster, mientras que otras se aplican a nivel de la instancia.Para obtener descripciones de , así como también de las métricas a nivel de instancia y clúster, consulte Referencia de métricas para Amazon Aurora.

Para el clúster de base de datos de Aurora, se monitorizan las siguientes categorías de métricas:

- CloudWatch: muestra las métricas de Amazon CloudWatch para Aurora a las que puede acceder desde la consola de RDS. También puede acceder a estas métricas desde la consola de CloudWatch. Cada métrica incluye un gráfico que muestra la métrica supervisada a lo largo de un periodo concreto. Para obtener una lista de las métricas de CloudWatch, consulte Métricas de Amazon CloudWatch para Amazon Aurora.
- Enhanced monitoring (Supervisión mejorada): muestra un resumen de las métricas del sistema operativo cuando su clúster de base de datos de Aurora activa la opción de Supervisión mejorada. RDS entrega las métricas de la supervisión mejorada a su cuenta de Amazon CloudWatch Logs. Cada métrica del sistema operativo incluye un gráfico que muestra la métrica supervisada a lo largo de un periodo concreto. Para obtener una descripción general, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada. Para ver una lista de métricas de Supervisión mejorada, consulte Métricas del sistema operativo en Supervisión mejorada.
- OS Process list (Lista de procesos de sistema operativo): muestra los detalles de cada proceso que se ejecuta en su clúster de base de datos.
- Performance Insights (Información sobre rendimiento): abre el panel de Información sobre rendimiento de la consola de Amazon RDS para una instancia de base de datos en el clúster de base de datos de Aurora. Performance Insights (Información sobre rendimiento) no es compatible a nivel del clúster.Para obtener más detalles acerca de Información sobre rendimiento, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora. Para obtener una lista de las métricas de Información sobre rendimiento, consulte Métricas de Amazon CloudWatch para Información de rendimiento de Amazon RDS.

Amazon RDS ahora ofrece una vista consolidada de las métricas de Información de rendimiento y CloudWatch en el panel de Información de rendimiento. Debe activarse Información de rendimiento para que la instancia de base de datos pueda utilizar esta vista. Puede elegir la nueva vista de monitorización en la pestaña Monitorización o Información de rendimiento en el panel de navegación.

Para ver las instrucciones para elegir esta vista, consulte Visualización de las métricas combinadas en la consola de Amazon RDS.

Visualización de las métricas combinadas en la consola de Amazon RDS

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.
Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.
Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Amazon RDS ofrece una vista consolidada de las métricas de Información de rendimiento y CloudWatch de la instancia de base de datos en el panel de Información de rendimiento. Puede utilizar el panel preconfigurado o crear un panel personalizado. El panel preconfigurado proporciona las métricas más utilizadas para ayudar a diagnosticar problemas de rendimiento en un motor de base de datos. Asimismo, puede crear un panel personalizado con las métricas de un motor de base

de datos que cumpla sus requisitos de análisis. A continuación, utilice este panel para todas las instancias de base de datos de ese tipo de motor de base de datos en su cuenta de AWS.

Puede elegir la vista de monitorización en la pestaña Monitorización o Información de rendimiento en el panel de navegación.

Información de rendimiento debe estar activada para el clúster de base de datos para ver las métricas combinadas en el panel de Información de rendimiento. Para obtener más información acerca de la activación de Información de rendimiento, consulte Activación y desactivación de Información de rendimiento de Aurora.

En las siguientes secciones, descubrirá cómo mostrar las métricas de Información de rendimiento y CloudWatch.

Temas

- Elección de la nueva vista de monitorización en la pestaña Monitorización
- Elección de la nueva vista de monitorización desde la página Información de rendimiento
- Creación de un panel personalizado con Información de rendimiento
- Elección del panel preconfigurado con Información de rendimiento

Elección de la nueva vista de monitorización en la pestaña Monitorización

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.
Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento.

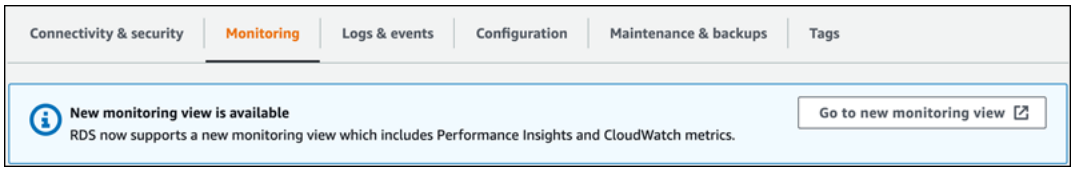
Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.
Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Desde la consola de Amazon RDS, puede elegir la nueva vista de monitorización a fin de consultar las métricas de Información de rendimiento y CloudWatch para su instancia de base de datos.

Para elegir la nueva vista de monitorización en la pestaña Monitorización

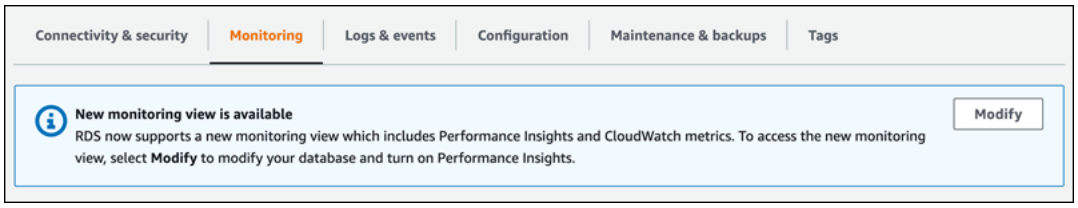
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, elija Bases de datos.
3. Elija el clúster de base de datos que desee monitorizar.
4. Desplácese hacia abajo y elija Monitorización.

   Aparece un banner con la opción de elegir la nueva vista de monitorización. En el siguiente ejemplo muestra el banner para elegir la nueva vista de monitorización.



5. Elija Ir a la nueva vista de supervisión para abrir el panel de Información de rendimiento con las métricas de Información de rendimiento y CloudWatch para el clúster de base de datos.
6. (Opcional) Si Información de rendimiento está desactivada para la instancia de base de datos, aparece un banner con la opción de modificar la instancia de base de datos y activar Información de rendimiento

El siguiente ejemplo muestra el banner para modificar la instancia de base de datos en la pestaña Monitorización.



Elija Modificar para modificar la instancia de base de datos y activar Información de rendimiento. Para obtener más información acerca de la activación de Información de rendimiento, consulte Activación y desactivación de Información de rendimiento de Aurora.

Elección de la nueva vista de monitorización desde la página Información de rendimiento

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.

Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.

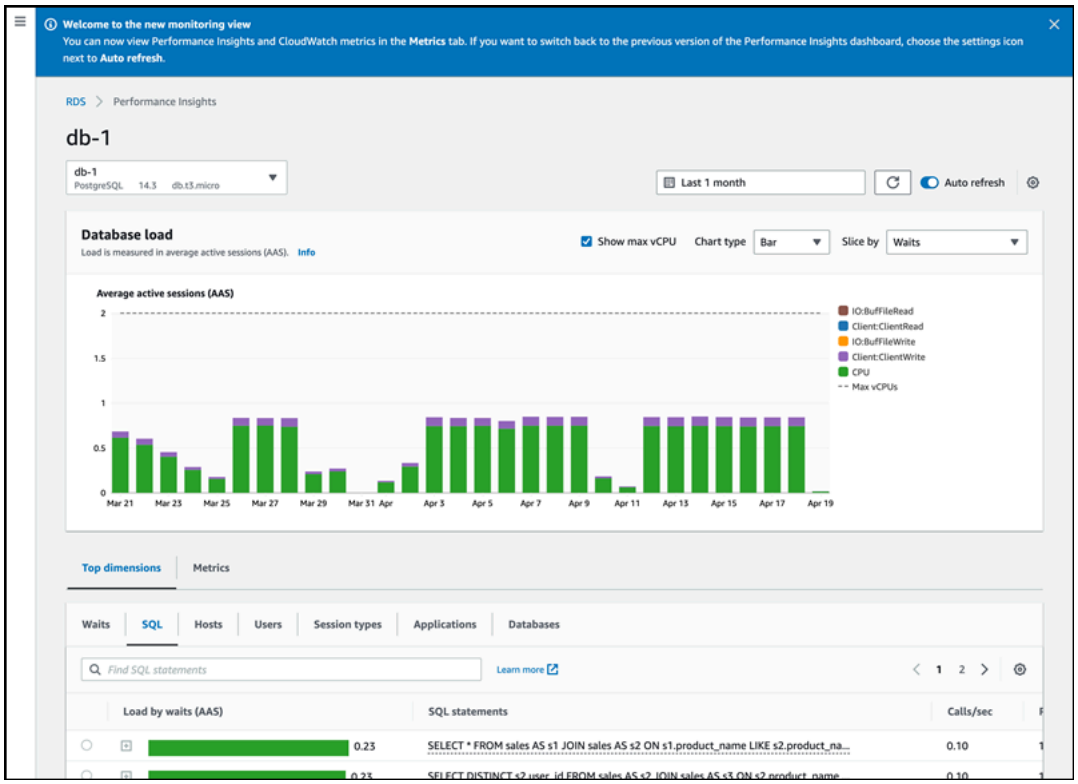
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.

Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Desde la consola de Amazon RDS, puede elegir la nueva vista de monitorización a fin de consultar las métricas de Información de rendimiento y CloudWatch para su instancia de base de datos.

Elección de la nueva vista de monitorización con Información de rendimiento en el panel de navegación

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Seleccione una instancia de base de datos para ver el panel de Información de rendimiento donde aparecen las métricas de Información de rendimiento y CloudWatch para su instancia de base de datos.



Creación de un panel personalizado con Información de rendimiento

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.

Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.

Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.

Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

En la nueva vista de monitorización, puede crear un panel personalizado con las métricas que necesita para cumplir con los requisitos de análisis.

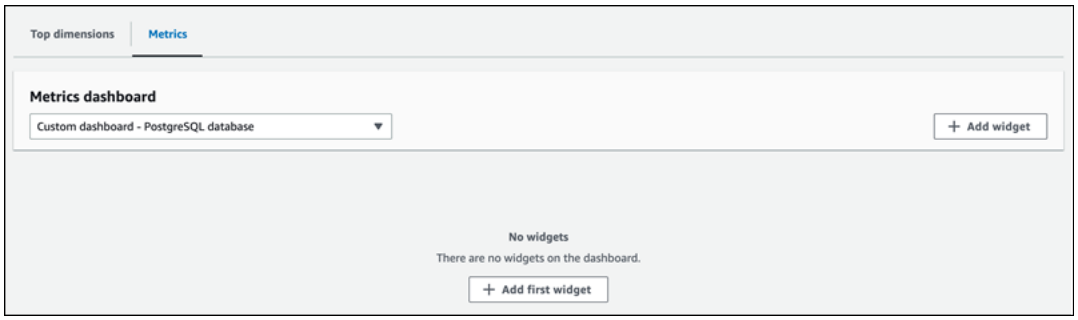
Puede crear un panel personalizado seleccionando las métricas de Información de rendimiento y CloudWatch para la instancia de base de datos. Puede utilizar este panel personalizado para otras instancias de base de datos del mismo tipo de motor de base de datos en su cuenta de AWS.

> **Note**
>
> El panel personalizado admite hasta 50 métricas.

Utilice el menú de configuración del widget para editar o eliminar el panel y mover o cambiar el tamaño de la ventana del widget.

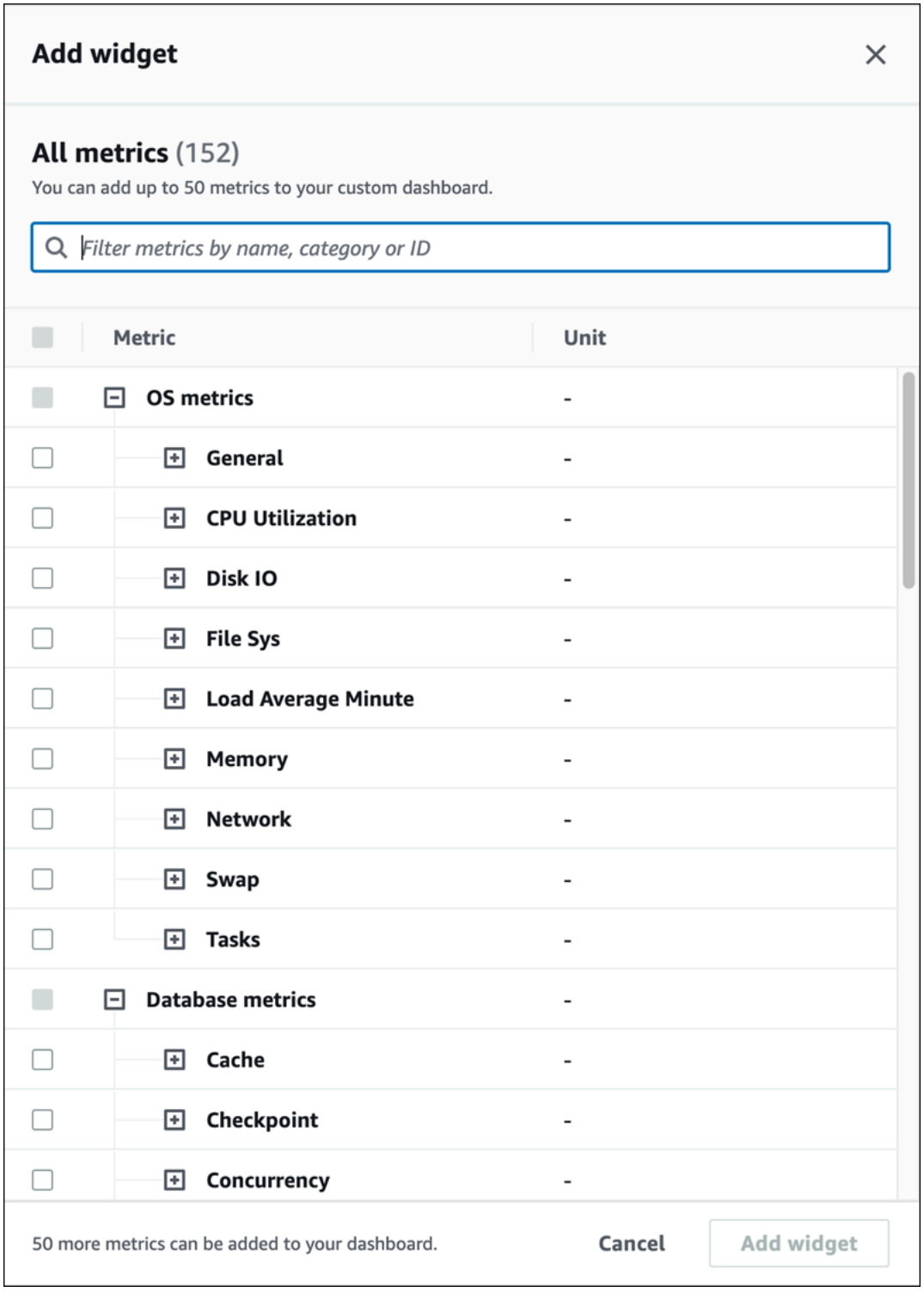
Creación de un panel personalizado con Información de rendimiento en el panel de navegación

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.
4. Desplácese hacia abajo hasta la pestaña Métricas de la ventana.
5. Seleccione el panel personalizado en la lista desplegable. En el siguiente ejemplo se muestra cómo se crea el panel personalizado.



6. Seleccione Agregar widget para abrir la ventana Agregar widget. Puede abrir y ver las métricas del sistema operativo (SO), las métricas de la base de datos y las métricas de CloudWatch disponibles en la ventana.

   El siguiente ejemplo muestra la ventana Agregar widget con las métricas.



7. Seleccione las métricas que desea ver en el panel y elija Agregar widget. Puede utilizar el campo de búsqueda para buscar una métrica específica.

   Las métricas seleccionadas aparecen en el panel.

8. (Opcional) Si quiere modificar o eliminar el panel, seleccione el icono de configuración en la parte superior derecha del widget y, a continuación, seleccione una de las siguientes acciones en el menú.

   - Editar: para modificar la lista de métricas de la ventana. Seleccione Actualizar widget después de seleccionar las métricas del panel.
   - Eliminar: para eliminar el widget. En la ventana de confirmación, elija Eliminar.

Elección del panel preconfigurado con Información de rendimiento

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.
Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.
Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

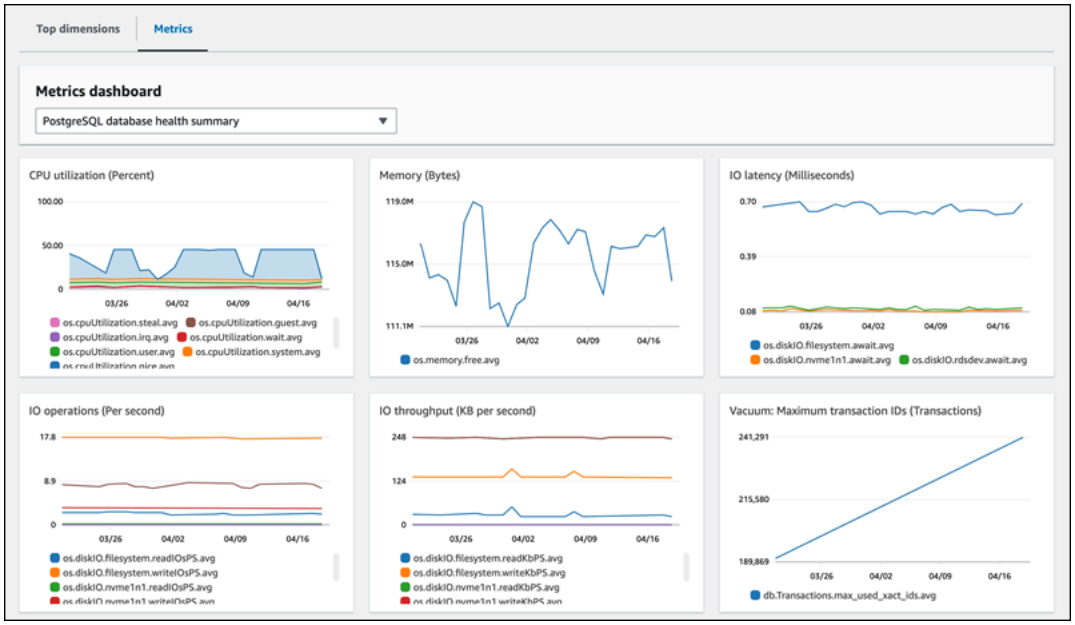
Puede ver las métricas más utilizadas en el panel preconfigurado. Este panel ayuda a diagnosticar problemas de rendimiento con un motor de base de datos y a reducir el tiempo medio de recuperación de horas a minutos.

> **Note**
>
> Este panel no se puede editar.

Elección del panel preconfigurado con Información de rendimiento en el panel de navegación

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.
4. Desplácese hacia abajo hasta la pestaña Métricas de la ventana.
5. Seleccione un panel preconfigurado de la lista desplegable.

   Puede ver las métricas de la instancia de base de datos en el panel. En el siguiente ejemplo se muestra un panel de control de métricas preconfigurado.



Supervisión de métricas de Amazon Aurora con Amazon CloudWatch

Amazon CloudWatch es un repositorio de métricas. El repositorio recopila y procesa datos sin procesar de Amazon Aurora en métricas legibles y casi en tiempo real. Para ver la lista completa de métricas de Amazon Aurora enviadas a CloudWatch, consulte Referencia de métricas para Amazon Aurora.

Para analizar y solucionar problemas del rendimiento de las bases de datos a escala, utilice Información sobre las bases de datos de CloudWatch.

Temas

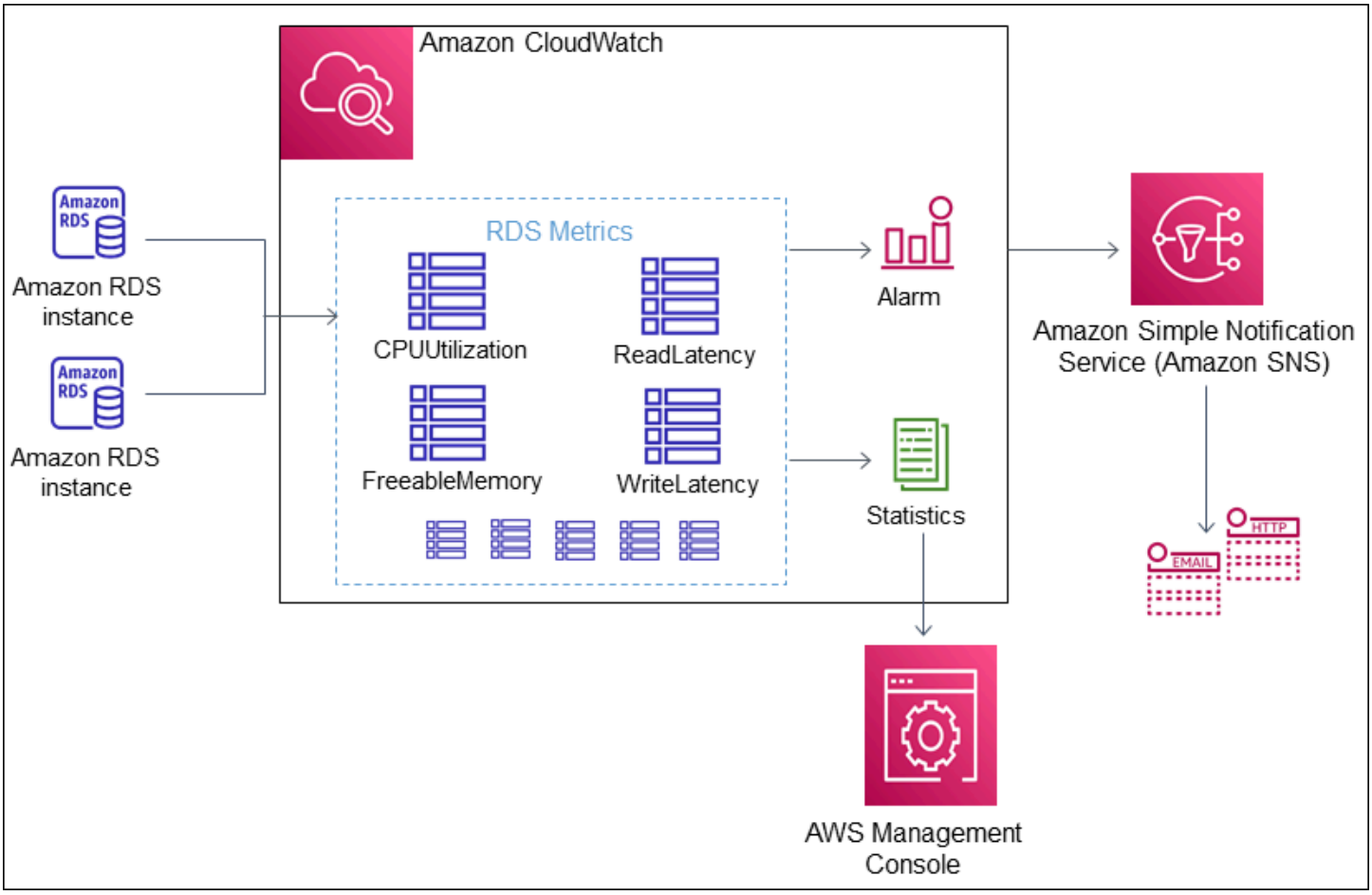
- Información general de Amazon Aurora y Amazon CloudWatch
- Visualización de las métricas de el clúster de base de datos en la consola de CloudWatch y la AWS CLI
- Exportación de las métricas de Información sobre rendimiento a CloudWatch
- Creación de alarmas de CloudWatch para supervisar Amazon Aurora

Información general de Amazon Aurora y Amazon CloudWatch

De forma predeterminada, Amazon Aurora envía datos de métricas a CloudWatch en periodos de 1 minuto. Por ejemplo, la métrica `CPUUtilization` registra el porcentaje de utilización de CPU para una instancia de base de datos a lo largo del tiempo. Los puntos de datos con un periodo de 60 segundos (1 minuto) están disponibles durante 15 días. Esto significa que puede acceder a información histórica y ver el rendimiento de su aplicación web o servicio.

Ahora puede exportar los paneles de métricas de Información de rendimiento desde Amazon RDS a Amazon CloudWatch. Puede exportar paneles de métricas personalizados o preconfigurados como un panel nuevo o añadirlos a un panel de CloudWatch existente. Los paneles de métricas exportados se pueden ver en la consola de CloudWatch. Para obtener más información sobre cómo exportar los paneles de métricas de Información de rendimiento a CloudWatch, consulte Exportación de las métricas de Información sobre rendimiento a CloudWatch.

Tal como se muestra en el siguiente diagrama, puede configurar alarmas para las métricas de CloudWatch. Por ejemplo, podría crear una alarma para cuando la utilización de la CPU para una instancia supere el 70 %. Puede configurar Amazon Simple Notification Service para que le envíe un correo electrónico cuando se supere el umbral.



Amazon RDS publica los siguientes tipos de métricas en Amazon CloudWatch:

- Métricas de Aurora tanto en el nivel de clúster como de instancia

  Para ver una tabla de estas métricas, consulte Métricas de Amazon CloudWatch para Amazon Aurora.
- Métricas de Performance Insights

  Para ver una tabla de estas métricas, consulte Métricas de Amazon CloudWatch para Información de rendimiento de Amazon RDS y Métricas de contador de Información de rendimiento.
- Métricas de Supervisión mejorada (publicadas en registros de Amazon Cloudwatch)

  Para ver una tabla de estas métricas, consulte Métricas del sistema operativo en Supervisión mejorada.
- Métricas de uso de las cuotas de servicio de Amazon RDS en su Cuenta de AWS

Para ver una tabla de estas métricas, consulte Métricas de uso de Amazon CloudWatch para Amazon Aurora. Para obtener más información acerca de las cuotas de Amazon RDS, consulte Cuotas y restricciones para Amazon Aurora.

Para obtener más información acerca de CloudWatch, consulte ¿Qué es Amazon CloudWatch? en la Guía del usuario de Amazon CloudWatch. Para obtener más información acerca de la retención de métricas de CloudWatch, consulte Retención de métricas.

Visualización de las métricas de el clúster de base de datos en la consola de CloudWatch y la AWS CLI

A continuación, puede encontrar detalles sobre cómo consultar las métricas de su instancia de base de datos mediante CloudWatch. Para obtener información acerca del monitoreo de métricas para el sistema operativo de su instancia de base de datos en tiempo real mediante CloudWatch Logs, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.

Si utiliza recursos de Amazon Aurora, Amazon Aurora envía métricas y dimensiones a Amazon CloudWatch cada minuto.

Ahora puede exportar los paneles de métricas de Información de rendimiento desde Amazon RDS a Amazon CloudWatch y ver esas métricas en la consola de CloudWatch. Para obtener más información sobre cómo exportar los paneles de métricas de Información de rendimiento a CloudWatch, consulte Exportación de las métricas de Información sobre rendimiento a CloudWatch.

Utilice los siguientes procedimientos para consultar las métricas de Amazon Aurora en la consola de CloudWatch y la CLI.

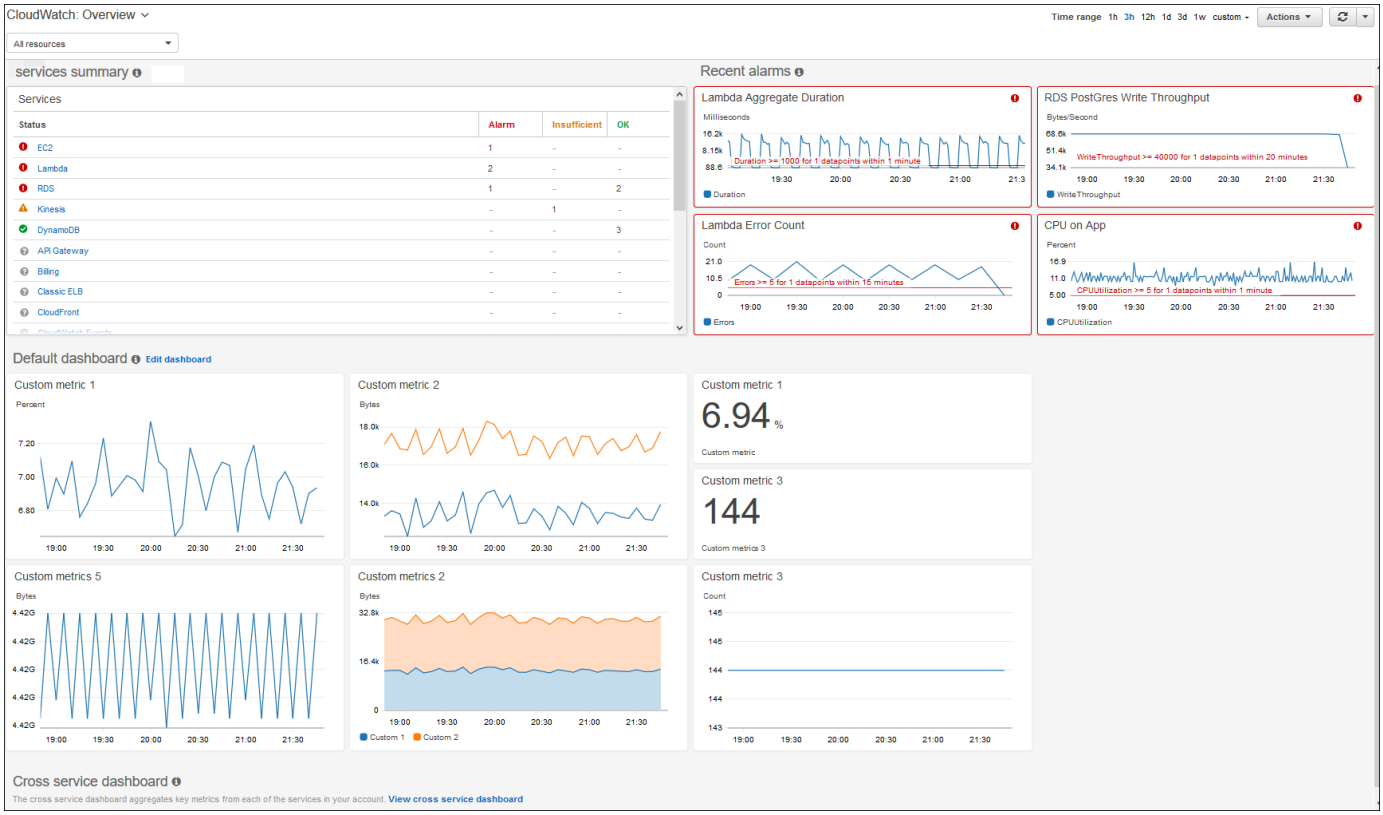
Consola

Para consultar las métricas desde la consola de Amazon CloudWatch

Las métricas se agrupan en primer lugar por el espacio de nombres de servicio y, a continuación, por las diversas combinaciones de dimensiones dentro de cada espacio de nombres.

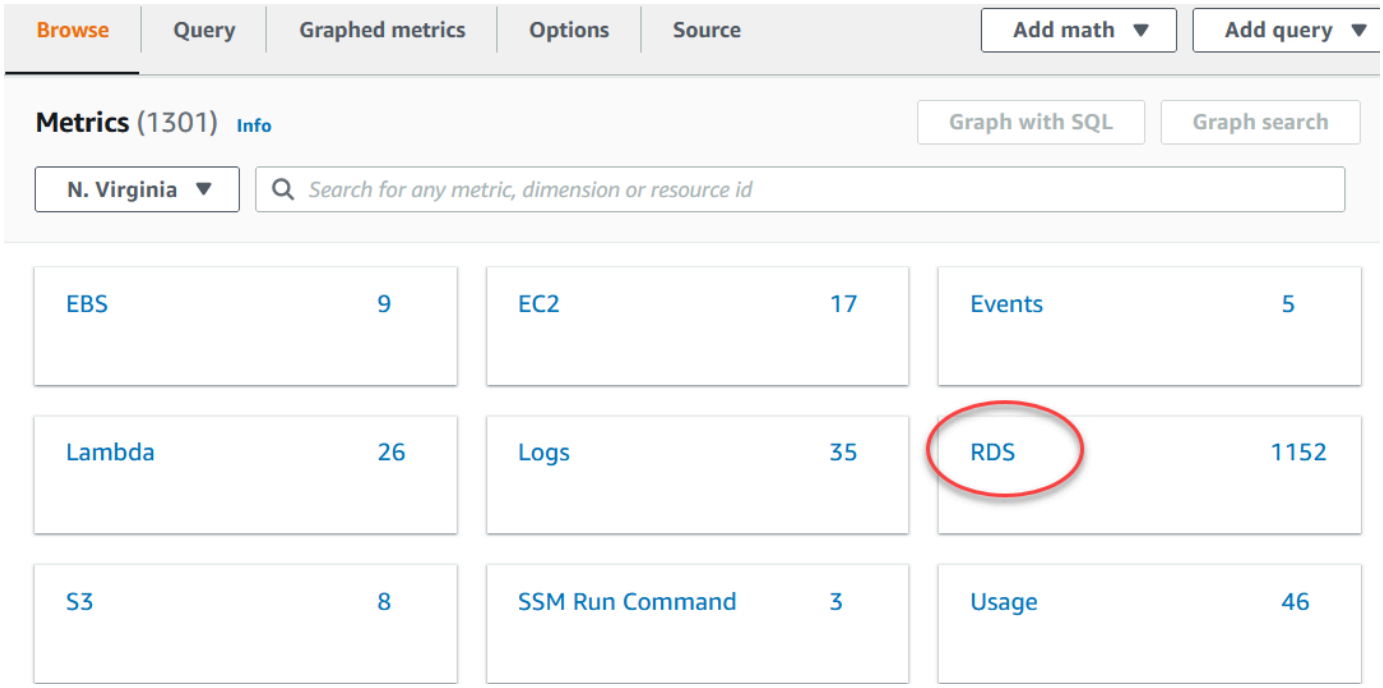
1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.

   Aparece la página de inicio de información general de CloudWatch.



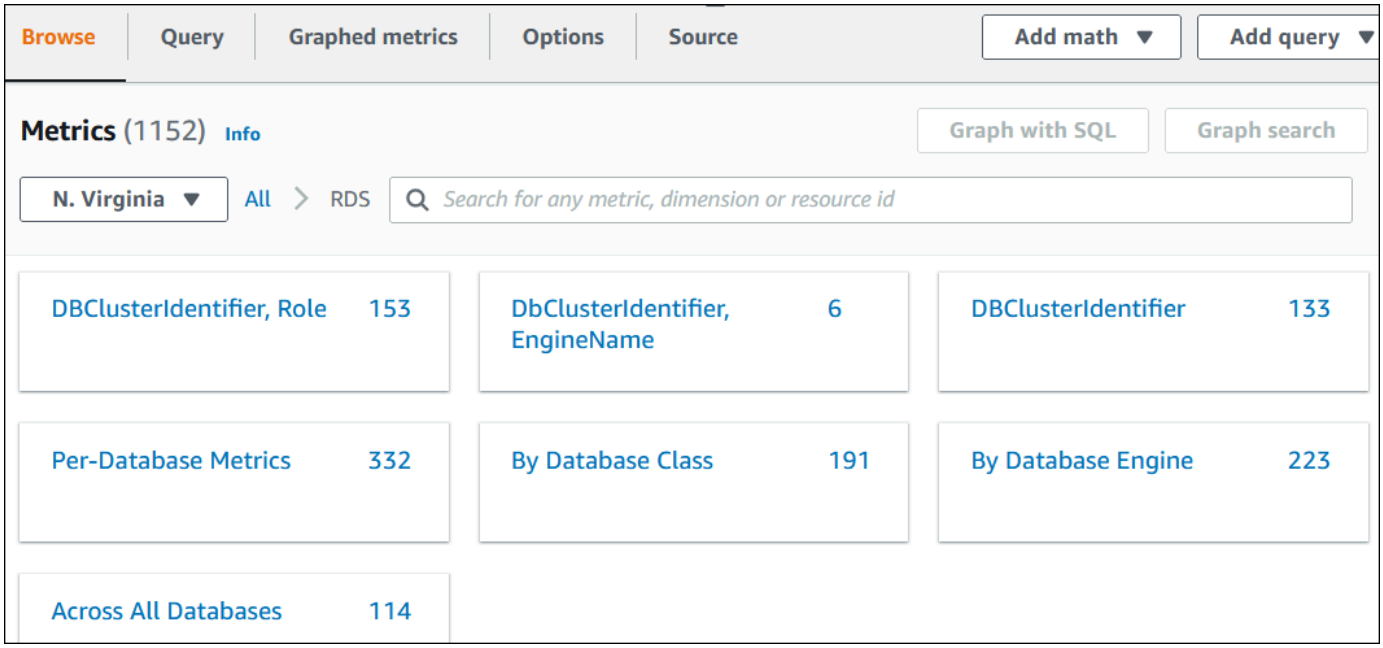
2. Si es necesario, cambie la Región de AWS. En la barra de navegación, elija la Región de AWS donde se encuentran los recursos de AWS. Para obtener más información, consulte Puntos de conexión y Regiones de .

3. En el panel de navegación, elija Metrics (Métricas) y, a continuación, All metrics (Todas las métricas).

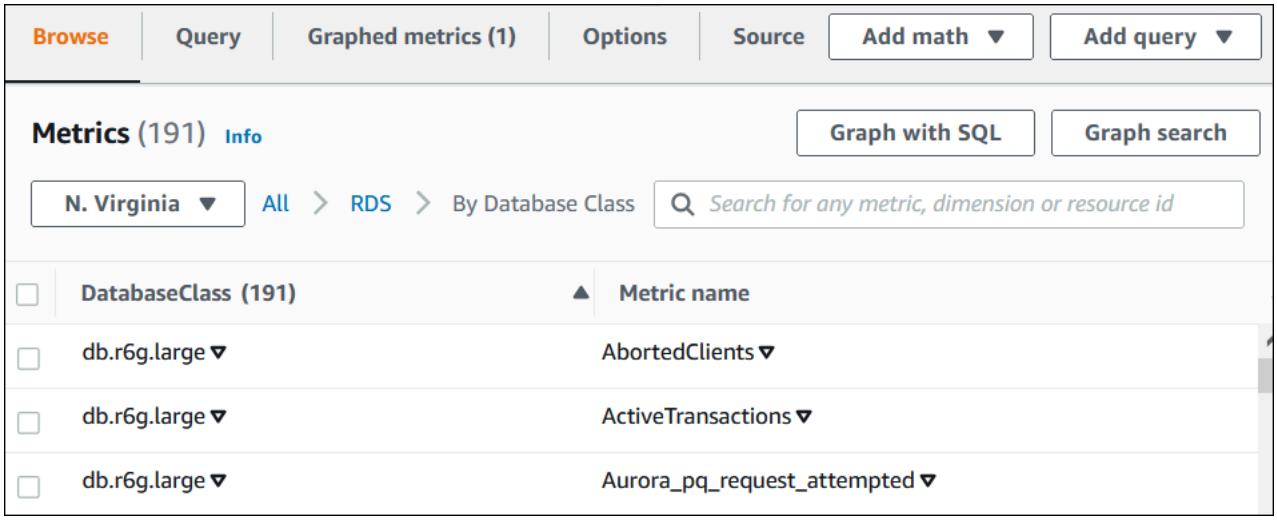


4. Desplácese hacia abajo y elija el espacio de nombres de la métrica de RDS.

   En la página, se muestran las dimensiones de Amazon Aurora. Para obtener una descripción completa de estas dimensiones, consulte Dimensiones de Amazon CloudWatch para Aurora..



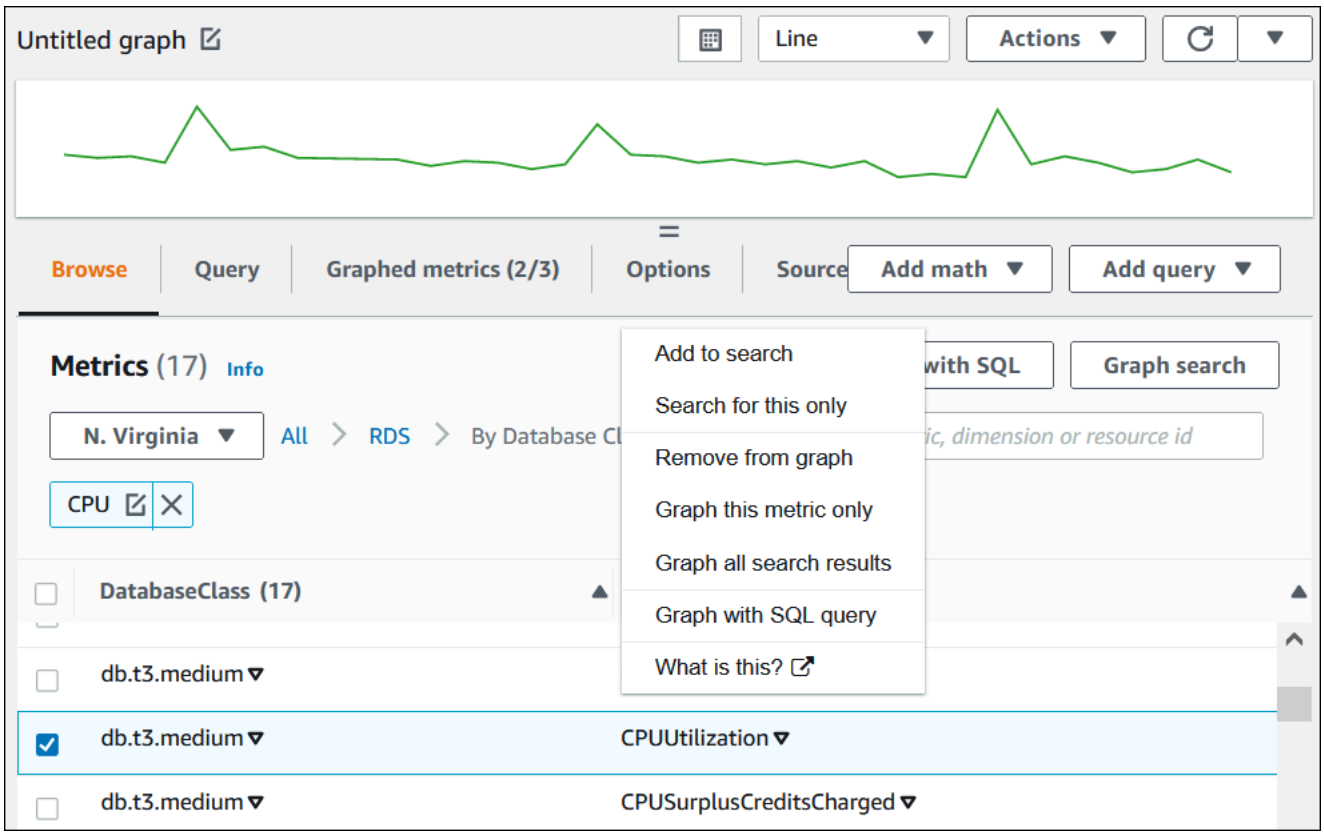
5. Seleccione una dimensión de métrica, por ejemplo, By Database Class (Por clase de base de datos).



6. Realice cualquiera de las siguientes acciones:

   - Para ordenar las métricas, utilice el encabezado de columna.
   - Para representar gráficamente una métrica, active la casilla de verificación situada junto a ella.
   - Para filtrar por recurso, elija el ID de recurso y, a continuación, elija Add to search (Añadir a la búsqueda).
   - Para filtrar por métrica, elija el nombre de la métrica y, a continuación, elija Add to search (Añadir a la búsqueda).

   En el siguiente ejemplo se filtra la clase db.t3.medium y se grafica la métrica CPUUtilization.



Encontrará más detalles sobre la forma de analizar el uso de los recursos de Aurora PostgreSQL mediante las métricas de CloudWatch. Para obtener más información, consulte Uso de las métricas de Amazon CloudWatch para analizar el uso de los recursos de Aurora PostgreSQL

AWS CLI

Para obtener información sobre métricas con la AWS CLI, utilice el comando de CloudWatch `list-metrics`. En el siguiente ejemplo, se enumeran todas las métricas del espacio de nombres de `AWS/RDS`.

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Para obtener datos de métricas, utilice el comando `get-metric-data`.

El siguiente ejemplo obtiene estadísticas de `CPUUtilization` para la instancia `my-instance` en el periodo de 24 horas específico, con un nivel de detalle de 5 minutos.

Cree un archivo JSON denominado `CPU_metric.json` con el siguiente contenido.

```
{
```

```
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
     "MetricStat" : {
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

Para Linux, macOS o:Unix

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

En:Windows

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

El resultado de ejemplo aparece como se muestra a continuación:

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
            "Timestamps": [
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
                ...
            ],
```

```
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
                13.02521708695145,
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Para obtener más información, consulte Obtención de estadísticas para una métrica en la Guía del usuario de Amazon CloudWatch.

Exportación de las métricas de Información sobre rendimiento a CloudWatch

Información sobre rendimiento le permite exportar los paneles de métricas preconfiguradas o personalizadas de su instancia de base de datos en Amazon CloudWatch. Puede exportar el panel de métricas como un panel nuevo o añadirlo a un panel de CloudWatch existente. Si decide añadir el panel a un panel de CloudWatch existente, puede crear una etiqueta de encabezado para que las métricas aparezcan en una sección independiente del panel de CloudWatch.

Puede ver el panel de métricas exportado en la consola de CloudWatch. Si añade nuevas métricas a un panel de métricas de Información sobre rendimiento después de exportarlo, debe exportar de nuevo ese panel para ver las nuevas métricas en la consola de CloudWatch.

También puede seleccionar un widget de métricas en el panel de Información sobre rendimiento y ver los datos de las métricas en la consola de CloudWatch.

Para obtener más información sobre cómo ver métricas en la consola de CloudWatch, consulte Visualización de las métricas de el clúster de base de datos en la consola de CloudWatch y la AWS CLI.

En las siguientes secciones, exporte las métricas de Información de rendimiento a CloudWatch como un nuevo panel o a un panel existente, y consulte las métricas de Información de rendimiento en CloudWatch.

Temas

- Exportación de métricas de Información sobre rendimiento como nuevo panel a CloudWatch
- Añadir métricas de Información sobre rendimiento a un panel de CloudWatch existente
- Visualización de un widget de métricas de Información sobre rendimiento en CloudWatch

Exportación de métricas de Información sobre rendimiento como nuevo panel a CloudWatch

Elija un panel de métricas preconfigurado o personalizado del panel de Información sobre rendimiento y expórtelo como un panel nuevo a CloudWatch. Puede ver el panel exportado en la consola de CloudWatch.

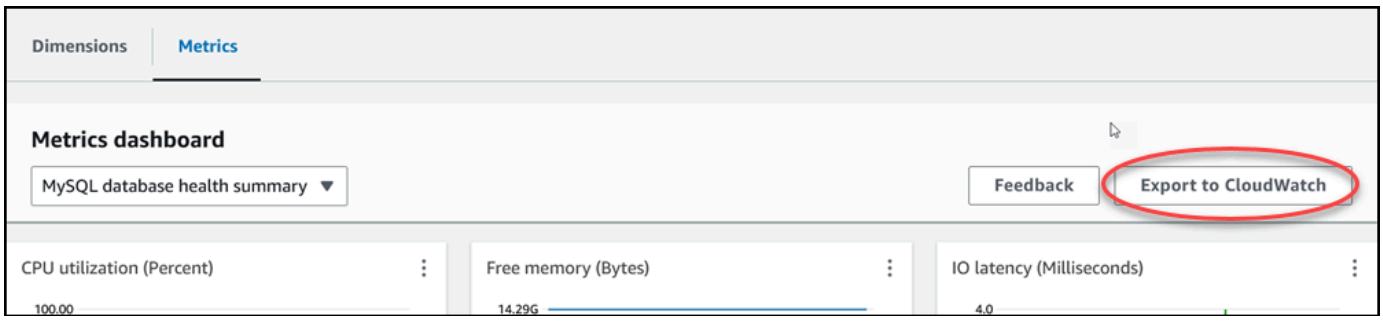
Exportación de un panel de métricas de Información sobre rendimiento como nuevo panel a CloudWatch

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.

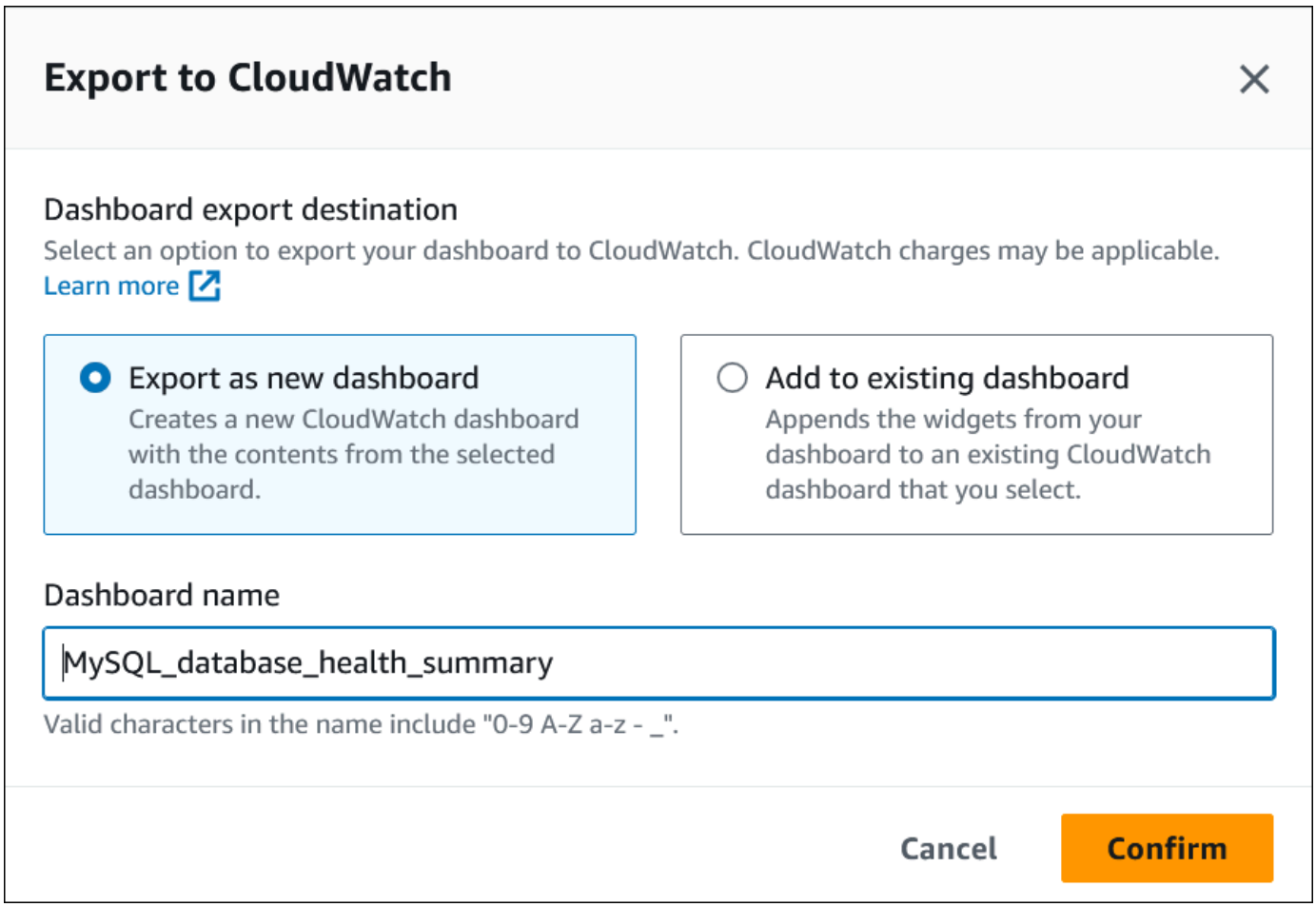
   Se muestra el panel de Información de rendimiento para la instancia de base de datos.
4. Vaya hacia abajo y seleccione Métricas.

   De forma predeterminada, aparece el panel preconfigurado con las métricas de Información sobre rendimiento.
5. Elija un panel preconfigurado o personalizado y, a continuación, seleccione Exportar a CloudWatch.

   Aparecerá la ventana Exportar a CloudWatch.

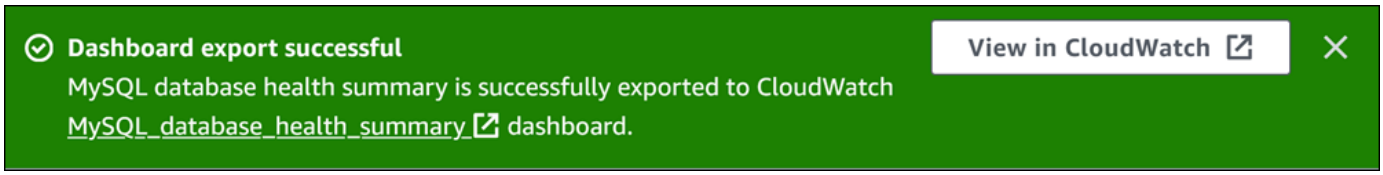


6. Seleccione Exportar como panel nuevo.



7. Introduzca un nombre para el nuevo panel en el campo Nombre del panel y seleccione Confirmar.

   Aparecerá un banner con un mensaje cuando el panel se haya exportado correctamente.



8. Seleccione el enlace o Ver en CloudWatch en el banner para ver el panel de métricas en la consola de CloudWatch.

Añadir métricas de Información sobre rendimiento a un panel de CloudWatch existente

Añada un panel de métricas preconfigurado o personalizado a un panel de CloudWatch existente. Puede añadir una etiqueta al panel de métricas para que aparezca en una sección independiente del panel de CloudWatch.

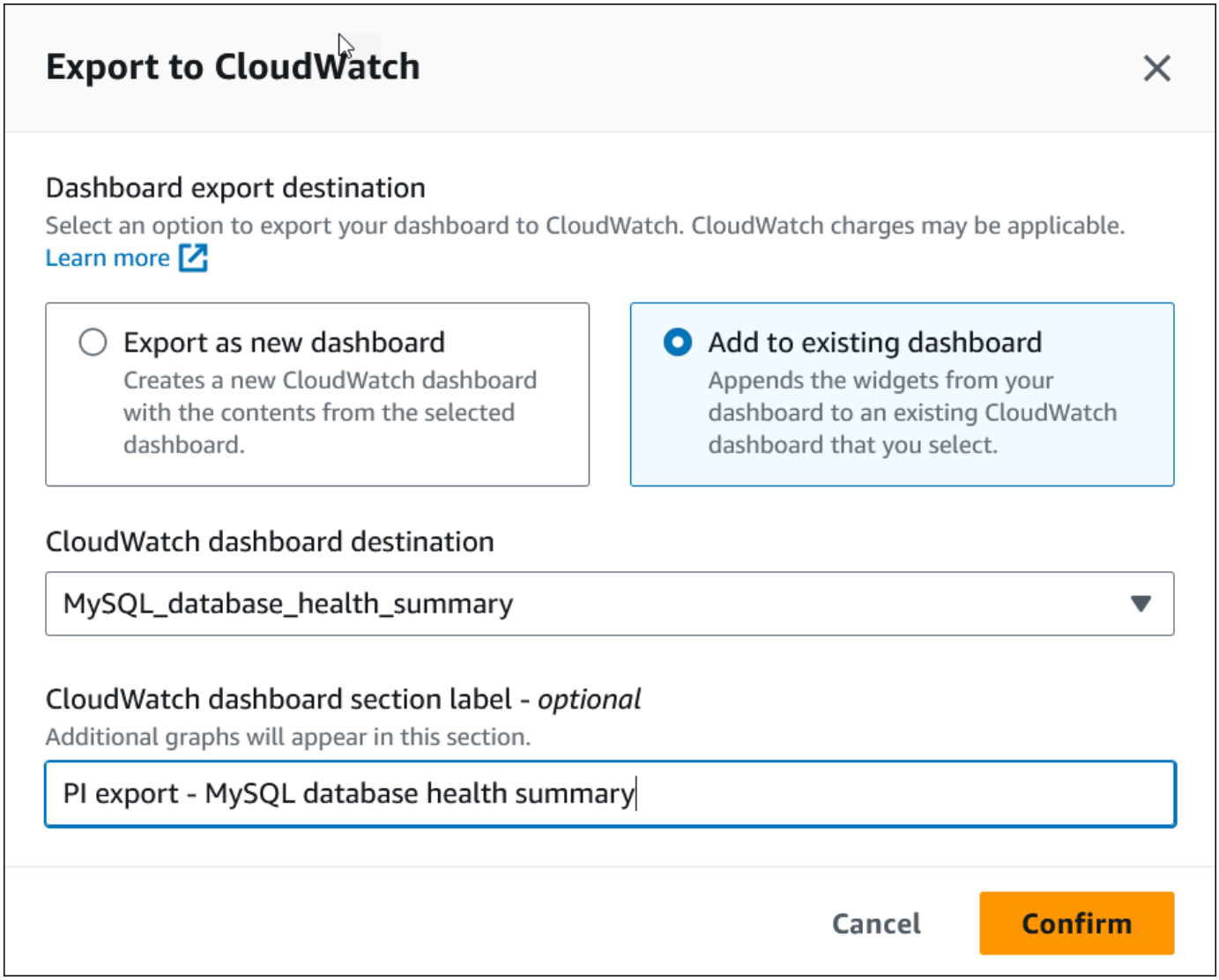
Exportación de métricas a un panel de CloudWatch existente

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.

   Se muestra el panel de Información de rendimiento para la instancia de base de datos.
4. Vaya hacia abajo y seleccione Métricas.

   De forma predeterminada, aparece el panel preconfigurado con las métricas de Información sobre rendimiento.
5. Elija un panel preconfigurado o personalizado y, a continuación, seleccione Exportar a CloudWatch.

   Aparecerá la ventana Exportar a CloudWatch.
6. Seleccione Agregar al panel existente.



7. Especifique el destino y la etiqueta del panel y, a continuación, elija Confirmar.

   - Destino del panel de CloudWatch: elija un panel de CloudWatch existente.
   - Etiqueta de la sección del panel de CloudWatch (opcional): introduzca un nombre para que las métricas de Información sobre rendimiento aparezcan en esta sección del panel de CloudWatch.

   Aparecerá un banner con un mensaje cuando el panel se haya exportado correctamente.

8. Seleccione el enlace o Ver en CloudWatch en el banner para ver el panel de métricas en la consola de CloudWatch.

Visualización de un widget de métricas de Información sobre rendimiento en CloudWatch

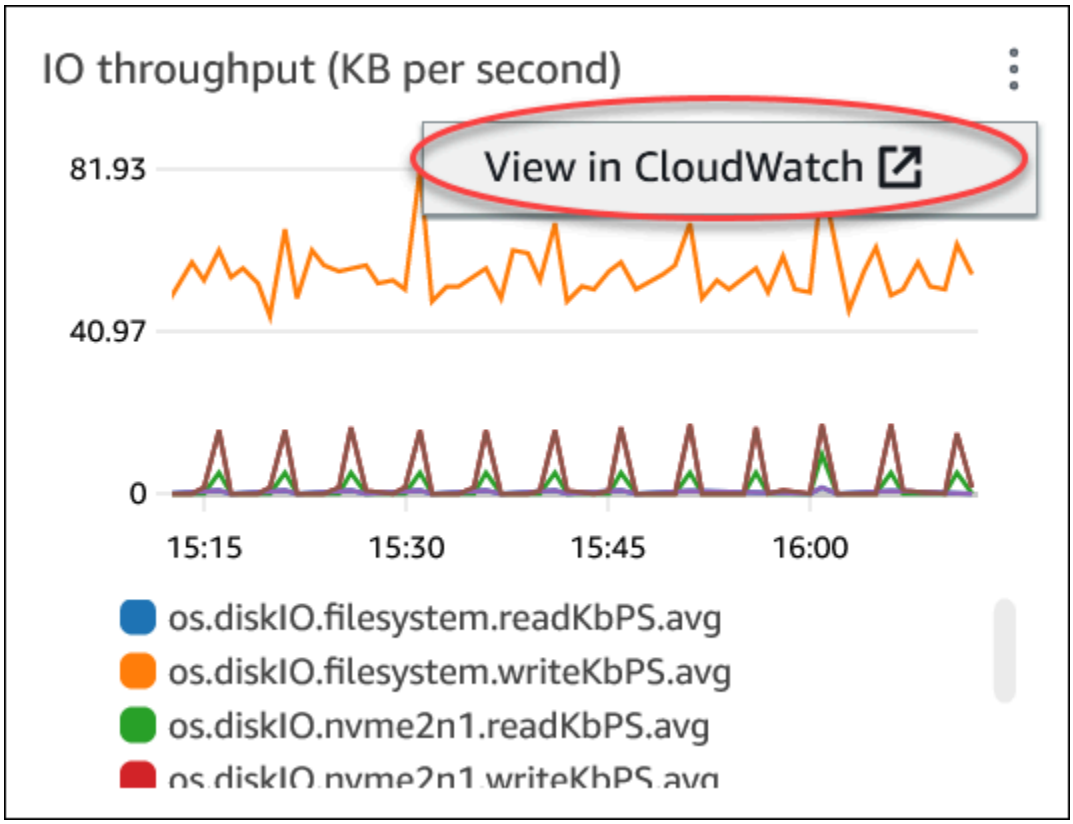
Seleccione un widget de métricas de Información sobre rendimiento en el panel de información de rendimiento de Amazon RDS y consulte los datos de métricas en la consola de CloudWatch.

Exportación de un widget de métricas y visualización de los datos de métricas en la consola de CloudWatch

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.

   Se muestra el panel de Información de rendimiento para la instancia de base de datos.
4. Vaya hacia abajo hasta llegar a Métricas.

   De forma predeterminada, aparece el panel preconfigurado con las métricas de Información sobre rendimiento.
5. Seleccione un widget de métricas y, a continuación, seleccione Ver en CloudWatch en el menú.



Los datos de métricas aparecen en la consola de CloudWatch.

Creación de alarmas de CloudWatch para supervisar Amazon Aurora

Puede crear una alarma de CloudWatch que envíe un mensaje de Amazon SNS cuando la alarma cambie de estado. Una alarma vigila una métrica determinada durante el periodo especificado. La alarma realiza una o varias acciones en función del valor de la métrica con respecto a un umbral determinado durante varios periodos de tiempo. La acción es una notificación que se envía a un tema de Amazon SNS o a una política de Amazon EC2 Auto Scaling.

Las alarmas invocan acciones únicamente para los cambios de estado prolongados. Las alarmas de CloudWatch no invocan acciones simplemente porque estén en un estado particular. El estado debe haber cambiado y debe haberse mantenido durante el número de periodos de tiempo especificado.

> **Note**
>
> Para Aurora, use las métricas de rol `WRITER` o `READER` para configurar alarmas en lugar de depender de métricas para instancias de base de datos específicas. Los roles de instancia de base de datos Aurora pueden cambiar roles con el tiempo. Puede encontrar estas métricas basadas en roles en la consola de CloudWatch.
> Aurora Auto Scaling establece automáticamente alarmas en función de las métricas de rol de `READER`. Para obtener más información sobre Aurora Auto Scaling, consulte Amazon Aurora Auto Scaling con réplicas de Aurora.

Puede utilizar la función matemática de métricas DB_PERF_INSIGHTS de la consola de CloudWatch para consultar Amazon RDS para conocer las métricas de los contadores de Información sobre rendimiento. La función DB_PERF_INSIGHTS también incluye la métrica DBLoad en intervalos de menos de un minuto. Puede establecer alarmas de CloudWatch sobre estas métricas.

Para obtener más información sobre cómo crear una alarma, consulte Create an alarm on Performance Insights counter metrics from an AWS database (Crear una alarma en las métricas de contador de Información sobre rendimiento desde una base de datos de AWS).

Para configurar una alarma mediante la AWS CLI

- Llame a `put-metric-alarm`. Para obtener más información, consulte la referencia de comandos de la AWS CLI.

Para configurar una alarma mediante la API de CloudWatch

- Llame a `PutMetricAlarm`. Para obtener más información, consulte la referencia de la API de Amazon CloudWatch.

Para obtener más información sobre la configuración de los temas de Amazon SNS y la creación de alarmas, consulte Uso de las alarmas de Amazon CloudWatch.

Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch

Supervise la carga de la base de datos (carga de DB) de su flota de instancias de bases de datos de Amazon Aurora con Información sobre las bases de datos. La carga de la base de datos mide el nivel de actividad de la sesión en la base de datos. Puede utilizar Información sobre las bases de datos para analizar y solucionar problemas del rendimiento de las bases de datos de Amazon Aurora a escala.

Con Información sobre las bases de datos, puede visualizar la carga de la base de datos en su flota de bases de datos y filtrar la carga por esperas, instrucciones SQL, hosts o usuarios.

De forma predeterminada, RDS habilita el modo estándar de Información sobre las bases de datos para las bases de datos de Amazon Aurora. Al activar el modo avanzado de Información sobre las bases de datos para un clúster de bases de datos, RDS habilita Información sobre las bases de datos para cada instancia de base de datos del clúster.

Para obtener información sobre el uso de Información sobre las bases de datos en la consola de Amazon CloudWatch, consulte Información sobre bases de datos de CloudWatch en la Guía del usuario de Amazon CloudWatch.

Precios

Para obtener información acerca de los precios, consulte Precios de Amazon CloudWatch.

Temas

- Compatibilidad con el motor de base de datos de Amazon Aurora, la región y la clase de instancia para Información sobre las bases de datos
- Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora
- Activación del modo estándar de Información sobre las bases de datos para Amazon Aurora
- Configuración de la base de datos para supervisar las consultas SQL lentas con Información sobre las bases de datos para Amazon Aurora
- Consideraciones acerca de Información sobre las bases de datos para Amazon Aurora

Compatibilidad con el motor de base de datos de Amazon Aurora, la región y la clase de instancia para Información sobre las bases de datos

Las siguiente tabla indica los motores de base de datos de Amazon Aurora que admiten la Información sobre las bases de datos.

Motor de base de datos de Amazon Aurora	Versiones de motor y regiones compatibles	Restricciones de clase de instancia
Amazon Aurora MySQL-Compatible Edition	Para obtener más informaci ón sobre la disponibilidad en versiones y regiones de Informaci ón sobre las bases de datos con Aurora MySQL, consulte Performance Insights con Aurora MySQL.	Información sobre las bases de datos tiene las siguientes restricciones de clase de motor: • db.t2: no se admite • db.t3: no se admite • db.t4g.micro y db.t4g.small: no se admite
Edición de Amazon Aurora compatible con PostgreSQL	Para obtener más informaci ón sobre la disponibilidad en versiones y regiones de Informaci ón sobre las bases de datos con Aurora PostgreSQL, consulte Performance Insights con Aurora PostgreSQL.	No aplicable
Base de datos ilimitada de Aurora PostgreSQL	Para obtener más información acerca de cómo usar informaci ón sobre la base de datos con Base de datos ilimitada de Aurora PostgreSQL, consulte Supervisi ón de la base de datos ilimitada de Aurora PostgreSQL con	No aplicable

Motor de base de datos de Amazon Aurora	Versiones de motor y regiones compatibles	Restricciones de clase de instancia
	información de base de datos de CloudWatch.	

Información sobre las bases de datos es compatible con Amazon Aurora Serverless v2.

Compatibilidad con el motor de base de datos de Amazon Aurora, la región y la clase de instancia para características de información sobre las bases de datos

Las siguiente tabla indica los motores de base de datos de Amazon Aurora que admiten las características de la información sobre las bases de datos.

Característica	Niveles de precios	Regiones admitidas	Motores de bases de datos compatibles	Clases de instancias admitidas
Estadísticas de SQL para Performance Insights	Todos	Todos	Todos	Todos
Análisis del rendimiento de la base de datos durante un período de tiempo	Solo nivel de pago	Todos	Todos	Todos excepto db.serverless (Aurora Serverless v2)
Visualización de las recomendaciones proactivas de Información de rendimiento	Solo nivel de pago	• Este de EE. UU. (Ohio) • Este de EE. UU. (Norte de Virginia)	Todos	Todos excepto db.serverless (Aurora Serverless v2)

Característica	Niveles de precios	Regiones admitidas	Motores de bases de datos compatibles	Clases de instancias admitidas
		• Oeste de EE. UU. (Norte de California) • Oeste de EE. UU. (Oregón) • Asia-Pacífico (Bombay) • Asia-Pacífico (Seúl) • Asia-Pacífico (Singapur) • Asia-Pacífico (Sídney) • Asia-Pacífico (Tokio) • Canadá (centro) • Europa (Fráncfort) • Europa (Irlanda) • Europa (Londres) • Europa (París) • Europa (Estocolmo) • América del Sur (São Paulo)		

Compatibilidad con la región de Amazon Aurora para información sobre las bases de datos

Aurora admite Información sobre las bases de datos en las siguientes Regiones de AWS.

- Este de EE. UU. (Norte de Virginia)
- Este de EE. UU. (Ohio)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- África (Ciudad del Cabo)
- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Hyderabad)
- Asia-Pacífico (Yakarta)
- Asia-Pacífico (Malasia)
- Asia-Pacífico (Melbourne)
- Asia-Pacífico (Bombay)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Oeste de Canadá (Calgary)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milán)
- Europa (París)
- Europa (España)
- Europa (Estocolmo)

- Europa (Zúrich)
- Israel (Tel Aviv)
- Medio Oriente (Baréin)
- Medio Oriente (EAU)
- América del Sur (São Paulo)
- AWS GovCloud (Este de EE. UU.)
- AWS GovCloud (Oeste de EE. UU.)

Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora

Para activar el modo avanzado de Información sobre las bases de datos para Amazon Aurora, use estos procedimientos.

Activación del modo avanzado de Información sobre las bases de datos al crear un clúster de base de datos

Active el modo avanzado de Información sobre las bases de datos al crear una base de datos para Amazon Aurora.

Console

En la consola, puede activar el modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija Creación de base de datos.
4. En la sección Información sobre bases de datos, seleccione el Modo avanzado. A continuación, elija las siguientes opciones:

- Retención: el número de días durante los que se conservan los datos de Información de rendimiento. El periodo de retención debe ser de 15 meses para el modo avanzado de información sobre las bases de datos.
- AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.

5. Elija Creación de base de datos.

AWS CLI

Para activar el modo avanzado de la Información sobre las bases de datos al crear un clúster de bases de datos, llame al comando create-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--database-insights-mode advanced` para activar el modo avanzado de Información sobre las bases de datos.
- `--engine`: el motor de base de datos para el clúster de bases de datos.
- `--db-cluster-identifier`: el identificador del clúster de bases de datos .
- `--enable-performance-insights` para activar Información sobre rendimiento para Información sobre las bases de datos.
- `--performance-insights-retention-period`: es el periodo de retención de los datos del clúster de bases de datos Multi-AZ. Para activar Información sobre las bases de datos, el período de retención debe ser de al menos 465 días.

En el siguiente ejemplo se habilita el modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode advanced \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Para Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode advanced ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Para activar el modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos multi-AZ, especifique los siguientes parámetros para la operación de la API CreateDBCluster de Amazon RDS.

- De `DatabaseInsightsMode` a `advanced`
- De `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` en al menos 465 días

Activación del modo avanzado de Información sobre las bases de datos al modificar un clúster de base de datos

Active Información sobre las bases de datos al modificar una base de datos para Amazon Aurora. La modificación de un clúster de base de datos para habilitar el modo avanzado de información de bases de datos no provoca tiempo de inactividad.

> **Note**
>
> Para habilitar Información sobre las bases de datos, cada instancia de base de datos de un clúster de bases de datos debe tener la misma configuración de Información de rendimiento y Supervisión mejorada.

Console

En la consola, puede activar el modo avanzado de Información sobre las bases de datos al modificar un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo avanzado de Información sobre las bases de datos al modificar de un clúster de bases de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija un clúster de bases de datos y elija Modificar.
4. En la sección Información sobre bases de datos, seleccione el Modo avanzado. A continuación, elija las siguientes opciones:
   - Retención: el número de días durante los que se conservan los datos de Información de rendimiento. El periodo de retención debe ser de 15 meses para el modo avanzado de información sobre las bases de datos.
   - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Continuar.
6. En Programación de modificaciones, elija Aplicar inmediatamente. Si elige Aplicar durante el próximo periodo de mantenimiento programado, la base de datos ignora esta configuración y activa de inmediato el modo avanzado de Información sobre las bases de datos.
7. Seleccione Modificar instancia.

AWS CLI

Para activar el modo avanzado de Información sobre las bases de datos al modificar un clúster de bases de datos, llame al comando modify-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--database-insights-mode advanced` para activar el modo avanzado de Información sobre las bases de datos.
- `--db-cluster-identifier`: el identificador del clúster de bases de datos .
- `--enable-performance-insights` para activar Información sobre rendimiento para Información sobre las bases de datos.

- `--performance-insights-retention-period`: es el periodo de retención de los datos del clúster de bases de datos. Para activar el modo avanzado de Información sobre las bases de datos, el período de retención debe ser de al menos 465 días.

En el siguiente ejemplo se habilita el modo avanzado de Información sobre las bases de datos al modificar un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Para activar el modo avanzado de Información sobre las bases de datos al modificar un clúster de base de datos, especifique los siguientes parámetros para la operación de la API ModifyDBCluster de Amazon RDS.

- De `DatabaseInsightsMode` a `advanced`
- De `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` en al menos 465 días

Activación del modo estándar de Información sobre las bases de datos para Amazon Aurora

Para activar el modo estándar de Información sobre las bases de datos para Amazon Aurora, use estos procedimientos.

Activación del modo estándar de Información sobre las bases de datos al crear un clúster de base de datos

Active el modo estándar de Información sobre las bases de datos al crear una base de datos para Amazon Aurora.

Console

En la consola, puede activar el modo estándar de Información sobre las bases de datos al crear un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo estándar de Información sobre las bases de datos al crear con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija Creación de base de datos.
4. En la sección Información sobre las bases de datos, seleccione el Modo estándar. A continuación, seleccione una de las siguientes opciones para activar o desactivar la Información de rendimiento:
   - Para desactivar la Información de rendimiento, anule la selección de Habilitar Información sobre rendimiento.
   - Para activar la Información de rendimiento, seleccione Habilitar Información sobre rendimiento. Para configurar la Información sobre rendimiento, especifique las siguientes opciones:
     - Retención: el tiempo durante el que se conservan los datos de la Información de rendimiento. El período de retención debe ser de al menos 7 días.
     - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Creación de base de datos.

AWS CLI

Para activar el modo estándar de la Información sobre las bases de datos al crear un clúster de bases de datos, llame al comando create-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--database-insights-mode standard` para activar el modo estándar de Información sobre las bases de datos.
- `--engine`: el motor de base de datos para el clúster de bases de datos.
- `--db-cluster-identifier`: el identificador del clúster de bases de datos .
- `--enable-performance-insights` o `--no-enable-performance-insights` para habilitar o deshabilitar Información sobre rendimiento. Si especifica `--enable-performance-insights`, también debe especificar el `--performance-insights-retention-period`: el periodo de retención de los datos del clúster de bases de datos. El período de retención debe ser de al menos 7 días.

En el siguiente ejemplo se habilita el modo estándar de Información sobre las bases de datos e Información de rendimiento al crear un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Para Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

En el siguiente ejemplo se habilita el modo estándar de Información sobre las bases de datos y se deshabilita Información de rendimiento al crear un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Para Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Para activar el modo estándar de Información sobre las bases de datos al crear un clúster de base de datos, especifique los siguientes parámetros para la operación de la API CreatedBCluster de Amazon RDS.

- De `DatabaseInsightsMode` a `standard`
- De `EnablePerformanceInsights` a `True` o `False`. Si configura `EnablePerformanceInsights` como `True`, debe configurar `PerformanceInsightsRetentionPeriod` como al menos 7 días.

Activación del modo estándar de Información sobre las bases de datos al modificar un clúster de base de datos

Active el modo estándar de Información sobre las bases de datos al modificar una base de datos para Amazon Aurora. La modificación de un clúster de base de datos para habilitar el modo estándar de información de base de datos no provoca tiempo de inactividad.

> **Note**
>
> Para habilitar Información sobre las bases de datos, cada instancia de base de datos de un clúster de bases de datos debe tener la misma configuración de Información de rendimiento y Supervisión mejorada.

Console

En la consola, puede activar el modo estándar de Información sobre las bases de datos al modificar un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo estándar de Información sobre las bases de datos al modificar un clúster de bases de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija un clúster de bases de datos y elija Modificar.
4. En la sección Información sobre las bases de datos, seleccione el Modo estándar. A continuación, elija entre las siguientes opciones:
   - Para desactivar la Información de rendimiento, anule la selección de Habilitar Información sobre rendimiento.
   - Para activar la Información de rendimiento, seleccione Habilitar Información sobre rendimiento. Para configurar la Información sobre rendimiento, especifique las siguientes opciones:
     - Retención: el tiempo durante el que se conservan los datos de la Información de rendimiento. El período de retención debe ser de al menos 7 días.
     - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Continuar.
6. En Programación de modificaciones, elija Aplicar inmediatamente. Si elige Aplicar durante el próximo periodo de mantenimiento programado, la base de datos ignora esta configuración y activa de inmediato el modo estándar de Información sobre las bases de datos.
7. Seleccione Modificar instancia.

AWS CLI

Para activar el modo estándar de Información sobre las bases de datos al modificar un clúster de bases de datos, llame al comando modify-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--database-insights-mode standard` para activar el modo estándar de Información sobre las bases de datos.
- `--db-cluster-identifier`: el identificador del clúster de bases de datos .
- `--enable-performance-insights` o `--no-enable-performance-insights` para habilitar o deshabilitar Información sobre rendimiento. Si especifica `--enable-performance-insights`, también debe especificar el `--performance-insights-retention-period`: el periodo de retención de los datos del clúster de bases de datos Multi-AZ. El período de retención debe ser de al menos 7 días.

En el siguiente ejemplo se habilita el modo estándar de Información sobre las bases de datos y se habilita la Información de rendimiento al modificar un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

En el siguiente ejemplo se habilita el modo estándar de Información sobre las bases de datos y se deshabilita la Información de rendimiento al modificar un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Para activar el modo estándar de Información sobre las bases de datos al modificar un clúster de base de datos, especifique los siguientes parámetros para la operación de la API ModifyDBCluster de Amazon RDS.

- De `DatabaseInsightsMode` a `standard`
- De `EnablePerformanceInsights` a `True` o `False`. Si configura `EnablePerformanceInsights` como `True`, debe configurar `PerformanceInsightsRetentionPeriod` como al menos 7 días.

Configuración de la base de datos para supervisar las consultas SQL lentas con Información sobre las bases de datos para Amazon Aurora

Para supervisar las consultas SQL lentas de la base de datos, puede utilizar la sección Consultas SQL lentas del panel de Información sobre las bases de datos. Antes de configurar la base de datos para supervisar consultas SQL lentas, la sección Consultas SQL lentas está en blanco.

Para obtener más información sobre cómo supervisar consultas SQL lentas en el panel de Información sobre las bases de datos, consulte Visualización del panel de instancia de base de datos para Información sobre las bases de datos de CloudWatch en la Guía del usuario de Amazon CloudWatch.

Para configurar la base de datos para supervisar las consultas de SQL lentas con información de las bases de datos, complete los siguiente pasos:

1. Habilite la exportación de registros a registros de CloudWatch.

2. Cree o modifique el grupo de parámetros del clúster de base de datos para el clúster de base de datos.

Para obtener información sobre la configuración de las exportaciones de registros, consulte Publicación de registros de bases de datos en Registros de Amazon CloudWatch en la Guía del usuario de Amazon Aurora.

Para crear o modificar el grupo de parámetros del clúster de bases de datos, consulte los temas siguientes.

- Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora
- Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora

Amazon Aurora MySQL

Para configurar el clúster de bases de datos MySQL de Amazon Aurora para supervisar las consultas SQL lentas, defina los siguientes parámetros.

- `slow_query_log`: se establece como `1`.
- `long_query_time`: se establece como `1.0`.
- `log_output`: se establece como `FILE`.

Amazon Aurora PostgreSQL

Para configurar el clúster de bases de datos PostgreSQL de Amazon Aurora para supervisar las consultas SQL lentas, defina los siguientes parámetros. Tenga en cuenta que la configuración de estos parámetros puede reducir el rendimiento del clúster de base de datos.

- `log_min_duration_statement`: se establece como `1000`.
- `log_statement`: se establece como `none`.
- `log_destination`: se establece como `stderr`.

> **Note**
>
> Para Aurora MySQL, puede configurar el parámetro `long_query_time` con un detalle de 1 microsegundo. Por ejemplo, puede establecer este parámetro en `0.000001`. Dependiendo

de la cantidad de consultas en la instancia de la base de datos, el valor del parámetro `long_query_time` puede reducir el rendimiento. Comience con el valor `1.0` y ajústelo en función de la carga de trabajo. Cuando establece este parámetro en `0`, Información sobre las bases de datos registra todas las consultas.

Para obtener información sobre los registros de Aurora MySQL y Aurora PostgreSQL, consulte lo siguiente.

- Archivos de registro de base de datos de Aurora MySQL
- Archivos de registro de bases de datos de Aurora PostgreSQL

Consideraciones acerca de Información sobre las bases de datos para Amazon Aurora

A continuación, presentamos algunas consideraciones acerca de Información sobre las bases de datos para Amazon Aurora.

- No puede administrar Información sobre las bases de datos para una instancia de base de datos en un clúster de bases de datos.
- Para habilitar el modo avanzado de Información sobre las bases de datos, debe habilitar Información de rendimiento y establecer el período de retención de Información de rendimiento en 465 días (15 meses) como mínimo. No hay ningún costo adicional por establecer el periodo de retención de información sobre rendimiento en 15 meses, aparte del costo de la información de la base de datos. Para obtener más información sobre los precios de Database Insights, consulte Precios de Amazon CloudWatch.
- Para habilitar Información sobre las bases de datos, cada instancia de base de datos de un clúster de bases de datos debe tener la misma configuración de Información de rendimiento y Supervisión mejorada.
- La modificación de un clúster de base de datos para habilitar el modo de información de base de datos no provoca tiempo de inactividad.

Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.
Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.
Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Performance Insights amplía las características de supervisión existentes de Amazon Aurora para ilustrar y ayudarlo a analizar el rendimiento del clúster. En el panel de Performance Insights puede visualizar la carga de la base de datos en su carga de clúster de Amazon Aurora y filtrarla por esperas, instrucciones SQL, hosts o usuarios. Para obtener más información sobre el uso de

Performance Insights con Amazon DocumentDB, consulte la Guía para desarrolladores de Amazon DocumentDB.

Temas

- Uso de Performance Insights en Amazon Aurora
- Activación y desactivación de Información de rendimiento de Aurora
- Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL
- Configuración de directivas de acceso para información sobre rendimiento
- Análisis de métricas mediante el panel de Información sobre rendimiento
- Visualización de las recomendaciones proactivas de Información de rendimiento
- Recuperación de métricas con la API de Información sobre rendimiento para Aurora
- Registro de llamadas de Performance Insights mediante el uso de AWS CloudTrail
- API de Información de rendimiento y puntos de conexión de VPC de interfaz (AWS PrivateLink)

Uso de Performance Insights en Amazon Aurora

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.
Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS.

Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.
Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

De forma predeterminada, RDS habilita Información sobre rendimiento en el asistente de creación de la consola para todos los motores de Amazon RDS. Si activa Información sobre rendimiento en el nivel de clúster de base de datos, RDS habilita Información sobre rendimiento para cada instancia de base de datos del clúster. Si tiene más de una base de datos en una instancia de base de datos, Performance Insights agrega datos de rendimiento.

Encontrará información general sobre Performance Insights para Amazon Aurora en el siguiente vídeo.

Uso de Performance Insights para analizar el rendimiento de Amazon Aurora PostgreSQL

Temas

- Carga de base de datos
- Máximo de la CPU
- El motor de base de datos de Amazon Aurora, la región y la clase de instancia son compatibles con Información de rendimiento
- Precios y retención de datos de Performance Insights

Carga de base de datos

La carga de base de datos mide el nivel de actividad de la sesión en la base de datos. `DBLoad` es la métrica clave de Información sobre rendimiento y Información sobre rendimiento recopila la carga de la base de datos cada segundo.

Temas

- Sesiones activas
- Sesiones activas promedio

- Ejecuciones activas promedio
- Dimensiones

Sesiones activas

Una sesión de base de datos representa el diálogo de una aplicación con una base de datos relacional. Una sesión activa es una conexión que ha enviado trabajo al motor de base de datos y está esperando una respuesta.

Una sesión está activa cuando se ejecuta en la CPU o a la espera de que un recurso esté disponible para que pueda continuar. Por ejemplo, una sesión activa puede esperar a que se lea una página (o bloque) en la memoria y, a continuación, consumir CPU mientras lee los datos de la página.

Sesiones activas promedio

El promedio de sesiones activas (AAS) es la unidad para la métrica de `DBLoad` en Performance Insights. Mide cuántas sesiones están activas simultáneamente en la base de datos.

Cada segundo, Información de rendimiento muestra el número de sesiones que ejecutan una consulta simultáneamente. Para cada sesión activa, Información de rendimiento recopila los siguientes datos:

- Instrucción SQL
- Estado de la sesión (en ejecución en la CPU o en espera)
- Host
- Usuario que ejecuta el SQL

Información de rendimiento calcula el AAS que se obtiene dividiendo el número total de sesiones entre el número total de ejemplos de un periodo de tiempo específico. Por ejemplo, en la tabla siguiente se muestran 5 ejemplos consecutivos de una consulta en ejecución realizada a intervalos de 1 segundo.

Ejemplo	Número de sesiones que ejecutan una consulta	AAS	Cálculo
1	2	2.	2 sesiones en total / 1 ejemplo

Ejemplo	Número de sesiones que ejecutan una consulta	AAS	Cálculo
2	0	1	2 sesiones en total / 2 ejemplos
3	4	2	6 sesiones en total / 3 ejemplos
4	0	1.5	6 sesiones en total / 4 ejemplos
5	4	2	10 sesiones en total / 5 ejemplos

En el ejemplo anterior, la carga de base de datos para el intervalo de tiempo es de 2 AAS. Esta medición significa que, de media, 2 sesiones han estado activas a la vez durante el plazo en que se han tomado las 5 muestras.

Ejecuciones activas promedio

Las ejecuciones activas promedio (AAE) por segundo están relacionadas con las sesiones activas promedio. Para calcular el AAE, Performance Insights divide el tiempo total de ejecución de una consulta por el intervalo de tiempo. En la tabla siguiente se muestra el cálculo de AAE para la misma consulta de la tabla anterior.

Tiempo transcurrido (en segundos)	Tiempo total de ejecución (en segundos)	AAE	Cálculo
60	120	2	120 segundos de ejecución/60 segundos transcurridos
120	120	1	120 segundos de ejecución/120 segundos transcurridos

Tiempo transcurrido (en segundos)	Tiempo total de ejecución (en segundos)	AAE	Cálculo
180	380	2.11	380 segundos de ejecución/180 segundos transcurridos
240	380	1.58	380 segundos de ejecución/240 segundos transcurridos
300	600	2	600 segundos de ejecución/300 segundos transcurridos

En la mayoría de los casos, el AAS y el AAE de una consulta dan aproximadamente lo mismo. Sin embargo, dado que las entradas de los cálculos son diferentes orígenes de datos, los cálculos suelen variar ligeramente.

Dimensiones

La métrica `db.load` es distinta de las demás métricas de series temporales porque puede desglosarla en subcomponentes llamados dimensiones. Las dimensiones son una especie de categorías “dividir por” de las diferentes características de la métrica `DBLoad`.

Cuando se diagnostican problemas de rendimiento, las siguientes dimensiones suelen ser las más útiles:

Temas

- Eventos de espera
- SQL principal

Para obtener una lista completa de las dimensiones de los motores de Aurora, consulte Carga de base de datos dividida por dimensiones.

Eventos de espera

Un evento de espera hace que una instrucción SQL espere a que ocurra un evento específico antes de que pueda continuar ejecutándose. Los eventos de espera son una dimensión o categoría importante de la carga de base de datos, porque indican dónde se ve obstaculizado el trabajo.

Cada sesión activa se ejecuta en la CPU o en espera. Por ejemplo, las sesiones consumen CPU cuando buscan memoria para un búfer, llevan a cabo un cálculo o ejecutan código de procedimiento. Cuando las sesiones no consumen CPU, pueden estar en espera de que se libere un búfer de memoria, se lea un archivo de datos o se escriba un registro. Cuanto más tiempo espere una sesión por los recursos, menos tiempo se ejecutará en la CPU.

Cuando ajusta una base de datos, a menudo intenta averiguar los recursos que esperan las sesiones. Por ejemplo, dos o tres eventos de espera podrían representar el 90 por ciento de la carga de base de datos. Esta medida significa que, en promedio, las sesiones activas pasan la mayor parte del tiempo en espera de un pequeño número de recursos. Si puede averiguar la causa de estas esperas, puede intentar una solución.

Los eventos de espera varían en función del motor de base de datos:

- Para ver una lista de los eventos de espera de Aurora MySQL que se utilizan con más frecuencia, consulte Eventos de espera de Aurora MySQL. Para obtener información sobre cómo ajustar con estos eventos de espera, consulte Ajuste de Aurora MySQL.
- Para obtener más información sobre todos los MySQL, consulte Wait Event Summary Tables (Tablas de resumen de eventos de espera) en la documentación de MySQL.
- Para ver una lista de los eventos de espera de Aurora PostgreSQL que se utilizan con más frecuencia, consulte Eventos de espera de Amazon Aurora PostgreSQL. Para obtener información sobre cómo ajustar con estos eventos de espera, consulte Ajuste con eventos de espera de Aurora PostgreSQL.
- Para obtener más información sobre todos los eventos de espera de PostgreSQL, consulte Ecentos de espera de PostgreSQL en la documentación de PostgreSQL.

SQL principal

Mientras que los eventos de espera muestran los cuellos de botella, la dimensión SQL principal indica qué consultas contribuyen más a la carga de base de datos. Por ejemplo, es posible que, aunque haya muchas consultas ejecutándose actualmente en la base de datos, una de ellas

consuma el 99 % de la carga de base de datos. En este caso, es posible que la carga alta indique un problema con la consulta.

De forma predeterminada, en la consola de Performance Insights se muestran las principales consultas SQL que contribuyen a la carga de la base de datos. En la consola se muestran también estadísticas importantes sobre cada instrucción. Para diagnosticar los problemas de rendimiento de una instrucción específica, puede examinar su plan de ejecución.

Máximo de la CPU

En el panel, el gráfico de Carga de base de datos recopila, agrega y muestra información de la sesión. Para ver si las sesiones activas superan el máximo de la CPU, observe su relación con la línea Máximo de la CPU virtual. Información sobre rendimiento determina el valor Máximo de la CPU virtual mediante el número de núcleos de vCPU (CPU virtual) de la instancia de base de datos. Para Aurora sin servidor v2, Max vCPU (vCPU máximo) representa el número estimado de vCPU.

Se puede ejecutar un proceso en una vCPU a la vez. Si el número de procesos supera el número de vCPU, los procesos comienzan a ponerse en cola. Cuando las colas aumentan, el rendimiento de la base de datos disminuye. Si la carga de base de datos suele estar por encima de la línea Máximo de la CPU virtual y el estado de espera principal es CPU, la CPU del sistema está sobrecargada. En este caso, quizá sea conveniente limitar las conexiones con la instancia, ajustar las consultas SQL con una carga de CPU alta o pensar en la posibilidad de usar una clase de instancia de mayor tamaño. Si hay instancias altas y uniformes en cualquier estado de espera, eso indica que es posible que haya problemas de contención de recursos o cuellos de botella que hay que resolver. Esto puede ser así aunque la carga de base de datos no cruce la línea de Máximo de la CPU virtual.

El motor de base de datos de Amazon Aurora, la región y la clase de instancia son compatibles con Información de rendimiento

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.
> Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes

del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.
Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.
Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Las siguiente tabla indica los motores de base de datos de Amazon Aurora que admiten la Información de rendimiento.

Motor de base de datos de Amazon Aurora	Versiones de motor y regiones compatibles	Restricciones de clase de instancia
Amazon Aurora MySQL-Compatible Edition	Para obtener más informaci ón sobre la disponibilidad en versiones y regiones de Performance Insights con Aurora MySQL, consulte Performance Insights con Aurora MySQL.	Performance Insights tiene las siguientes restricciones de clase de motor: • db.t2: no se admite • db.t3: no se admite • db.t4g.micro y db.t4g.small: no se admite

Motor de base de datos de Amazon Aurora	Versiones de motor y regiones compatibles	Restricciones de clase de instancia
Edición de Amazon Aurora compatible con PostgreSQL	Para obtener más información sobre la disponibilidad en versiones y regiones de Performance Insights con Aurora PostgreSQL, consulte Performance Insights con Aurora PostgreSQL.	N/A

Compatibilidad del motor de la base de datos, la región y la clase de instancia de Amazon Aurora con características de Información de rendimiento

Las siguiente tabla indica los motores de base de datos de Amazon Aurora que admiten características de Información de rendimiento.

Característica	Niveles de precios	Regiones admitidas	Motores de bases de datos compatibles	Clases de instancias admitidas
Estadísticas de SQL para Performance Insights	Todos	Todos	Todos	Todos
Análisis del rendimiento de la base de datos durante un período de tiempo	Solo nivel de pago	Todos	Todos	Todos excepto db.serverless (Aurora Serverless v2)
Visualización de las recomenda	Solo nivel de pago	• Este de EE. UU. (Ohio)	Todos	Todos excepto db.server

Característica	Niveles de precios	Regiones admitidas	Motores de bases de datos compatibles	Clases de instancias admitidas
ciones proactivas de Información de rendimiento		• Este de EE. UU. (Norte de Virginia) • Oeste de EE. UU. (Norte de California) • Oeste de EE. UU. (Oregón) • Asia-Pacífico (Bombay) • Asia-Pacífico (Seúl) • Asia-Pacífico (Singapur) • Asia-Pacífico (Sídney) • Asia-Pacífico (Tokio) • Canadá (centro) • Europa (Fráncfort) • Europa (Irlanda) • Europa (Londres) • Europa (París) • Europa (Estocolmo)		less (Aurora Serverless v2)

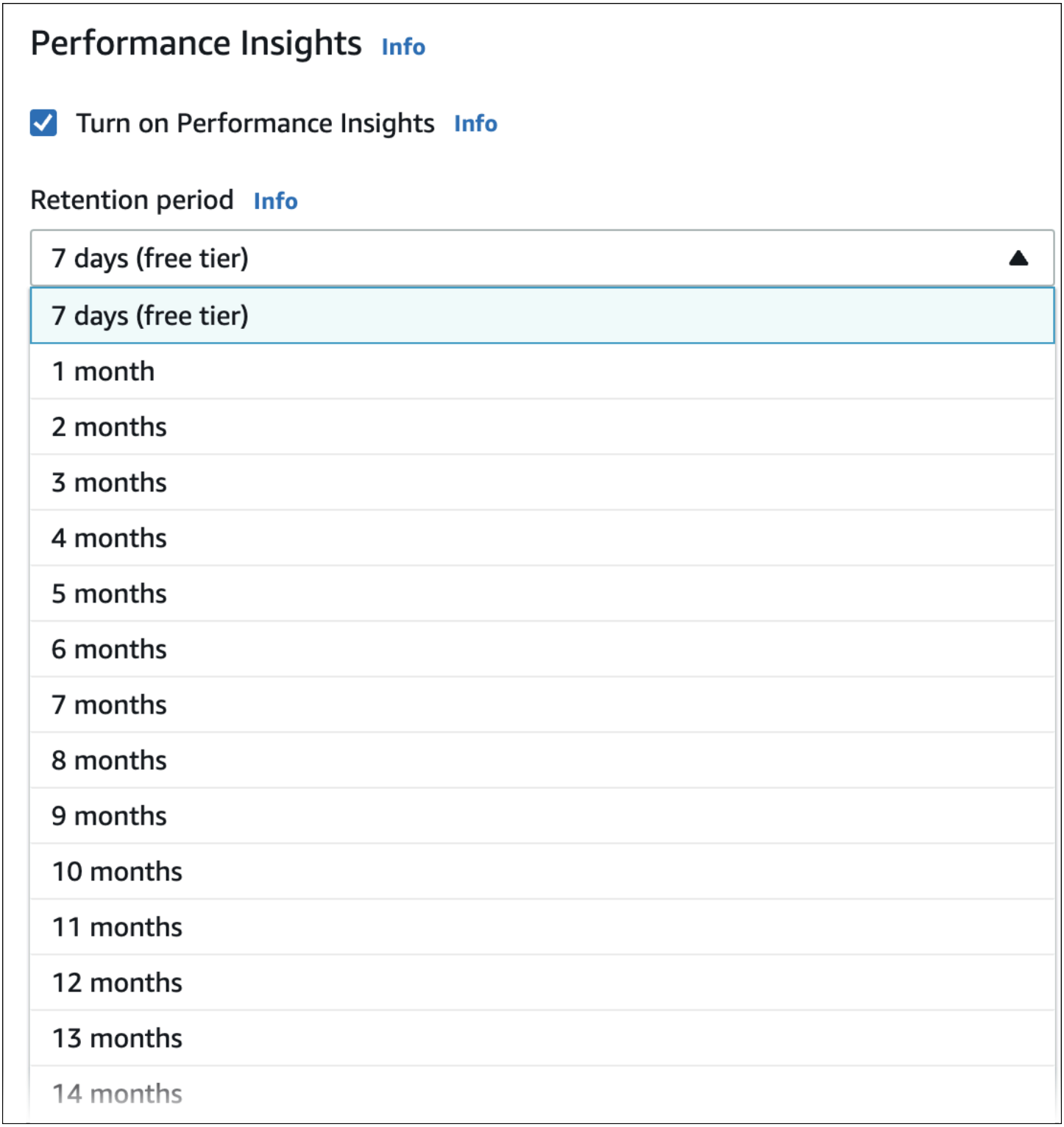
Característica	Niveles de precios	Regiones admitidas	Motores de bases de datos compatibles	Clases de instancias admitidas
		• América del Sur (São Paulo)		

Precios y retención de datos de Performance Insights

De forma predeterminada, Performance Insights ofrece una capa gratuita que incluye 7 días de historial de datos de rendimiento y 1 millón de solicitudes de API al mes. También puede comprar períodos de retención más largos. Para obtener información completa sobre los precios, consulte los precios de Performance Insights.

En la consola de RDS, puede elegir cualquiera de los siguientes períodos de retención de sus datos de Performance Insights:

- Predeterminado (7 días)
- ***n***meses, donde***n***es un número del 1 al 24



Para obtener información para configurar un período de retención con AWS CLI, consulte ???.

Activación y desactivación de Información de rendimiento de Aurora

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.

Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.

Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.

Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Puede activar la información sobre rendimiento para el clúster de base de datos durante su creación. Si es necesario, puede desactivarlo más adelante. Para ello, modifique el clúster de base de datos en la consola. La activación y desactivación de Información de rendimiento no provoca tiempo de inactividad, un reinicio ni una conmutación por error.

> **Note**
>
> Performance Schema es una herramienta de rendimiento opcional utilizada por Aurora MySQL. Si activa o desactiva Esquema de rendimiento, debe reiniciar. Sin embargo, si activa o desactiva Performance Insights, no es necesario que se reinicie. Para obtener más información, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

Si utiliza información sobre rendimiento con las bases de datos globales de Aurora, active información sobre rendimiento de forma individual para las bases de datos de cada Región de AWS. Para obtener más información, consulte Supervisión de una base de datos Amazon Aurora global con Amazon RDS la información sobre rendimiento.

El agente Performance Insights consume CPU y memoria limitadas en el host de base de datos. Cuando la carga de la base de datos es alta, el agente limita el impacto en el rendimiento mediante la recopilación de datos con menos frecuencia.

Console

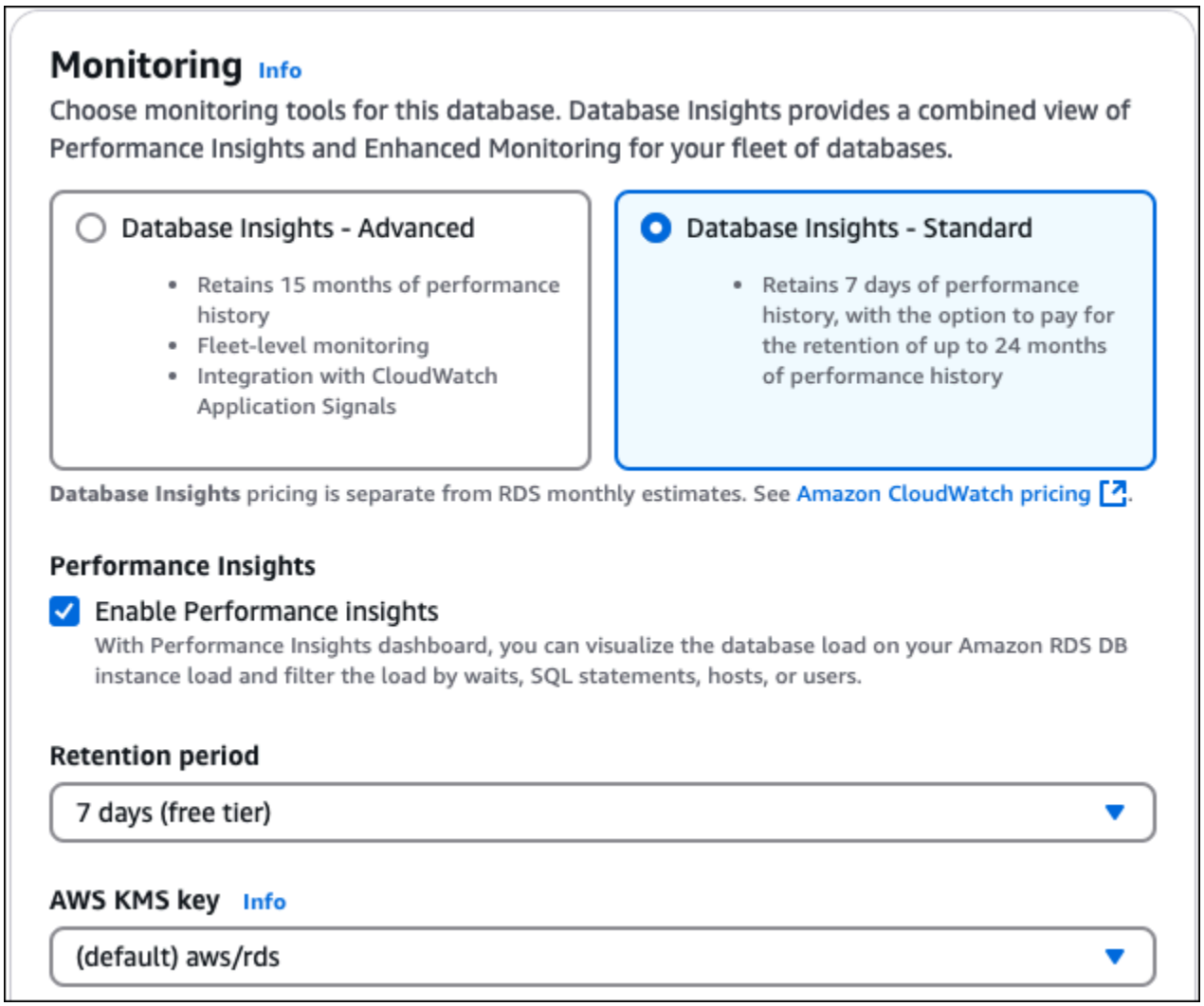
En la consola, puede activar o desactivar Información sobre rendimiento al crear un clúster de base de datos. La habilitación de información sobre rendimiento en el clúster le permite administrar la configuración y las opciones de información sobre rendimiento para el clúster de base de datos. La configuración del clúster se aplica a todas las instancias de base de datos en el clúster.

Activación o desactivación de Performance Insights al crear un clúster de base de datos

Tras crear un nuevo clúster de base de datos, Amazon RDS habilita información sobre rendimiento de forma predeterminada. Para desactivar información sobre rendimiento para el clúster de base de datos, elija la opción Información de base de datos: estándar y desmarque la opción Habilitar información sobre rendimiento.

Para crear un clúster de base de datos, siga las instrucciones para el motor de base de datos que se indican en Creación de un clúster de base de datos de Amazon Aurora.

La siguiente captura de pantalla muestra la sección Información de rendimiento.



Si elige Activar Información de rendimiento, dispondrá de las siguientes opciones:

- Retención (solo para el modo estándar de información sobre rendimiento): la cantidad de tiempo durante el que se retienen los datos de información sobre rendimiento. La configuración de retención es Predeterminada (7 días). Para retener los datos de rendimiento durante más tiempo, especifique de 1 a 24 meses. Para obtener más información acerca de los periodos de retención, consulte Precios y retención de datos de Performance Insights.
- AWS KMS key – especifique su AWS KMS key. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cambio de una política de AWS KMS para Información de rendimiento.

Activación o desactivación de información sobre rendimiento al modificar un clúster de base de datos

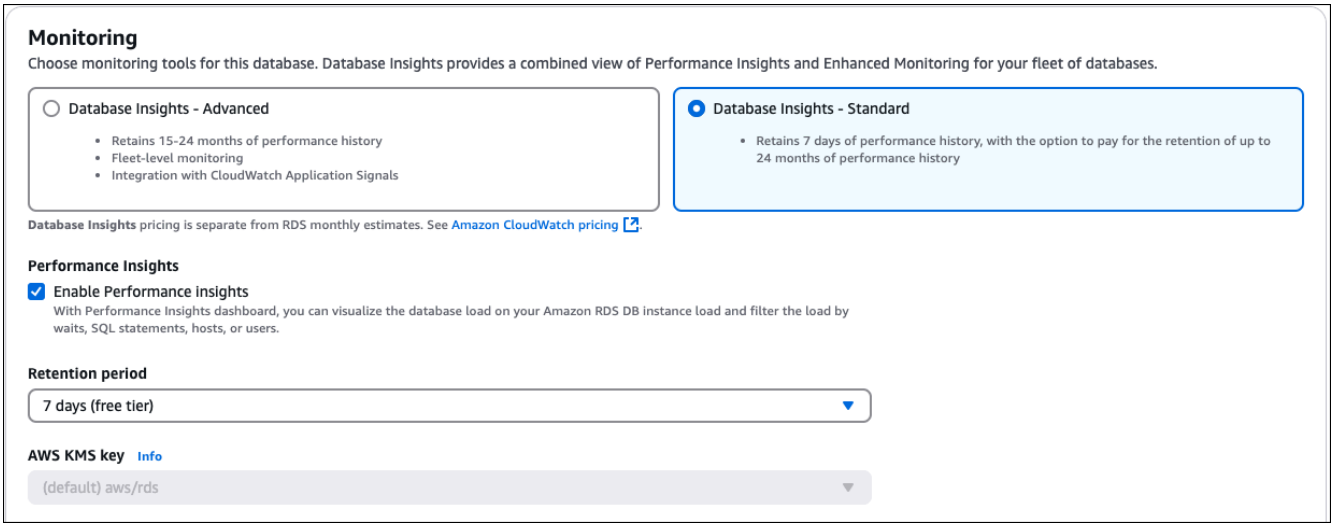
En la consola, puede modificar un clúster de base de datos para administrar información sobre rendimiento.

Para administrar información sobre rendimiento para una clúster de base de datos mediante la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija un clúster de base de datos y elija Modificar.
4. Para activar la Información de rendimiento, seleccione Habilitar Información sobre rendimiento. Para desactivar información sobre rendimiento para el clúster de base de datos, elija la opción Información de base de datos: estándar y desmarque la opción Habilitar información sobre rendimiento.

   Si elige Activar Información de rendimiento, dispondrá de las siguientes opciones:

   - Retención (solo para el modo estándar de información sobre rendimiento): la cantidad de tiempo durante el que se retienen los datos de información sobre rendimiento. La configuración de retención es Predeterminada (7 días). Para retener los datos de rendimiento durante más tiempo, especifique de 1 a 24 meses. Para obtener más información acerca de los periodos de retención, consulte Precios y retención de datos de Performance Insights.
   - AWS KMS key – especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.



5. Elija Continuar.
6. En Scheduling of Modifications (Programación de modificaciones) (Programación de modificaciones), elija Apply immediately (Aplicar inmediatamente). Si elige Apply during the next scheduled maintenance window (Aplicar durante la próxima ventana de mantenimiento programada), la instancia ignora esta configuración y activa de inmediato Performance Insights.
7. Elija Modificar instancia.

AWS CLI

Cuando utilice el comando create-db-instance de la AWS CLI, active información sobre rendimiento especificando `--enable-performance-insights` y estableciendo `--database-insights-mode` en `advanced` o `standard`. Para desactivar información sobre rendimiento, especifique `--no-enable-performance-insights` y establezca `database-insights-mode` en `standard`.

Estos valores también pueden especificarse con los siguientes comandos de AWS CLI:

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-replica
- modify-db-instance
- restore-db-instance-from-s3

Administración de Información de rendimiento en un clúster de base de datos con la AWS CLI

- Llame al comando AWS CLI de modify-db-clúster y suministre los siguientes valores:
  - `--db-cluster-identifier`: nombre de la instancia de base de datos en su clúster de base de datos.
  - `--enable-performance-insights` para activar o `--no-enable-performance-insights` para desactivar
  - `database-insights-mode`: el modo de información sobre base de datos para el clúster de base de datos. Para desactivar información sobre rendimiento, establezca este valor en `standard`.

  El siguiente ejemplo activa Información de rendimiento para `sample-db-cluster`.

  Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights
```

  Para Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights
```

Al activar Información de rendimiento en la CLI, puede especificar, si lo desea, el tiempo en días que se retienen los datos de Información de rendimiento mediante la opción `--performance-insights-retention-period`. Puede especificar7, *`mes`** 31 (donde *`mes`* es un número comprendido entre 1 y 23), o 731. Por ejemplo, si desea retener los datos de rendimiento durante 3 meses, especifique 93, que es 3 * 31. El valor predeterminado es 7 días. Para obtener más información acerca de los periodos de retención, consulte Precios y retención de datos de Performance Insights.

El ejemplo siguiente activa Información de rendimiento para `sample-db-cluster` y especifica que los datos de Información de rendimiento se retengan durante 93 días (3 meses).

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Para Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
    --performance-insights-retention-period 93
```

Si especifica un período de retención, como 94 días, que no es un valor válido, RDS emitirá un error.

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> Solo puede activar Información de rendimiento para una instancia de un clúster de base de datos en la que Información de rendimiento no se administre en el clúster.

RDS API

Si crea una nueva instancia de base de datos en el clúster de la base de datos mediante la operación CreateDBInstance de la operación de la API de Amazon RDS, active Información de rendimiento configurando `EnablePerformanceInsights` como `True`. Para desactivar información sobre rendimiento, establezca `EnablePerformanceInsights` en `False` y establezca `DatabaseInsightsMode` en `standard`.

También puede especificar el valor `EnablePerformanceInsights` utilizando las siguientes operaciones de la API:

- CreateDBCluster
- ModifyDBCluster
- ModifyDBInstance
- CreateDBInstanceReadReplica
- RestoreDBInstanceFromS3

Cuando activa Información de rendimiento, puede especificar, si lo desea, la cantidad de tiempo en días que se retienen los datos de Información de rendimiento con el parámetro `PerformanceInsightsRetentionPeriod`. Puede especificar7, *`mes`** 31 (donde *`mes`* es un número comprendido entre 1 y 23), o `731`. Por ejemplo, si desea retener los datos de rendimiento durante 3 meses, especifique `93`, que es 3 * 31. El valor predeterminado es 7 días. Para obtener más información acerca de los periodos de retención, consulte Precios y retención de datos de Performance Insights.

Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL

Performance Schema es una característica opcional para supervisar el rendimiento de tiempo de ejecución de Aurora MySQL con un nivel bajo de detalle. Performance Schema está diseñado para tener un impacto mínimo en el rendimiento de la base de datos. Performance Insights es una característica distinta que puede utilizar con o sin Performance Schema.

Temas

- Información general de Performance Schema
- Performance Insights y Performance Schema
- Administración automática de Performance Schema mediante Performance Insights
- Qué ocurre al activar Performance Schema
- Determinación de si Performance Insights está administrando Performance Schema
- Activación de Performance Schema para Aurora MySQL

Información general de Performance Schema

Performance Schema supervisa los eventos en las bases de datos Aurora MySQL. Un evento es una acción de servidor de base de datos que consume tiempo y se ha instrumentado para que se pueda recopilar información de tiempo. A continuación, se muestran ejemplos de eventos:

- Llamadas a funciones
- Esperas del sistema operativo
- Etapas de la ejecución de SQL
- Grupos de instrucciones SQL

El motor de almacenamiento `PERFORMANCE_SCHEMA` es un mecanismo para implementar la característica Performance Schema. El motor recopila datos de eventos mediante la instrumentación en el código fuente de la base de datos. El motor almacena eventos en tablas solo en la memoria en la base de datos de `performance_schema`. Puede consultar `performance_schema` al igual que puede consultar cualquier otra tabla. Para obtener más información, consulte MySQL Performance Schema en el Manual de referencia de MySQL.

Performance Insights y Performance Schema

Performance Insights y Performance Schema son características independientes, pero están conectadas. El comportamiento de Performance Insights para Aurora MySQL depende de si Performance Schema está activado y, de ser así, de si Performance Insights administra el Performance Schema automáticamente. La tabla siguiente describe el comportamiento.

Performance Schema activado	Modo de administración de Performance Insights	Comportamiento de Performance Insights
Sí	Automático	• Recopila información detallada de supervisión de bajo nivel • Recopila métricas de sesión activas cada segundo •

Performance Schema activado	Modo de administración de Performance Insights	Comportamiento de Performance Insights
		Muestra la carga de base de datos categorizada por eventos de espera detallados, que puede utilizar para identificar cuellos de botella
Sí	Manual	• Recopila eventos de espera y métricas por SQL • Recopila métricas de sesión activas cada segundo • Informa sobre estados de usuario, como insertar y enviar, que no ayudan a identificar cuellos de botella
No	N/A	• No recopila eventos de espera, métricas por SQL ni otra información de supervisión detallada y de bajo nivel • Recopila métricas de sesión activas cada cinco segundos en lugar de cada segundo • Informa sobre estados de usuario, como insertar y enviar, que no ayudan a identificar cuellos de botella

Administración automática de Performance Schema mediante Performance Insights

Al crear una instancia de base de datos de Aurora MySQL con Performance Insights activado, también se activa Performance Schema. En este caso, la Performance Insights administra automáticamente sus parámetros de Esquema de rendimiento. Esta es la configuración recomendada.

Si es así, Información de rendimiento administra automáticamente el Esquema de rendimiento, el Origen de `performance_schema` es `System default`.

> **Note**
>
> La clase de instancia t4g.medium no admite la administración automática del esquema de rendimiento.

También puede administrar manualmente Performance Schema. Si elige esta opción, debe definir los parámetros según los valores de la siguiente tabla.

Nombre del parámetro	Valor del parámetro
`performance_schema`	1 (la columna Source (Origen) tiene el valor `Modified`)
`performance-schema-consumer-events-waits-current`	`ON`
`performance-schema-instrument`	`wait/%=ON`
`performance_schema_consumer_global_instrumentation`	`1`
`performance_schema_consumer_thread_instrumentation`	`1`

Si cambia el valor del parámetro `performance_schema` manualmente y, más tarde, desea volver a la gestión automática, consulte Activación de Performance Schema para Aurora MySQL.

> **Important**
>
> Cuando Performance Insights activa Performance Schema, no cambia los valores del grupo de parámetros. Sin embargo, los valores se cambian en las instancias de base de datos que se están ejecutando. La única forma de ver los valores modificados es ejecutar elcomando `SHOW GLOBAL VARIABLES`.

Qué ocurre al activar Performance Schema

Performance Insights y Performance Schema tienen requisitos distintos para los reinicios de instancias de base de datos:

Performance Schema

Para activar o desactivar esta característica, debe reiniciar la instancia de base de datos.

Información de rendimiento

Para activar o desactivar esta característica, no es necesario reiniciar la instancia de base de datos.

Si Performance Schema no está activado actualmente y activa Performance Insights sin reiniciar la instancia de base de datos, Performance Schema no se activará.

Determinación de si Performance Insights está administrando Performance Schema

Para saber si Información de rendimiento está administrando actualmente Performance Schema en todas las versiones principales del motor, consulte la siguiente tabla.

Configuración del parámetro performance_schema	Configuración de la columna Source (Origen)	¿Performance Insights está administrando Performance Schema?
`0`	`System default`	Sí
`0` o `1`	`Modified`	No

En el siguiente procedimiento, determinará si Información de rendimiento está administrando Performance Schema de forma automática.

Para determinar si Performance Insights está administrando Performance Schema automáticamente

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Parameter groups (Grupos de parámetros).
3. Seleccione el grupo de parámetros para la instancia de base de datos.

4. Escriba **`performance_schema`** en la barra de búsqueda.
5. Compruebe si Origen es el valor predeterminado del sistema y si Valor es 0. Si es así, la Performance Insights administra automáticamente Performance Schema.

   En este ejemplo, Información de rendimiento no está administrando Performance Schema de forma automática.

Activación de Performance Schema para Aurora MySQL

Supongamos que Performance Insights está activado para su instancia de base de datos pero no está administrando actualmente Performance Schema. Si desea permitir que Performance Insights administre Performance Schema automáticamente, complete los siguientes pasos.

Para configurar Performance Schema para la administración automática

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Grupos de parámetros.
3. Seleccione el nombre del grupo de parámetros para la instancia de base de datos.
4. Seleccione Editar.
5. Escriba **`performance_schema`** en la barra de búsqueda.
6. Seleccione el parámetro `performance_schema`.
7. Seleccione Establecer en el valor predeterminado.
8. Para confirmar, seleccione Establecer los valores en su forma predeterminada.
9. Elija Guardar cambios.
10. Reinicie la instancia de base de datos.

> **Important**
>
> Al activar o desactivar Performance Schema, deberá reiniciar la instancia de base de datos.

Para obtener información sobre cómo modificar los parámetros de la instancia de base de datos, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora. Para obtener más información acerca del panel, consulte Análisis de métricas mediante el panel de Información sobre rendimiento. Para obtener más información sobre el esquema de rendimiento de MySQL, consulte MySQL Performance Schema (para 8.0) y MySQL Performance Schema (para 8.4) en la documentación de MySQL.

Configuración de directivas de acceso para información sobre rendimiento

Para acceder a Performance Insights, la entidad principal deberá tener los permisos adecuados de AWS Identity and Access Management (IAM).

> **Note**
>
> Para usar información de rendimiento con una clave administrada por el cliente, conceda a los usuarios los permisos `kms:Decrypt` y `kms:GenerateDataKey` para la clave de AWS AWS KMS.

Acceda a información de rendimiento mediante estos métodos:

- Asociación de la política administrada `AmazonRDSPerformanceInsightsReadOnly` para el acceso de solo lectura
- Asociación de la política administrada `AmazonRDSPerformanceInsightsFullAccess` para el acceso a todas las operaciones de la API de información de rendimiento
- Creación de una política de IAM personalizada con permisos específicos
- Configuración de permisos de AWS KMS para datos cifrados de información de rendimiento
- Configuración de un acceso detallado mediante permisos por recursos
- Uso del control de acceso basado en etiquetas para administrar los permisos mediante etiquetas de recursos

Asociación de la política AmazonRDSPerformanceInsightsReadOnly a una entidad principal de IAM

`AmazonRDSPerformanceInsightsReadOnly` es una política administrada de AWS que concede acceso a todas las operaciones de solo lectura de la API de Información sobre rendimiento de Amazon RDS.

Si adjunta `AmazonRDSPerformanceInsightsReadOnly` a un rol o conjunto de permisos, también debe asociar los siguientes permisos de CloudWatch:

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Con estos permisos, el destinatario puede utilizar la información de rendimiento con las demás características de la consola.

Para obtener más información sobre los permisos de CloudWatch, consulte la referencia de permisos de Amazon CloudWatch.

Para obtener más información acerca de `AmazonRDSPerformanceInsightsReadOnly`, consulte Política administrada por:AWS AmazonRDSPerformanceInsightsReadOnly.

Asociación de la política AmazonRDSPerformanceInsightsFullAccess a una entidad principal de IAM

`AmazonRDSPerformanceInsightsFullAccess` es una política administrada de AWS que concede acceso a todas las operaciones de la API de Información sobre rendimiento de Amazon RDS.

Si adjunta `AmazonRDSPerformanceInsightsFullAccess` a un rol o conjunto de permisos, también debe asociar los siguientes permisos de CloudWatch:

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Con estos permisos, el destinatario puede utilizar la información de rendimiento con las demás características de la consola.

Para obtener más información sobre los permisos de CloudWatch, consulte la referencia de permisos de Amazon CloudWatch.

Para obtener más información acerca de `AmazonRDSPerformanceInsightsFullAccess`, consulte Política administrada por AWS: AmazonRDSPerformanceInsightsFullAccess.

Creación de una política de IAM personalizada para la información sobre rendimiento

En el caso de usuarios que no tengan la política `AmazonRDSPerformanceInsightsReadOnly` o `AmazonRDSPerformanceInsightsFullAccess`, puede conceder acceso a Información sobre rendimiento a través de la creación o modificación de una política de IAM administrada por el usuario. Al asociar la política a un conjunto de permisos o a un rol, el destinatario puede utilizar Información de rendimiento.

Creación de una política personalizada

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Políticas.
3. Elija Create Policy (Crear política).
4. En la página Crear política, elija la opción JSON.
5. Copie y pegue el texto proporcionado en la sección Documento de política de JSON en la Guía de referencia de las políticas administradas de AWS para la política AmazonRDSPerformanceInsightsReadOnly o AmazonRDSPerformanceInsightsFullAccess.
6. Elija Review policy (Revisar política).
7. Proporcione un nombre para la política y, opcionalmente, una descripción, a continuación, elija Create policy (Crear política).

Ahora ya puede asociar la política a un conjunto de permisos o a un rol. En el procedimiento siguiente, se presupone que ya tiene un usuario para este fin.

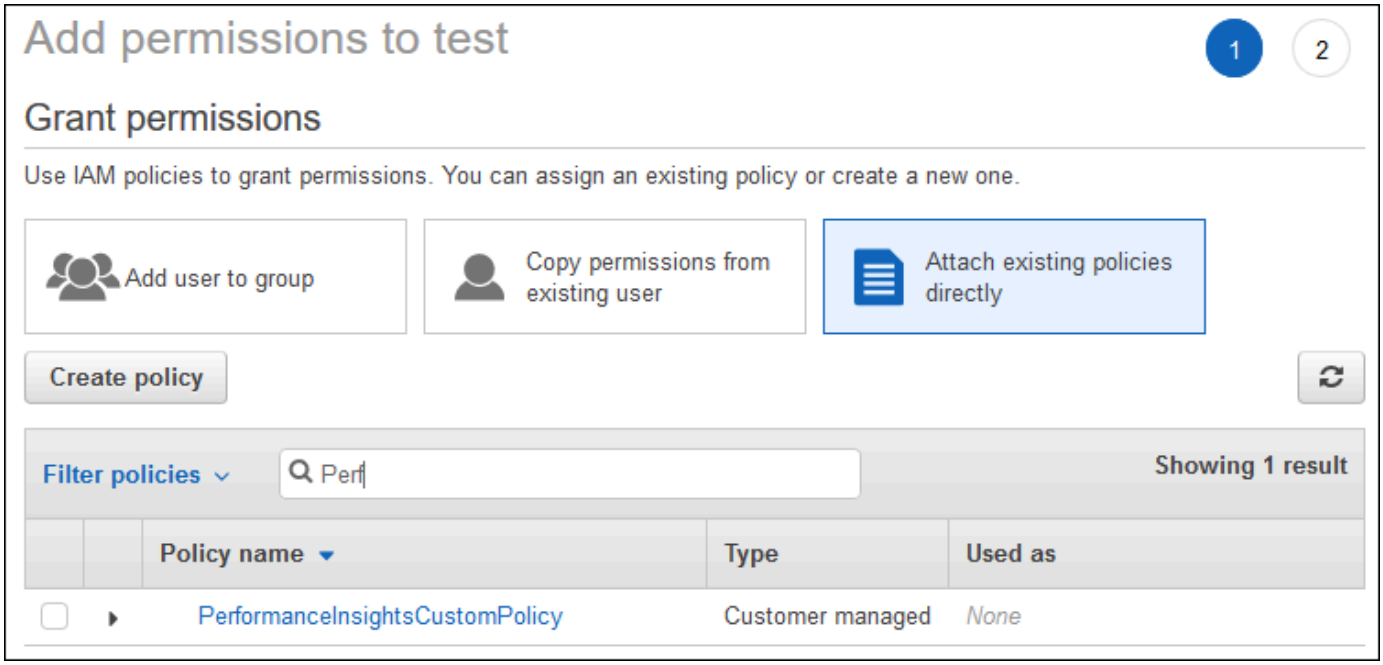
Asociación de la política a un usuario

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Users.
3. Elija en la lista un usuario existente.

> **Important**
>
> Para utilizar Performance Insights, asegúrese de tener acceso a Amazon RDS además de la política personalizada. Por ejemplo, la política predefinida `AmazonRDSPerformanceInsightsReadOnly` ofrece acceso de solo lectura a

Amazon RDS. Para obtener más información, consulte Administración de acceso mediante políticas.

4. En la página Summary, elija Add permissions.
5. Elija Asociar políticas existentes directamente. En Buscar, escriba los primeros caracteres del nombre de la política, como se muestra en la siguiente imagen.



6. Elija la política y, a continuación, elija Next: Review.
7. Elija Add permissions (Agregar permisos).

Cambio de una política de AWS KMS para Información de rendimiento

Performance Insights utiliza una AWS KMS key para cifrar información confidencial. Cuando active Performance Insights a través de la API o de la consola, podrá hacer una de las siguientes cosas:

- Elegir la predeterminada Clave administrada de AWS.

  Amazon RDS utiliza la Clave administrada de AWS para su nueva instancia de base de datos. Amazon RDS crea una Clave administrada de AWS para su cuenta de Cuenta de AWS. Su cuenta de Cuenta de AWS tiene una Clave administrada de AWS diferente para Amazon RDS para cada Región de AWS.
- Elija una clave administrada por el cliente.

Si especifica una clave administrada por el cliente, los usuarios de su cuenta que llamen a la API de Performance Insights necesitarán los permisos `kms:Decrypt` y `kms:GenerateDataKey` sobre la clave de KMS. Puede configurar estos permisos a través de directivas de IAM. Sin embargo, le recomendamos que administre estos permisos a través de su directiva de clave KMS. Para obtener más información, consulte Políticas de claves en AWS KMS en la Guía para desarrolladores de AWS Key Management Service.

Example

En el siguiente ejemplo se muestra cómo agregar instrucciones a la política de claves de KMS. Estas instrucciones permiten el acceso a la información sobre rendimiento. Dependiendo de cómo utilice la clave KMS, es posible que desee cambiar algunas restricciones. Antes de agregar sentencias a la directiva, elimine todos los comentarios.

JSON

```
{
    "Version":"2012-10-17",
    "Id" : "your-policy",
    "Statement" : [
        {
            "Sid" : "AllowViewingRDSPerformanceInsights",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/Role1"
                ]
                },
             "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
                ],
            "Resource": "*",
            "Condition" : {
            "StringEquals" : {
                "kms:ViaService" : "rds.us-east-1.amazonaws.com"
                },
            "ForAnyValue:StringEquals": {
                "kms:EncryptionContext:aws:pi:service": "rds",
                "kms:EncryptionContext:service": "pi",
```

```
                "kms:EncryptionContext:aws:rds:db-id": "db-
AAAAABBBBBCCCCDDDDDEEEEE"
                }
            }
        }
    ]
}
```

Cómo Información de rendimiento utiliza la clave administrada por el cliente de AWS KMS

Información de rendimiento utiliza claves administradas por el cliente para cifrar datos confidenciales. Al activar Información de rendimiento, puede proporcionar una clave de AWS KMS a través de la API. Información de rendimiento crea permisos de AWS KMS en esta clave. Utiliza la clave y realiza las operaciones necesarias para procesar los datos confidenciales. Los datos confidenciales incluyen campos como el usuario, la base de datos, la aplicación y el texto de la consulta SQL. Información de rendimiento garantiza que los datos permanezcan cifrados tanto en reposo como en tránsito.

Funcionamiento de la IAM de Información de rendimiento con AWS KMS

IAM otorga permisos a API específicas. Información de rendimiento tiene las siguientes API públicas, que puede restringir mediante políticas de IAM:

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

Puede utilizar las siguientes solicitudes de API para obtener datos confidenciales.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetrics`

Cuando utiliza la API para obtener datos confidenciales, Información de rendimiento utiliza las credenciales del intermediario. Esta comprobación garantiza que el acceso a los datos confidenciales esté limitado a quienes tengan acceso a la clave de KMS.

Al llamar a estas API, necesita permisos para llamar a la API a través de la política de IAM y permisos para invocar la acción `kms:decrypt` a través de la política de claves de AWS KMS.

La API `GetResourceMetrics` puede devolver datos confidenciales y no confidenciales. Los parámetros de la solicitud determinan si la respuesta debe incluir datos confidenciales. La API devuelve datos confidenciales cuando la solicitud incluye una dimensión confidencial en los parámetros del filtro o de grupo.

Para obtener más información acerca de las dimensiones que puede utilizar con la API `GetResourceMetrics`, consulte DimensionGroup.

Example Ejemplos

En el siguiente ejemplo se solicitan los datos confidenciales del grupo:`db.user`

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
        "Limit": 2
      }
    }
  ],
  "StartTime": 1693872000,
```

```
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

En el siguiente ejemplo se solicitan los datos no confidenciales de la métrica:`db.load.avg`

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

Concesión de acceso preciso para Información sobre rendimiento

El control de acceso preciso ofrece formas adicionales de controlar el acceso a Información sobre rendimiento. Este control de acceso puede permitir o denegar el acceso a dimensiones individuales para acciones de Información sobre rendimiento `GetResourceMetrics`, `DescribeDimensionKeys` y `GetDimensionKeyDetails`. Para utilizar el acceso preciso, especifique las dimensiones en la política de IAM mediante claves de condición. La evaluación del acceso sigue la lógica de evaluación de la política de IAM. Para obtener más información,

consulte Lógica de evaluación de políticas en la Guía del usuario de IAM. Si la instrucción de la política de IAM no especifica ninguna dimensión, entonces la instrucción controla el acceso a todas las dimensiones de la acción especificada. Para ver la lista de dimensiones disponibles, consulte DimensionGroup.

Para averiguar las dimensiones a las que están autorizadas a acceder sus credenciales, utilice el parámetro `AuthorizedActions` en `ListAvailableResourceDimensions` y especifique la acción. Los valores permitidos para `AuthorizedActions` son los siguientes:

- `GetResourceMetrics`
- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`

Por ejemplo, si especifica `GetResourceMetrics` para el parámetro `AuthorizedActions`, `ListAvailableResourceDimensions` devuelve la lista de dimensiones a las que la acción `GetResourceMetrics` está autorizada a acceder. Si especifica varias acciones en el parámetro `AuthorizedActions`, `ListAvailableResourceDimensions` devuelve una intersección de las dimensiones a las que esas acciones están autorizadas a acceder.

Example

El siguiente ejemplo proporciona acceso a las dimensiones especificadas para las acciones `GetResourceMetrics` y `DescribeDimensionKeys`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
```

```
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

A continuación se muestra la respuesta para la dimensión solicitada:

```
 // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
```

```
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

El siguiente ejemplo especifica un acceso permitido y dos denegados para las dimensiones.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },

          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
```

```
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }
    ]
```

```
}
```

A continuación se muestran las respuestas para las dimensiones solicitadas:

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // allowed for DescribeDimensionKeys because our IAM Policy
                  // denies it only for GetResourceMetrics
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

Uso del control de acceso basado en etiquetas para información de rendimiento

Puede controlar el acceso a métricas de información de rendimiento mediante etiquetas heredadas de la instancia de base de datos principal. Para controlar el acceso a las operaciones de información

de rendimiento, utilice políticas de IAM. Estas políticas pueden comprobar las etiquetas de la instancia de base de datos para determinar los permisos.

Funcionamiento de las etiquetas con información de rendimiento

La información de rendimiento aplica automáticamente las etiquetas de la instancia de base de datos para autorizar las métricas de información de rendimiento. Al agregar etiquetas a la instancia de base de datos, puede utilizarlas inmediatamente para controlar el acceso a los datos de información de rendimiento.

- Para agregar o actualizar etiquetas para las métricas de información de rendimiento, modifique las etiquetas de la instancia de base de datos.
- Para ver las etiquetas de las métricas de información de rendimiento, llame a `ListTagsForResource` en el recurso de métricas de información de rendimiento. Devolverá las etiquetas de la instancia de base de datos asociada a la métrica.

> **Note**
>
> Las operaciones `TagResource` y `UntagResource` devuelven un error si intenta usarlas directamente en las métricas de información de rendimiento.

Creación de políticas de IAM basadas en etiquetas

Para controlar el acceso a las operaciones de información de rendimiento, utilice la clave de condición `aws:ResourceTag` en las políticas de IAM. Estas políticas comprueban las etiquetas de la instancia de base de datos.

Example

Esta política impide el acceso a las métricas de información de rendimiento para las bases de datos de producción. La política rechaza la operación `pi:GetResourceMetrics` en información de rendimiento para cualquier recurso de base de datos etiquetado con `env:prod`.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Deny",
```

```
            "Action": "pi:GetResourceMetrics",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/env": "prod"
                }
            }
        }
    ]
}
```

Análisis de métricas mediante el panel de Información sobre rendimiento

Important

AWS ha anunciado la fecha de fin de la vida útil de información de rendimiento: 30 de junio de 2026. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles (de 1 a 24 meses) y los precios asociados. La API de Información de rendimiento seguirá existiendo sin cambios en los precios. Los costos de la API de Información de rendimiento aparecerán en la factura de AWS junto con el costo de información de la base de datos de CloudWatch.

Le recomendamos que actualice cualquier clúster de base de datos con el nivel de pago de información de rendimiento al modo avanzado de la información de la base de datos antes del 30 de junio de 2026. Para obtener información sobre la actualización al modo avanzado de Información de rendimiento, consulte Activación del modo avanzado de Información sobre las bases de datos para Amazon Aurora.

Si no realiza ninguna acción, los clústeres de base de datos que utilizan Información de rendimiento pasarán por defecto a utilizar el modo estándar de Información de rendimiento. Con el modo estándar de Información de base de datos, es posible que pierda el acceso al historial de datos de rendimiento de más de 7 días y que no pueda utilizar los planes de ejecución y las características de análisis bajo demanda en la consola de Amazon RDS. Después del 30 de junio de 2026, solo el modo avanzado de la información de base de datos admitirá los planes de ejecución y el análisis bajo demanda.

Con la información de la base de datos de CloudWatch, puede supervisar la carga de base de datos de la flota de bases de datos y analizar y solucionar problemas de rendimiento a escala. Para obtener más detalles acerca de Información de base de datos, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de

datos de CloudWatch. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

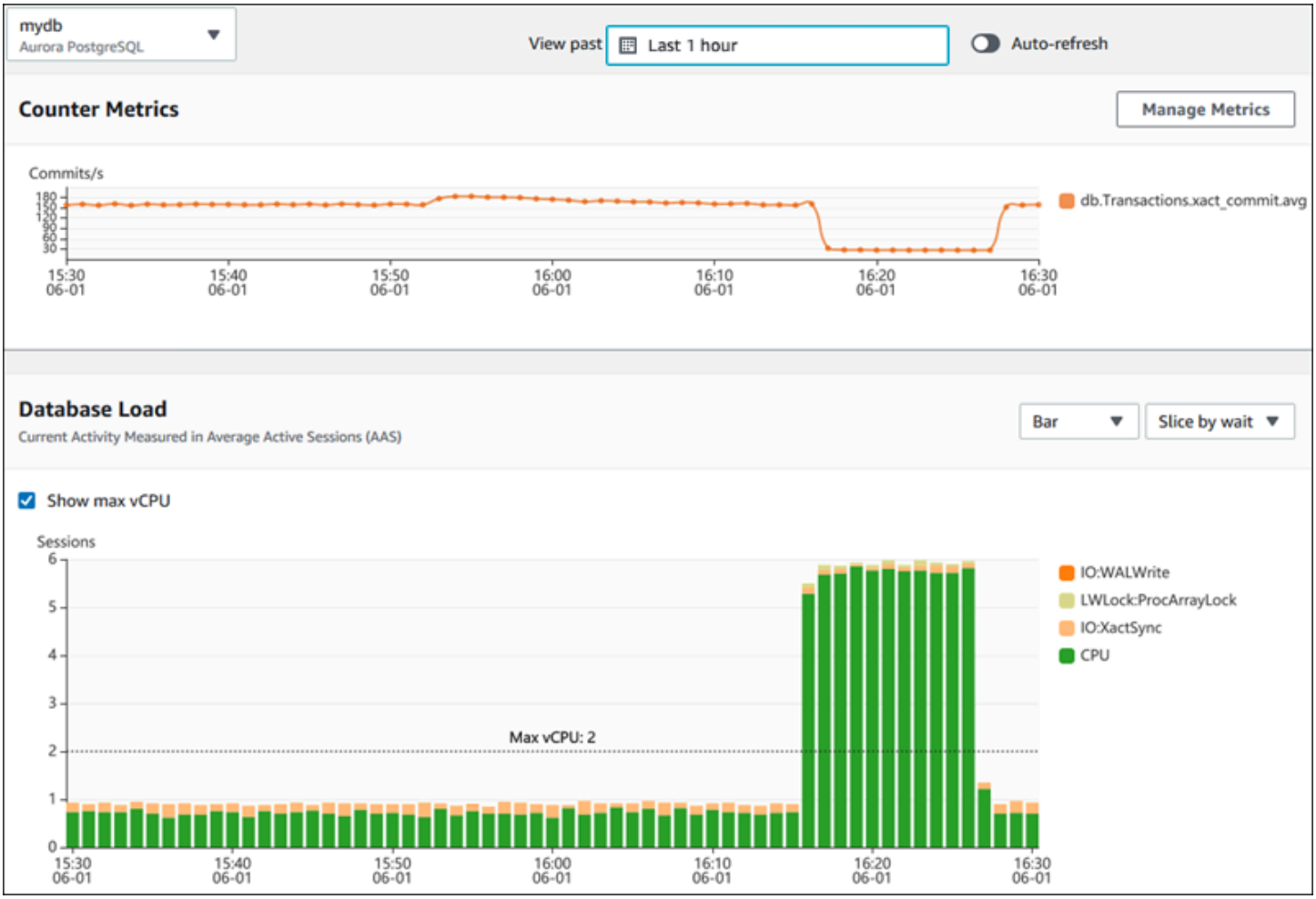
El panel de Performance Insights contiene información de desempeño de la base de datos para ayudarle a analizar y solucionar los problemas de desempeño. En la página del panel principal, encontrará información sobre la carga de la base de datos. Puede “dividir” la carga de la base de datos por dimensiones, como eventos de espera o SQL.

Panel de Performance Insights

- Información general del panel de Performance Insights
- Acceso al panel Información de rendimiento
- Análisis de carga de base de datos mediante eventos de espera
- Análisis del rendimiento de la base de datos durante un período de tiempo
- Análisis de consultas con la pestaña Top SQL en Información de rendimiento

Información general del panel de Performance Insights

El panel es la forma más sencilla de interactuar con Performance Insights. En el siguiente ejemplo, se muestra el panel de una instancia de base de datos de PostgreSQL.



Temas

- Filtro de intervalo de tiempo
- Gráfico Counter metrics (Métricas de contador)
- Gráfico Database load (Carga de base de datos)
- Tabla de dimensiones principales

Filtro de intervalo de tiempo

De forma predeterminada, el panel de Performance Insights muestra los datos de carga de la base de datos de la última hora. Puede ajustar este rango de modo que sea tan corto como 5 minutos o tan largo como 2 años. También puede seleccionar un rango relativo personalizado.

Puede seleccionar un rango absoluto con fecha y hora de inicio y fin. En el siguiente ejemplo, se muestra el intervalo de tiempo que comienza a medianoche del 25/09/24 y termina a las 23:59 del 28/09/24.

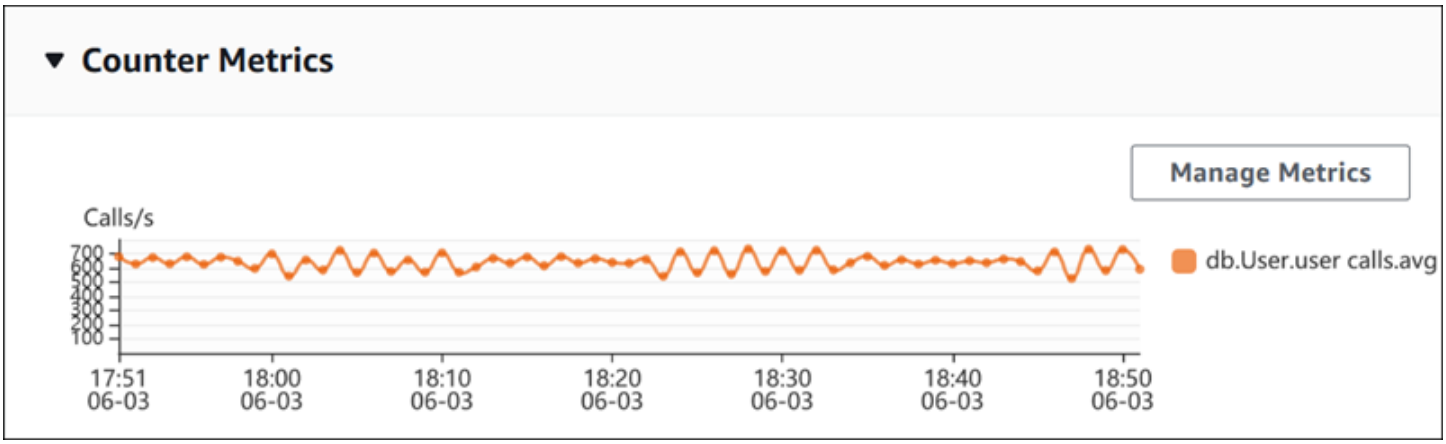
De forma predeterminada, la zona horaria para el panel de Información de rendimiento es el horario universal coordinado (UTC). También puede elegir la zona horaria local.

Gráfico Counter metrics (Métricas de contador)

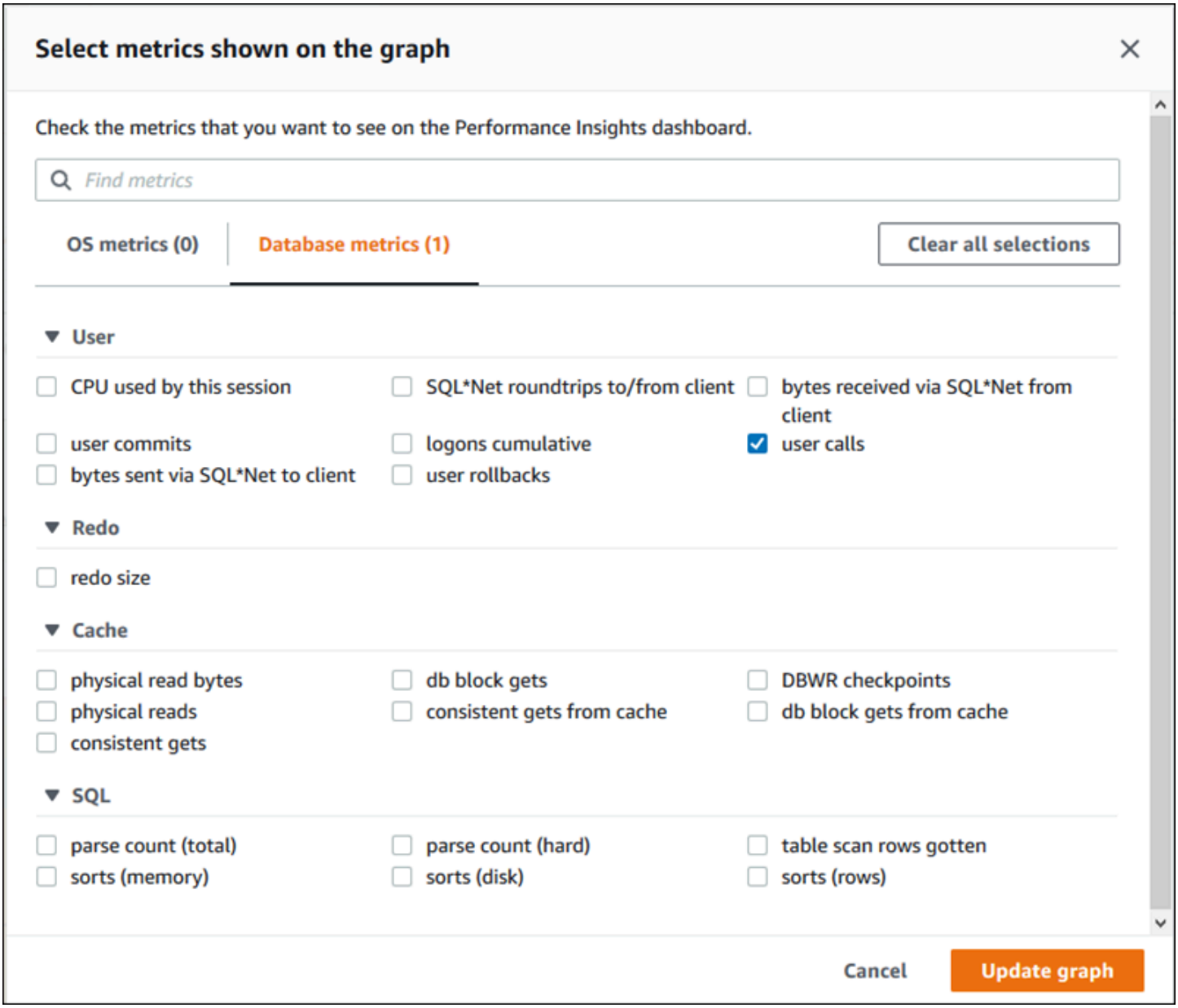
Con las métricas de contador, puede personalizar el panel de Información sobre rendimiento para que incluya hasta 10 gráficos adicionales. Estos gráficos muestran una selección de docenas de métricas de rendimiento de sistemas operativos y bases de datos. Esta información se puede correlacionar con la carga de base de datos para ayudar a identificar y analizar problemas de rendimiento.

El gráfico Counter metrics (Métricas de contador) muestra los datos para los contadores de rendimiento. Las métricas predeterminadas dependen del motor de base de datos:

- Aurora MySQL:– `db.SQL.Innodb_rows_read.avg`
- Aurora PostgreSQL – `db.Transactions.xact_commit.avg`



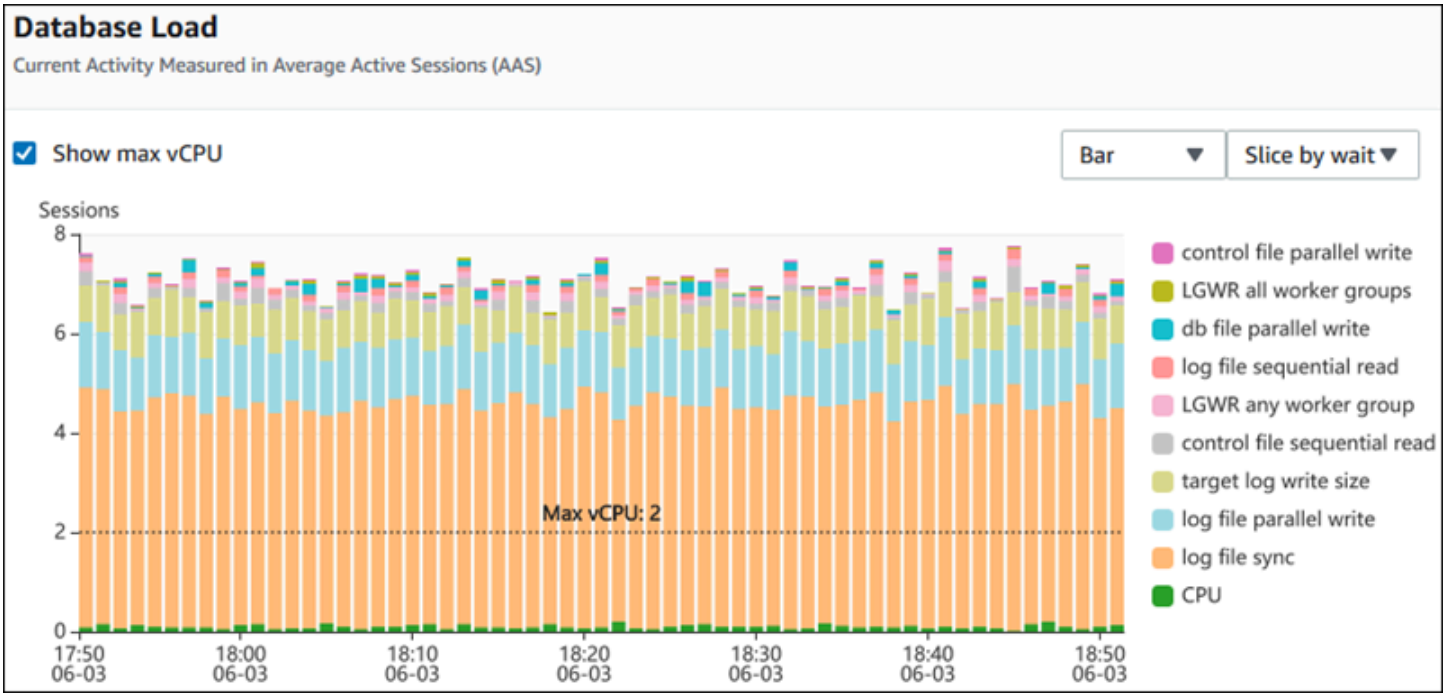
Para cambiar los contadores de rendimiento, elija Manage Metrics (Administrar métricas). Puede seleccionar varias métricas del sistema operativo o métricas de la base de datos, como se muestra en la siguiente captura de pantalla. Para ver los detalles de cualquier métrica, sitúe el cursor sobre el nombre de la métrica.



Para obtener descripciones de las métricas de contador que puede agregar para cada motor de base de datos, consulte Métricas de contador de Información de rendimiento.

Gráfico Database load (Carga de base de datos)

El gráfico Database load (Carga de base de datos) muestra cómo se compara la carga de base de datos con la capacidad de la instancia de base de datos representada por la línea Max vCPU (Máximo de vCPU). De forma predeterminada, el gráfico de líneas apilado representa la carga de base de datos como promedio de sesiones activas por unidad de tiempo. La carga de base de datos está dividida (agrupada) por estados de espera.

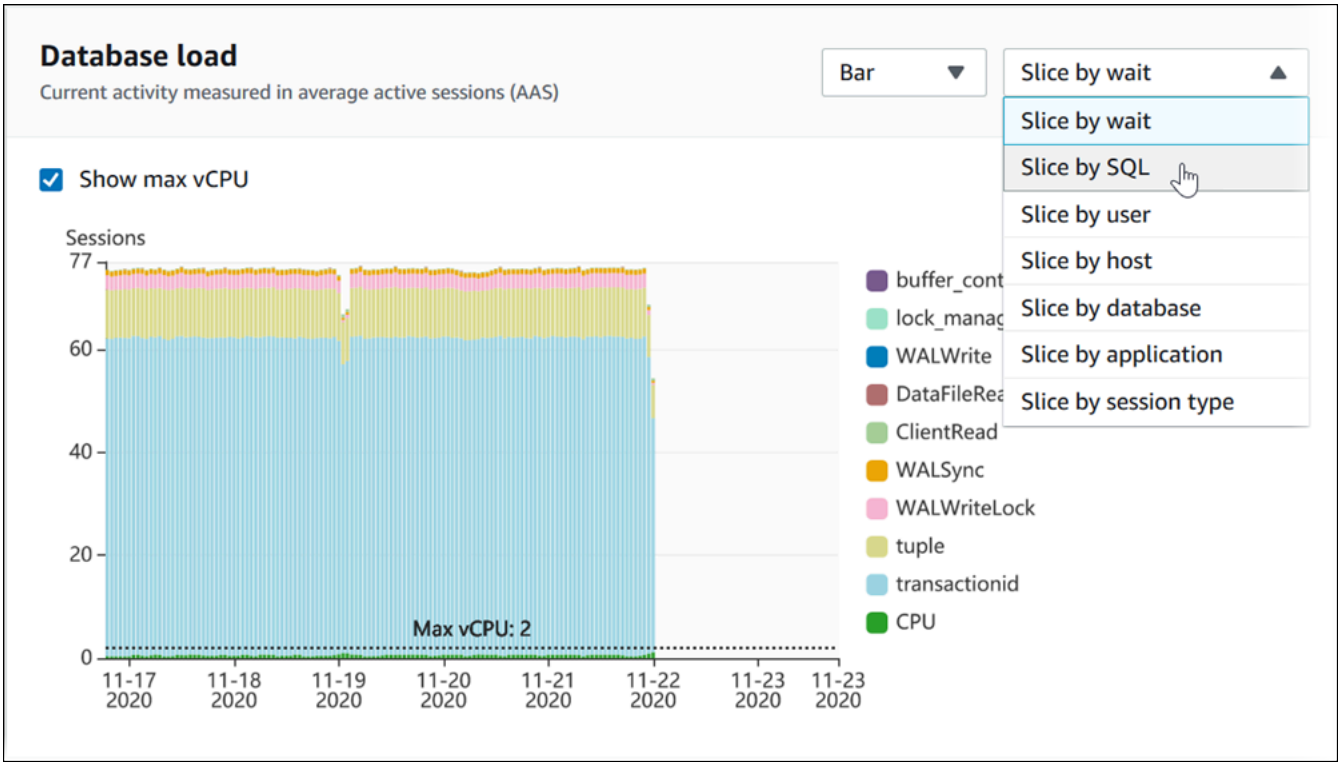


Carga de base de datos dividida por dimensiones

Puede elegir ver la carga como sesiones activas agrupadas por cualquier dimensión admitida. En la tabla siguiente se muestran las dimensiones admitidas para los distintos motores.

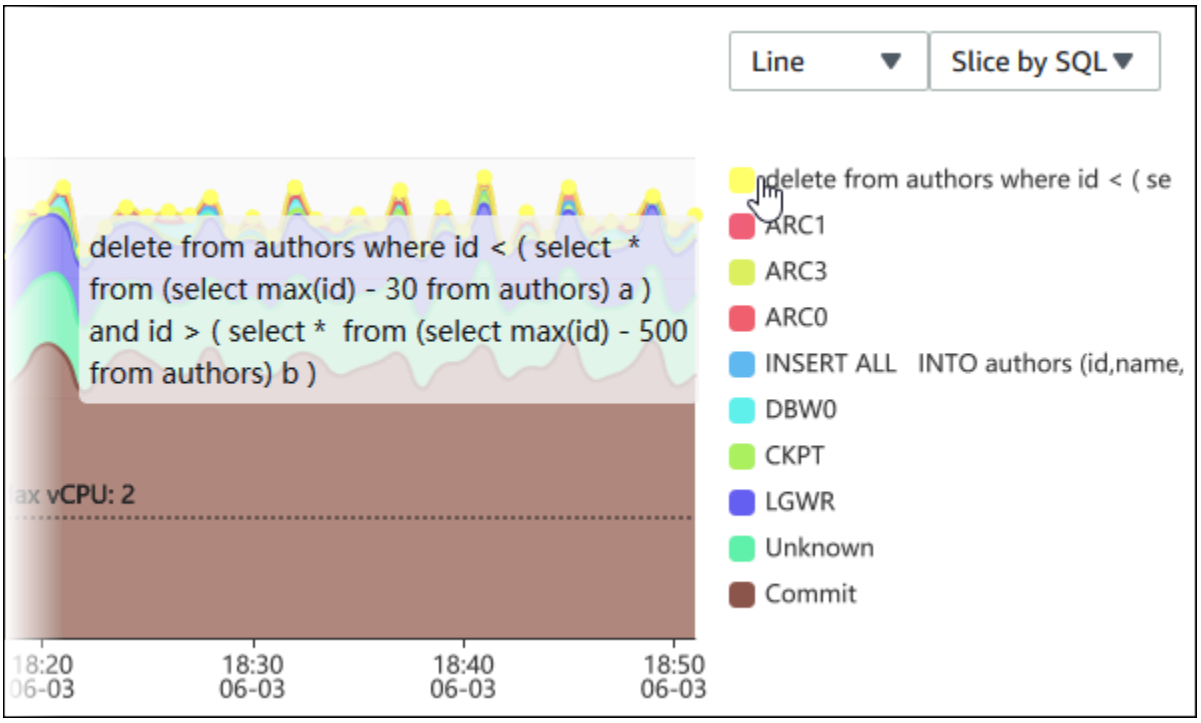
Dimensión	Aurora PostgreSQL	Aurora MySQL
Host	Sí	Sí
SQL	Sí	Sí
User	Sí	Sí
Esperas	Sí	Sí
Aplicación	Sí	No
Base de datos	Sí	Sí
Tipo de sesión	Sí	No

En la imagen siguiente, se muestran las dimensiones de una instancia de base de datos de PostgreSQL.



Detalles de carga de base de datos de un elemento de dimensión

Para consultar los detalles de un elemento de carga de base de datos dentro de una dimensión, pase el cursor sobre el nombre de elemento. En la imagen siguiente, se muestran los detalles de una instrucción de SQL.



Para consultar los detalles de cualquier elemento para el periodo de tiempo seleccionado en la leyenda, coloque el cursor sobre ese elemento.

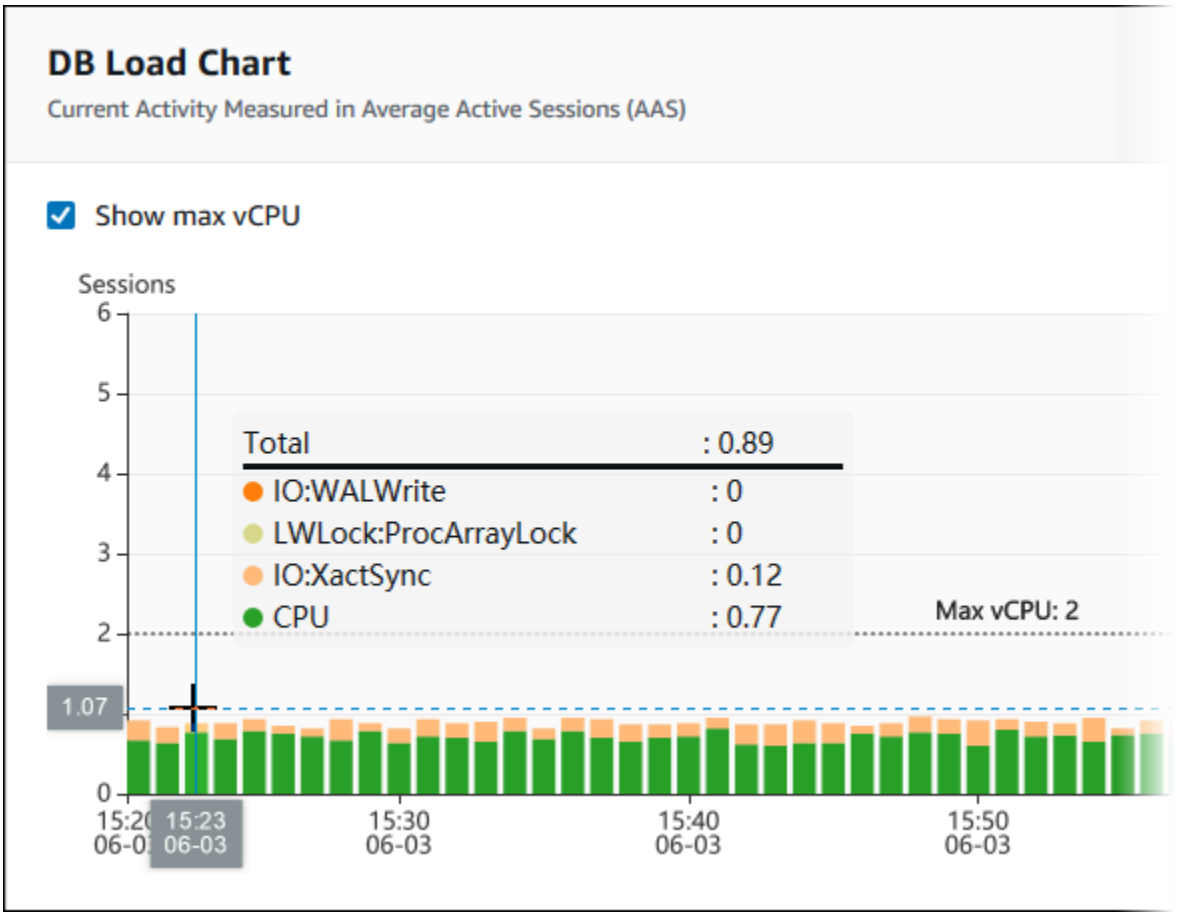
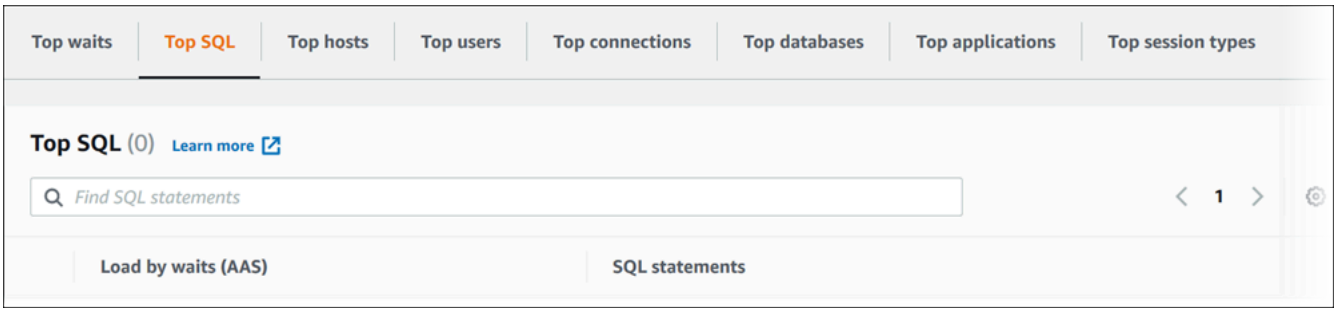


Tabla de dimensiones principales

La tabla de dimensiones principales divide la carga de base de datos por diferentes dimensiones. Una dimensión es una categoría o “dividir por” para diferentes características de la carga de base de datos. Si la dimensión es SQL, Top SQL (SQL principal) muestra las instrucciones SQL que más contribuyen a la carga de bases de datos.



Elija cualquiera de las siguientes pestañas de dimensión.

Tab	Descripción	Motores admitidos
SQL principal	Las instrucciones SQL que se están ejecutando	Todos
Esperas principales	El evento por el que la base de datos de backend está esperando.	Todos
Hosts principales	El nombre del host del cliente conectado.	Todos
Usuarios principales	El usuario que ha iniciado sesión en la base de datos.	Todos
Aplicaciones principales	El nombre de la aplicación que está conectada a la base de datos.	Aurora PostgreSQL solamente
Tipos de sesiones principales	El tipo de la sesión actual	Aurora PostgreSQL únicamente

Para obtener más información sobre cómo analizar las consultas mediante la pestaña Top SQL (SQL principal), consulte Información general sobre la pestaña Top SQL (SQL principal).

Acceso al panel Información de rendimiento

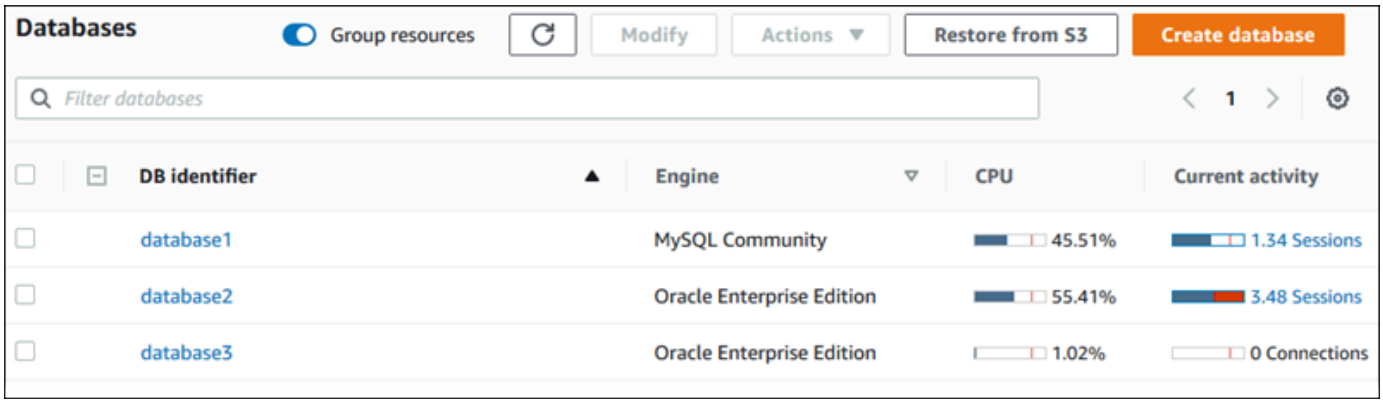
Amazon RDS ofrece una vista consolidada de las métricas de Información de rendimiento y CloudWatch en el panel Información de rendimiento.

Para acceder al panel Información de rendimiento, lleve a cabo el siguiente procedimiento.

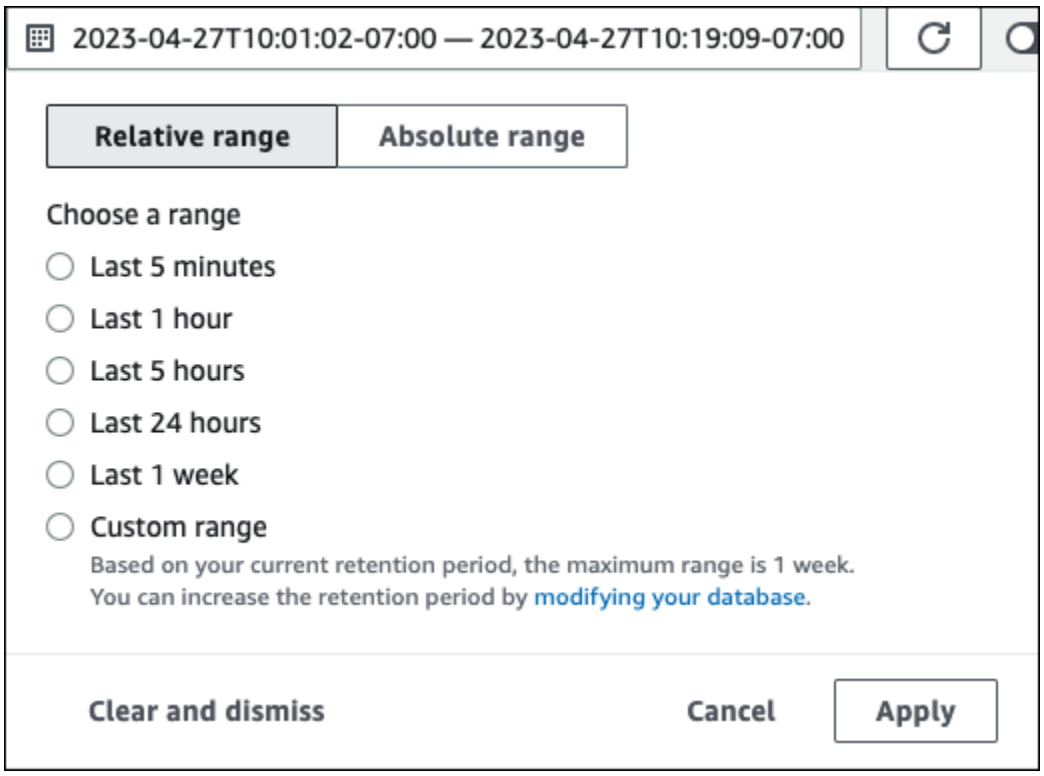
Para ver el panel de Performance Insights en la consola de administración de AWS

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.

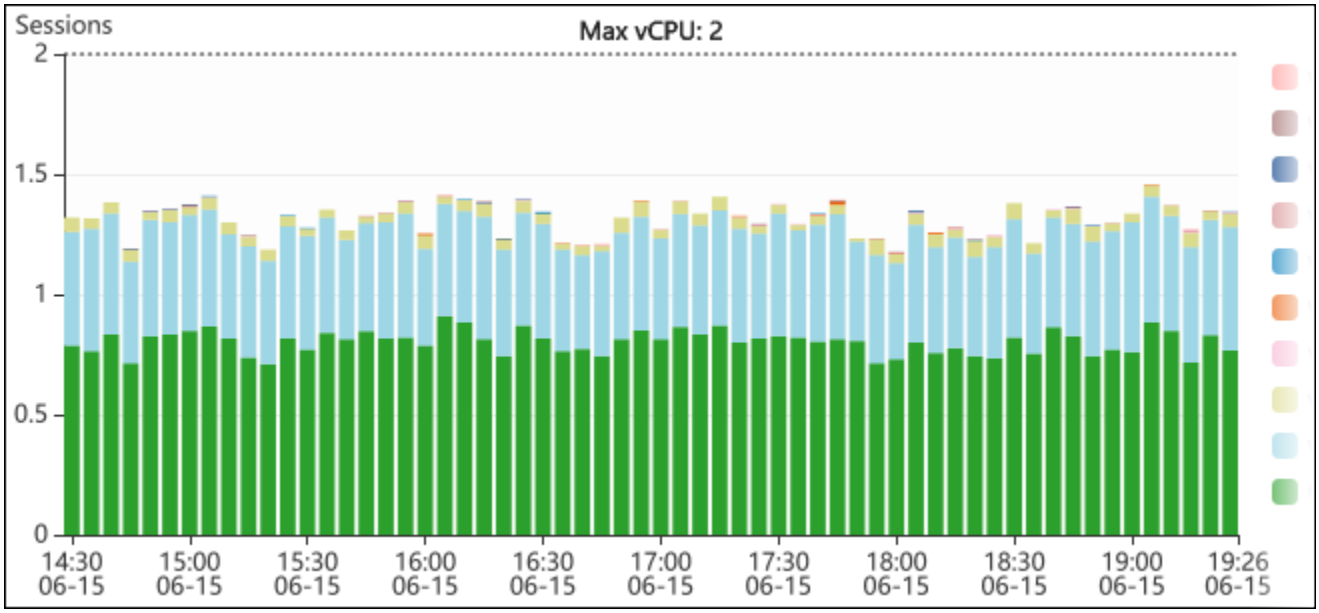
En instancias de base de datos con Información de rendimiento activado, también puede acceder al panel Información de rendimiento eligiendo el elemento Sesiones en la lista de instancias de bases de datos. En Actividad actual, el elemento Sesiones muestra la carga de la base de datos en el como promedio de sesiones activas en los últimos cinco minutos. La barra muestra gráficamente la carga. Cuando la barra está vacía, la instancia de base de datos está inactiva. Conforme aumenta la carga, la barra se va completando en azul. Cuando la carga supera el número de CPU virtuales (vCPU) en la clase de instancia de base de datos, la barra cambia a rojo, lo cual indica un posible cuello de botella.



4. (Opcional) Elija el intervalo de fecha o tiempo en la parte superior derecha y especifique un intervalo de tiempo relativo o absoluto diferente. Ahora puede especificar un período de tiempo y generar un informe de análisis del rendimiento de la base de datos. El informe proporciona información identificada y recomendaciones. Para obtener más información, consulte Creación de un informe de análisis de rendimiento en Información de rendimiento.

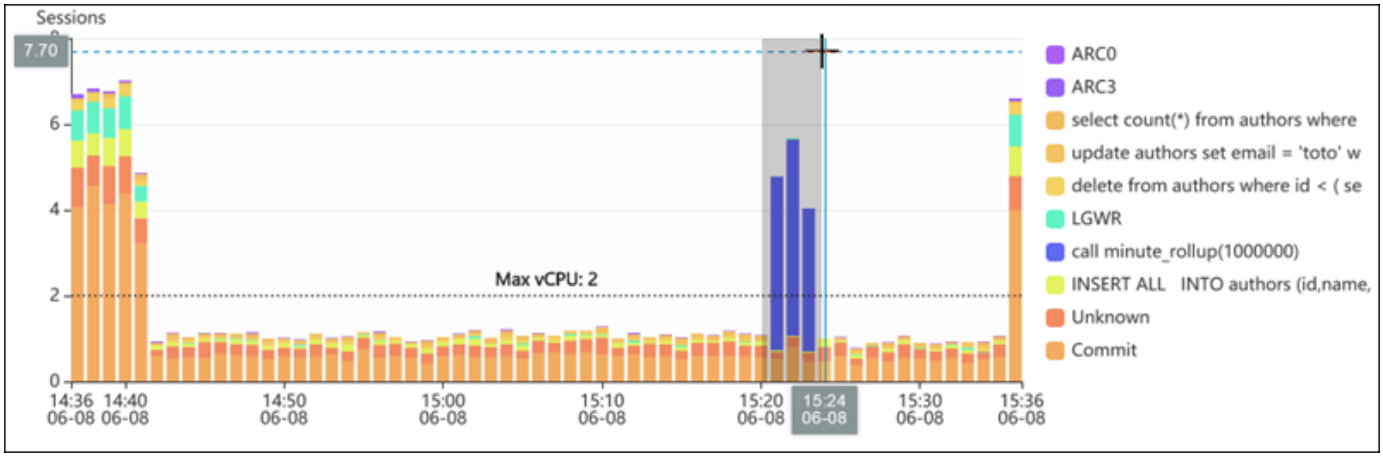


En la siguiente captura de pantalla, el intervalo de carga de la base de datos es de 5 horas.

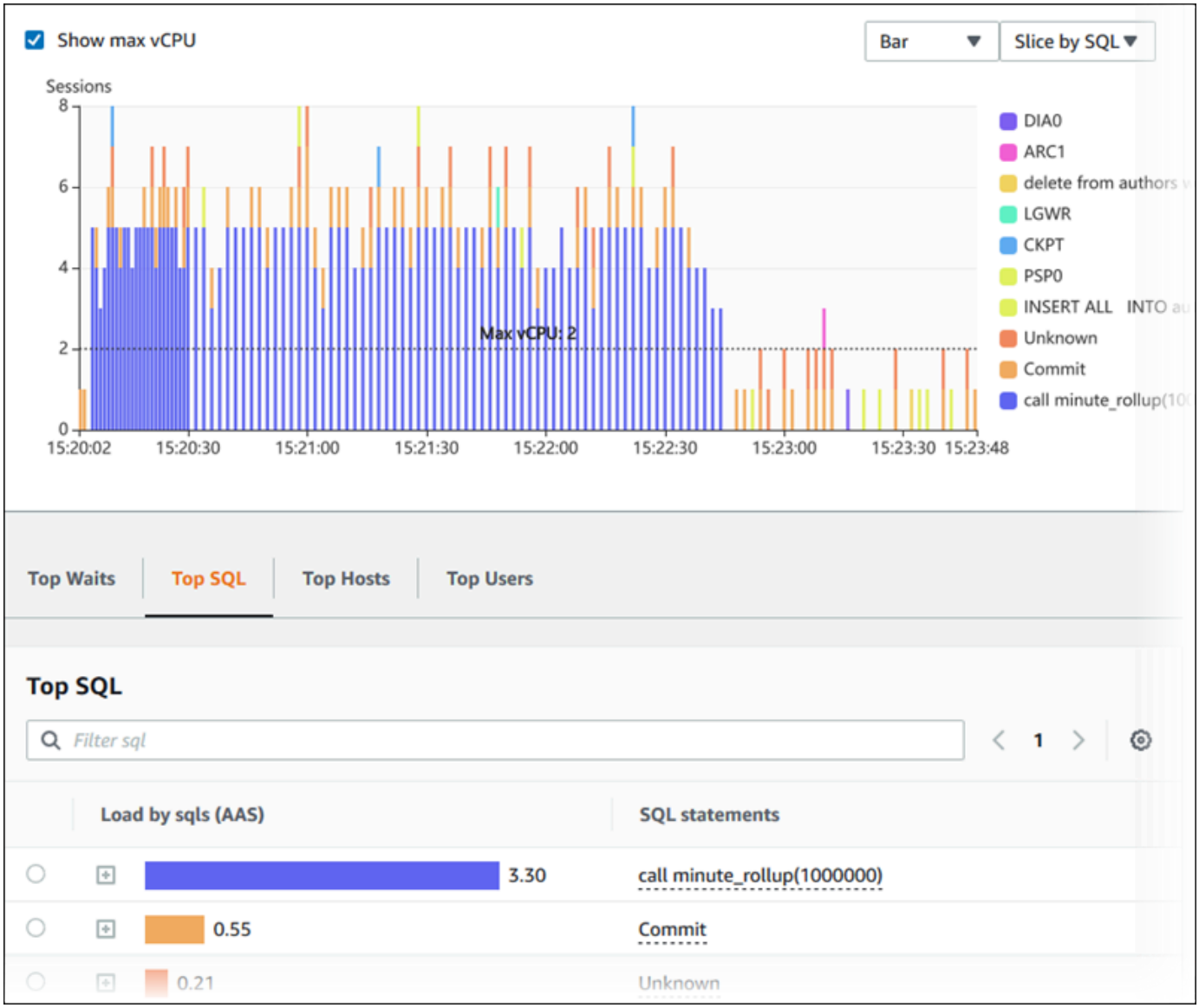


5. (Opcional) Para ampliar una parte del gráfico de carga de base de datos, elija la hora de inicio y arrástrela hasta el final del período que desee.

   El área seleccionada se resalta en el gráfico de carga de base de datos.



Cuando suelte el ratón, el gráfico de carga de base de datos ampliará la región de AWS seleccionada y se volverá a calcular la tabla de las dimensiones principales.



6. (Opcional) Para actualizar los datos automáticamente, habilite Actualización automática.

El panel de Información de rendimiento se actualiza automáticamente con nuevos datos. La frecuencia de actualización depende de la cantidad de datos mostrados:

- 5 minutos actualiza cada 10 segundos.
- 1 hora actualiza cada 5 minutos.
- 5 horas actualiza cada 5 minutos.
- 24 horas actualiza cada 30 minutos.
- 1 semana actualiza cada día.

- 1 mes actualiza cada día.

Análisis de carga de base de datos mediante eventos de espera

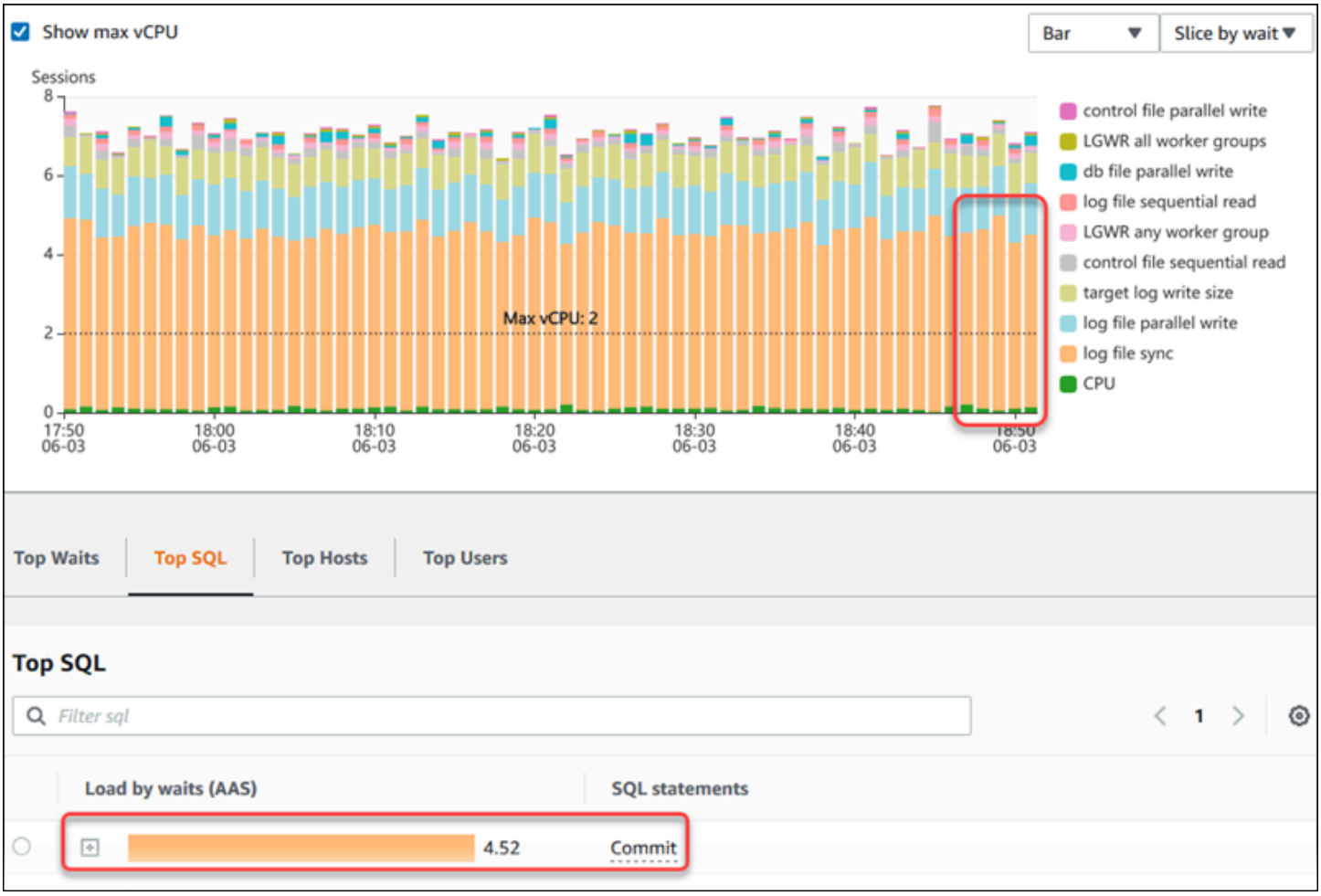
Si el gráfico Database load (Carga de base de datos) indica que hay un cuello de botella, puede averiguar de dónde procede la carga. Para ello, fíjese en la tabla de elementos de carga principales situada debajo del gráfico Database load (Carga de base de datos). Elija un elemento en particular, como una consulta SQL o un usuario, para ampliar la información de ese elemento y ver los detalles.

La carga de base de datos agrupada por esperas y principales consultas de SQL es la vista predeterminada del panel de Performance Insights. Esta combinación normalmente ofrece la máxima información sobre problemas de desempeño. La carga de la base de datos agrupada por esperas indica si hay algún cuello de botella de simultaneidad o recursos en la base de datos. En este caso, la pestaña SQL de la tabla de elementos de carga principales indica qué consultas están contribuyendo a esa carga.

Este es el flujo de trabajo típico para diagnosticar los problemas de desempeño:

1. Revise el gráfico Carga de base de datos para ver si hay algún incidente de carga de base de datos que sobrepase la línea Máximo de CPU.
2. De ser así, fíjese en el gráfico Database load (Carga de base de datos) e identifique qué estado o estados de espera son los principales responsables.
3. Para identificar las consultas de resumen que están provocando la carga, consulte qué consultas de la pestaña SQL de la tabla de elementos de carga principales están contribuyendo más a esos estados de espera. Para identificarlas, utilice la columna DB Load by Wait (Carga de base de datos por espera).
4. Elija una de estas consultas de resumen en la pestaña SQL para ampliarla y ver las consultas secundarias que contiene.

Por ejemplo, en el panel que se muestra a continuación, la espera de la sincronización de archivos de registro se corresponde con la mayor parte de la carga de base de datos. La espera de todos los nodos de trabajo de LGWR también es alta. El gráfico Top SQL (SQL principal) muestra lo que provoca las esperas de sincronización de archivos de registro: instrucciones `COMMIT` frecuentes. En este caso, confirmar con menos frecuencia reducirá la carga de la base de datos.



Análisis del rendimiento de la base de datos durante un período de tiempo

Analice el rendimiento de la base de datos con análisis bajo demanda mediante la creación de un informe de análisis de rendimiento durante un periodo de tiempo. Vea informes de análisis de rendimiento para detectar problemas de rendimiento, como cuellos de botella de recursos o cambios en una consulta en la instancia de base de datos. El panel de Información de rendimiento le permite seleccionar un período de tiempo específico y crear un informe de análisis de rendimiento. También puede añadir una o varias etiquetas al informe.

Para utilizar esta característica, debe utilizar el período de retención del nivel de pago. Para obtener más información, consulte Precios y retención de datos de Performance Insights

Puede seleccionar y ver el informe en la pestaña Informes de análisis de rendimiento: nuevo. El informe contiene la información, las métricas relacionadas y las recomendaciones para resolver el problema de rendimiento. Puede ver el informe durante el período de retención de Información de rendimiento.

El informe se elimina si la hora de inicio del período de análisis del informe está fuera del período de retención. También puede eliminar el informe antes de que finalice el período de retención.

Para detectar los problemas de rendimiento y generar el informe de análisis para su instancia de base de datos, debe activar Información de rendimiento. Para obtener más información acerca de la activación de Información de rendimiento, consulte Activación y desactivación de Información de rendimiento de Aurora.

Para obtener información sobre la compatibilidad de esta característica por región, motor de base de datos y clase de instancia, consulte Compatibilidad del motor de la base de datos, la región y la clase de instancia de Amazon Aurora con características de Información de rendimiento.

En las siguientes secciones, puede crear, ver y eliminar informes de análisis de rendimiento, así como añadir etiquetas.

Temas

- Creación de un informe de análisis de rendimiento en Información de rendimiento
- Visualización de un informe de análisis de rendimiento en Información de rendimiento
- Cómo añadir etiquetas a un informe de análisis de rendimiento en Información de rendimiento
- Eliminación de un informe de análisis de rendimiento en Información de rendimiento

Creación de un informe de análisis de rendimiento en Información de rendimiento

Puede crear un informe de análisis de rendimiento para un período específico en el panel de Información de rendimiento. Puede seleccionar un período de tiempo y añadir una o más etiquetas al informe de análisis.

El período de análisis puede oscilar entre 5 minutos y 6 días. Debe haber al menos 24 horas de datos de rendimiento antes de la hora de inicio del análisis.

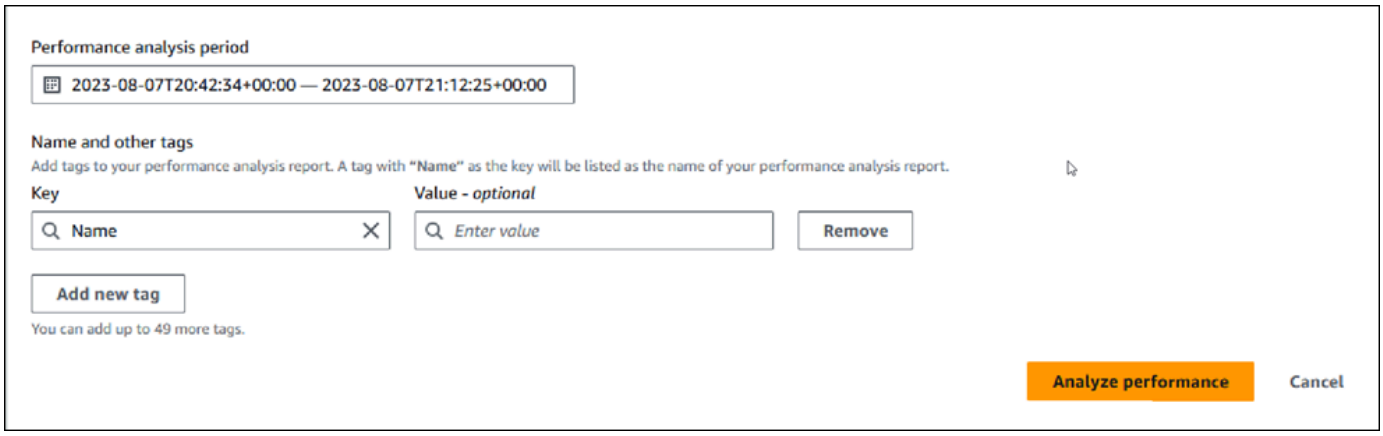
Para obtener información sobre la compatibilidad de esta característica por región, motor de base de datos y clase de instancia, consulte Compatibilidad del motor de la base de datos, la región y la clase de instancia de Amazon Aurora con características de Información de rendimiento.

Para crear un informe de análisis de rendimiento para un período de tiempo

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.

3. Elija una instancia de base de datos.
4. Elija Analizar rendimiento en la sección Carga de base de datos del panel de Información de rendimiento.

   Se muestran los campos para establecer el período de tiempo y añadir una o más etiquetas al informe de análisis de rendimiento.

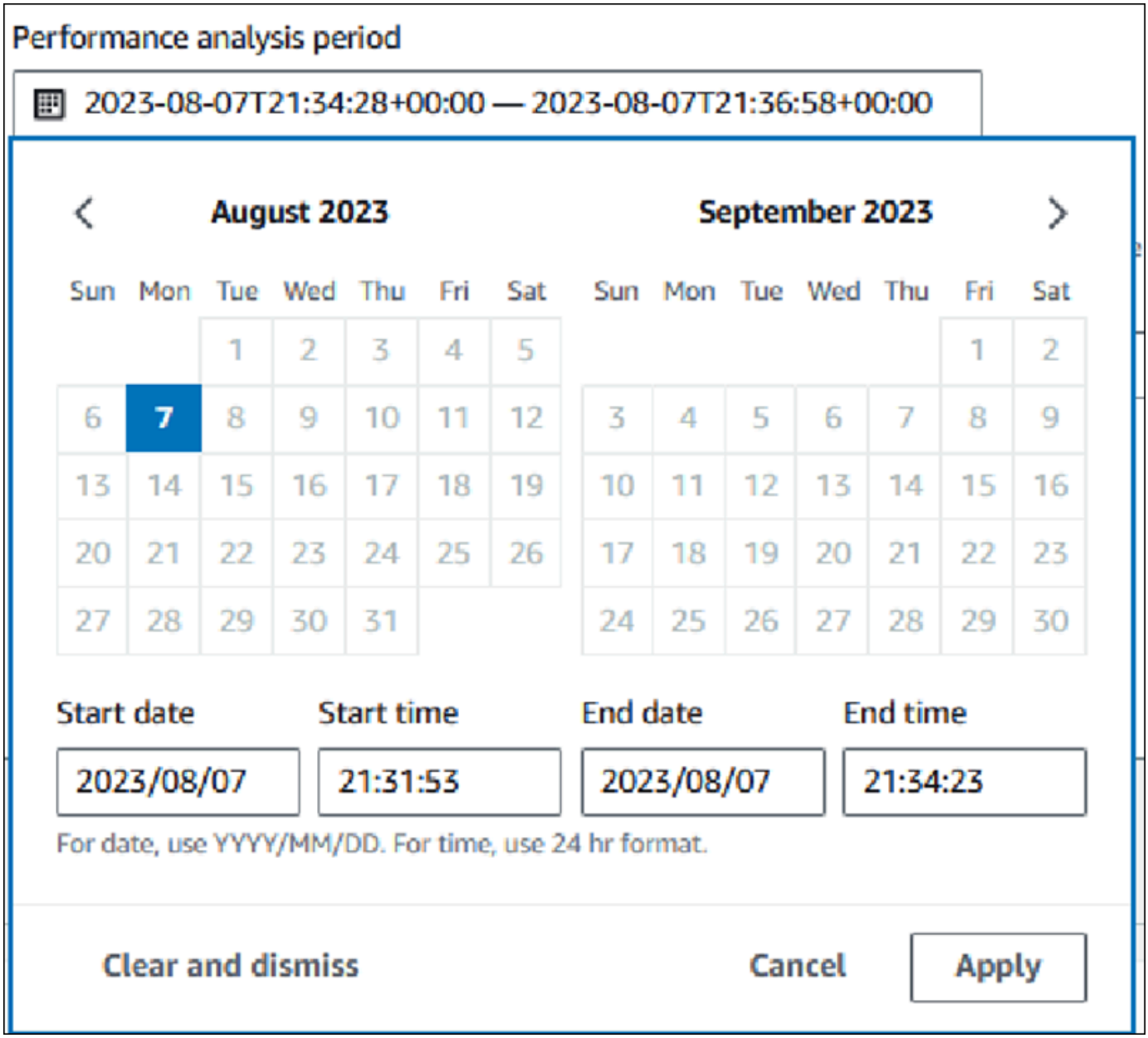


5. Elija un período de tiempo. Si establece un período de tiempo en el Intervalo relativo o Intervalo absoluto en la esquina superior derecha, solo puede introducir o seleccionar la fecha y la hora del informe de análisis dentro de este período de tiempo. Si selecciona un período de análisis fuera de este período de tiempo, aparece un mensaje de error.

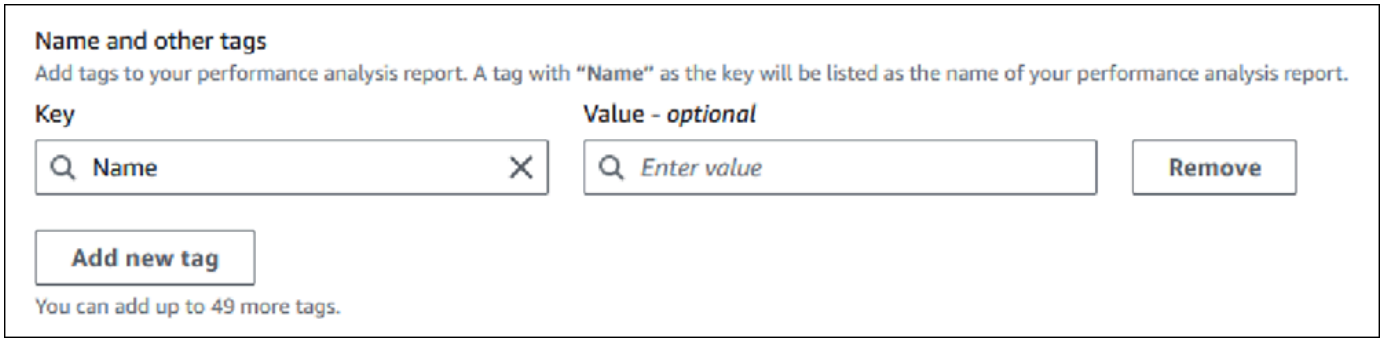
   Para establecer el período de tiempo, puede realizar una de las siguientes acciones:

   - Pulse y arrastre cualquiera de los controles deslizantes del gráfico de carga de la base de datos.

     El cuadro Período de análisis de rendimiento muestra el período de tiempo seleccionado y el gráfico de carga de la base de datos resalta el período de tiempo seleccionado.
   - Elija la Fecha de inicio, la Hora de inicio, la Fecha de finalización y la Hora de finalización en el cuadro Período de análisis de rendimiento.



6. (Opcional) Introduzca la Clave y el Valor-opcional para añadir una etiqueta al informe.



7. Elija Analizar rendimiento.

   Un banner muestra un mensaje que indica si el informe se ha generado correctamente o no ha tenido éxito. El mensaje también proporciona el enlace para ver el informe.

   En el siguiente ejemplo, se muestra el banner con el mensaje de creación correcta del informe.

   Puede ver el informe en la pestaña Informes de análisis de rendimiento: nuevo.

Puede crear un informe de análisis de rendimiento con la AWS CLI. Para ver un ejemplo sobre cómo crear un informe con la AWS CLI, consulte Creación de un informe de análisis de rendimiento para un período de tiempo.

Visualización de un informe de análisis de rendimiento en Información de rendimiento

La pestaña Informes de análisis de rendimiento: nuevo muestra todos los informes que se crean para la instancia de base de datos. En cada informe, se muestra lo siguiente:

- ID: identificador único del informe.
- Nombre: clave de etiqueta añadida al informe.
- Tiempo de creación del informe: hora en que creó el informe.
- Hora de inicio del análisis: hora de inicio del análisis en el informe.
- Hora de finalización del análisis: hora de finalización del análisis en el informe.

Para ver un informe de análisis de rendimiento

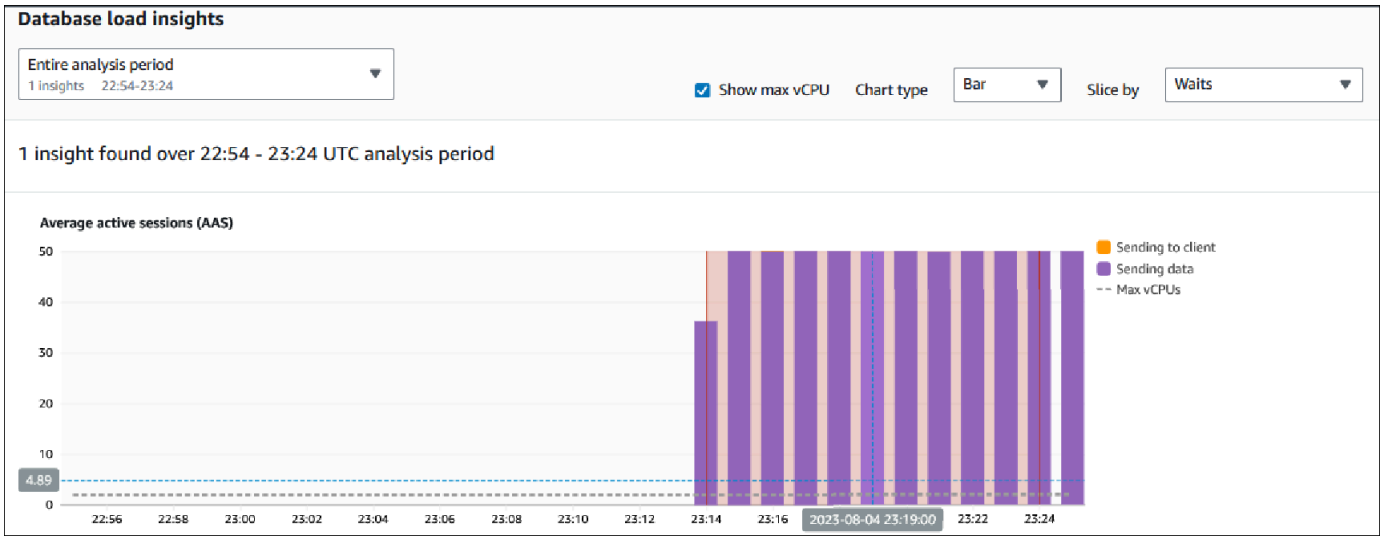
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija la instancia de base de datos para la que desee ver el informe de análisis.
4. Desplácese hacia abajo y elija la pestaña Informes de análisis de rendimiento: nuevo en el panel de Información de rendimiento.

   Se muestran todos los informes de análisis de los diferentes períodos de tiempo.
5. Elija el ID del informe que desea ver.

El gráfico de carga de la base de datos muestra todo el período de análisis de forma predeterminada si se identifica más de una información. Si el informe ha identificado una información, el gráfico de carga de la base de datos muestra la información de forma predeterminada.

El panel también muestra las etiquetas del informe en la sección Etiquetas.

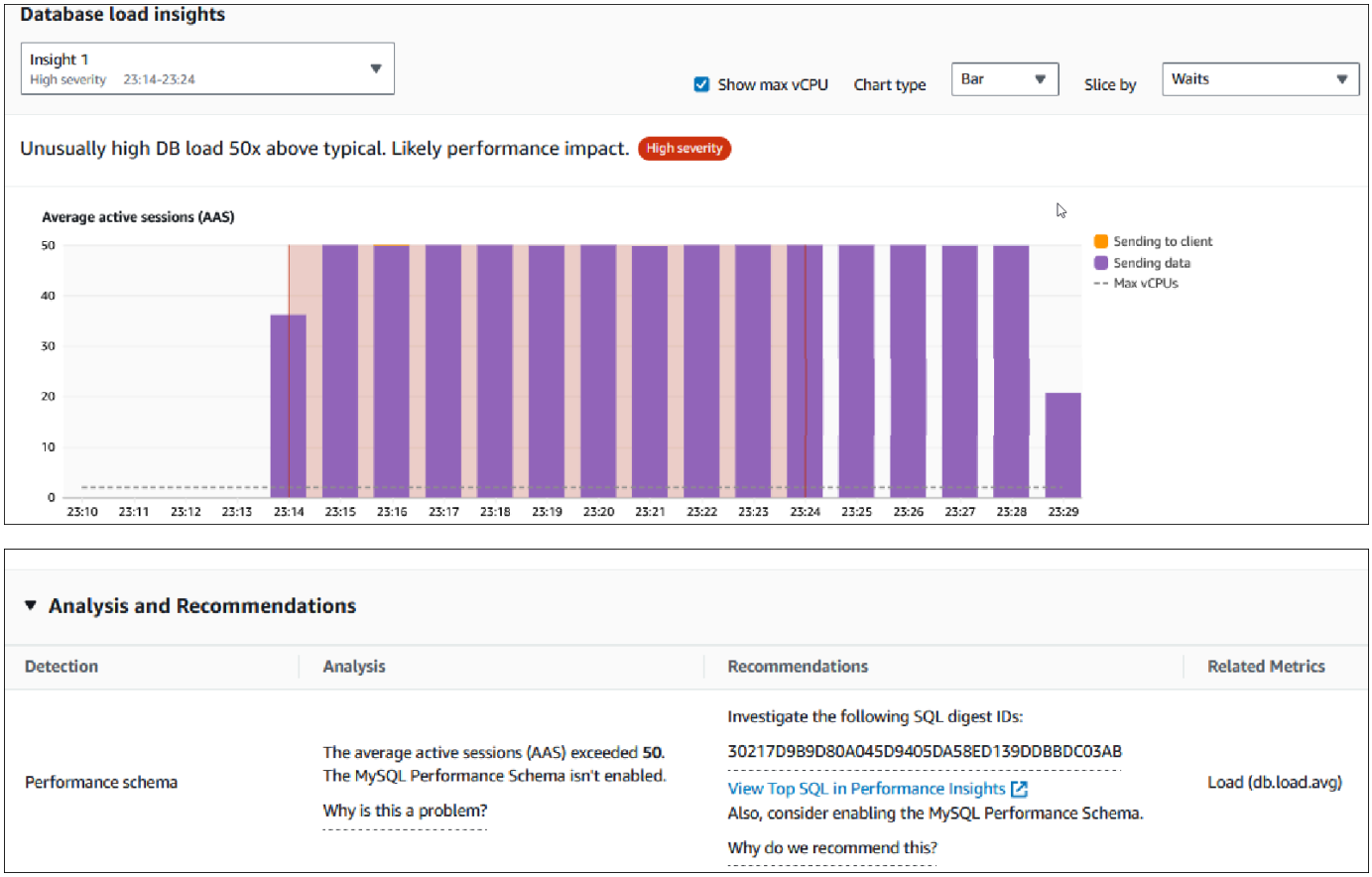
En el siguiente ejemplo se muestra todo el período de análisis del informe.



6. Elija la información en la lista Información de carga de la base de datos que desee ver si se identifica más de una información en el informe.

   El panel muestra el mensaje de información, el gráfico de carga de la base de datos que destaca el período de tiempo de la información, los análisis, las recomendaciones y la lista de etiquetas del informe.

   En el siguiente ejemplo se muestra la información de carga de la base de datos en el informe.



Cómo añadir etiquetas a un informe de análisis de rendimiento en Información de rendimiento

Puede añadir una etiqueta al crear o ver un informe. Puede añadir un máximo de 50 etiquetas a un informe.

Debe tener los permisos para añadir las etiquetas. Para obtener más información sobre las políticas de acceso para la Información de rendimiento, consulte Configuración de directivas de acceso para información sobre rendimiento

Para añadir una o más etiquetas al crear un informe, consulte el paso 6 del procedimiento Creación de un informe de análisis de rendimiento en Información de rendimiento.

Para añadir una o más etiquetas al ver un informe

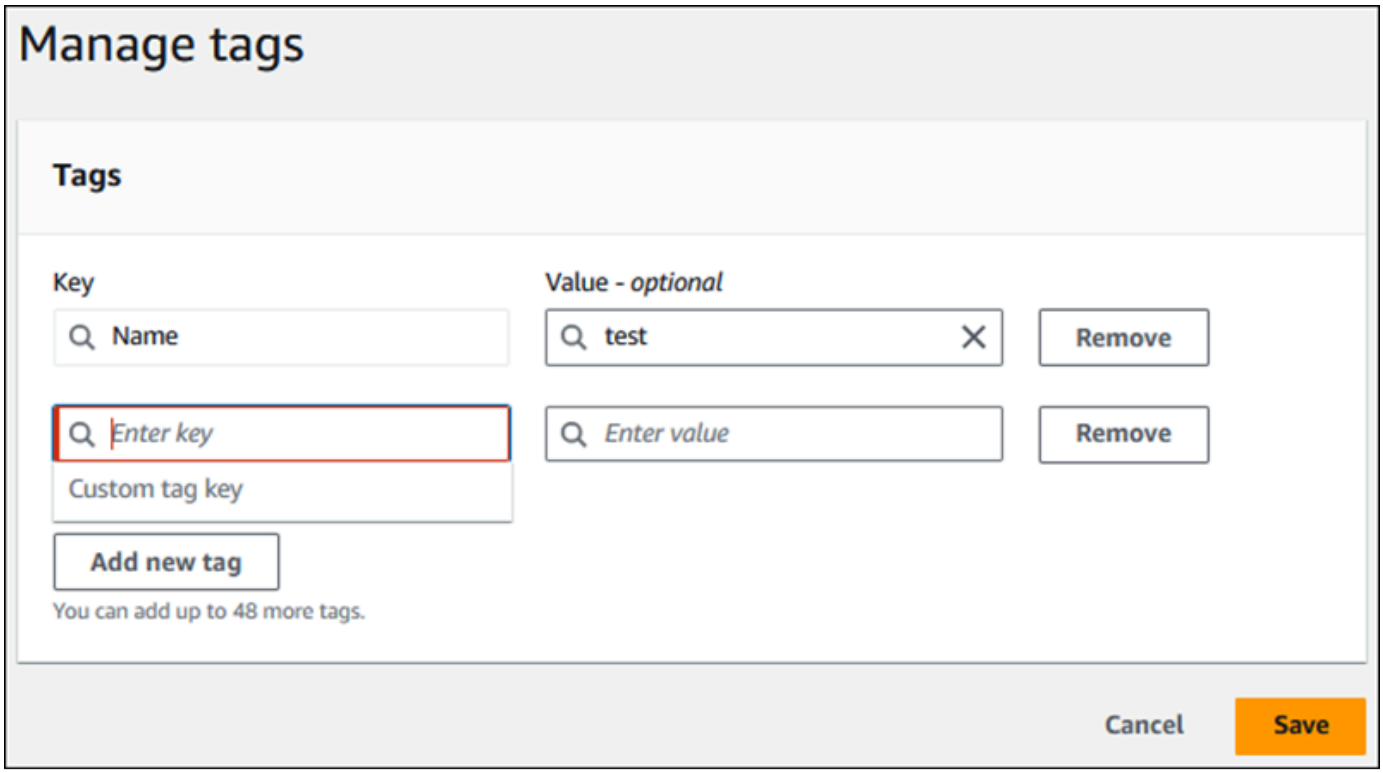
1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.

Se muestra el panel de Información de rendimiento para la instancia de base de datos.

4. Desplácese hacia abajo y elija la pestaña Informes de análisis de rendimiento: nuevo.
5. Elija el informe al que desee añadir las etiquetas.

   El panel muestra el informe.
6. Desplácese hacia abajo hasta Etiquetas y elija Administrar etiquetas.
7. Elija Añadir nueva etiqueta.
8. Introduzca la Clave y el Valor - opcional, y elija Agregar nueva etiqueta.

   En el ejemplo siguiente se ofrece la opción de añadir una etiqueta nueva en el informe seleccionado.



Se crea una etiqueta nueva para el informe.

La lista de etiquetas del informe se muestra en la sección Etiquetas del panel. Si desea eliminar una etiqueta del informe, elija Eliminar junto a la etiqueta.

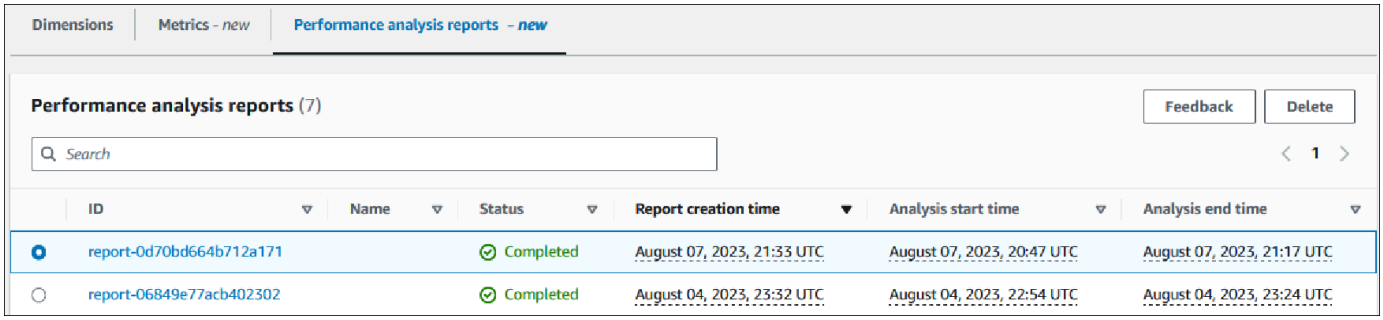
Eliminación de un informe de análisis de rendimiento en Información de rendimiento

Puede eliminar un informe de la lista de informes que se muestra en la pestaña Informes de análisis de rendimiento o mientras visualiza un informe.

Para eliminar un informe

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.
3. Elija una instancia de base de datos.

   Se muestra el panel de Información de rendimiento para la instancia de base de datos.
4. Desplácese hacia abajo y elija la pestaña Informes de análisis de rendimiento: nuevo.
5. Seleccione el informe que quiera eliminar y elija Eliminar en la esquina superior derecha.



   Aparece una ventana de confirmación. El informe se elimina después de que seleccione confirmar.
6. (Opcional) Elija el ID del informe que desea eliminar.

   En la página del informe, elija Eliminar en la esquina superior derecha.

   Aparece una ventana de confirmación. El informe se elimina después de que seleccione confirmar.

Análisis de consultas con la pestaña Top SQL en Información de rendimiento

En el panel de Performance Insights de Amazon RDS, puede encontrar información sobre las consultas recientes y en ejecución en la pestaña Top SQL (SQL principal) en la tabla Top dimensions (Dimensiones principales). Puede utilizar esta información para ajustar sus consultas.

Temas

- Información general sobre la pestaña Top SQL (SQL principal)
- Acceso a más texto SQL en el panel de Performance Insights
- Visualización de estadísticas de SQL en el panel de Performance Insights

Información general sobre la pestaña Top SQL (SQL principal)

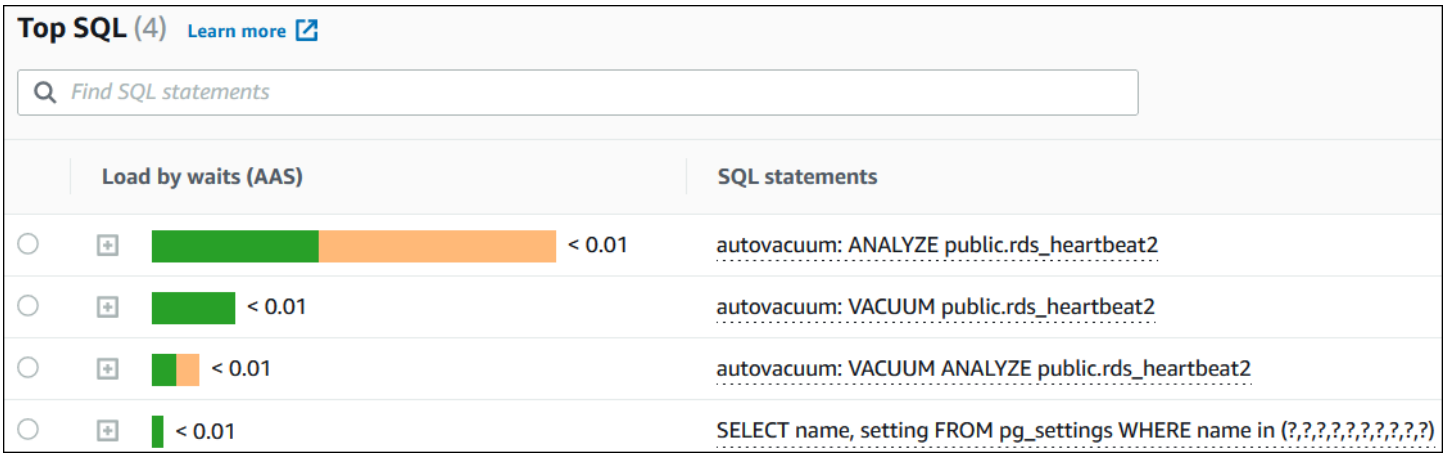
De forma predeterminada, la pestaña Top SQL (SQL principal) muestra las 25 consultas que contribuyen a la carga de base de datos. Para ayudar a ajustar las consultas, puede analizar información como el texto de la consulta y las estadísticas de SQL. También puede elegir las estadísticas que quiere que aparezcan en la pestaña Top SQL (SQL principal).

Temas

- Texto SQL
- Estadísticas de SQL
- Load by waits (AAS) (Carga por esperas [AAS])
- Ver información de SQL
- Escoger preferencias de estadísticas

Texto SQL

De forma predeterminada, cada fila de la tabla Top SQL (SQL principal) muestra 500 bytes de texto de ara cada instrucción.

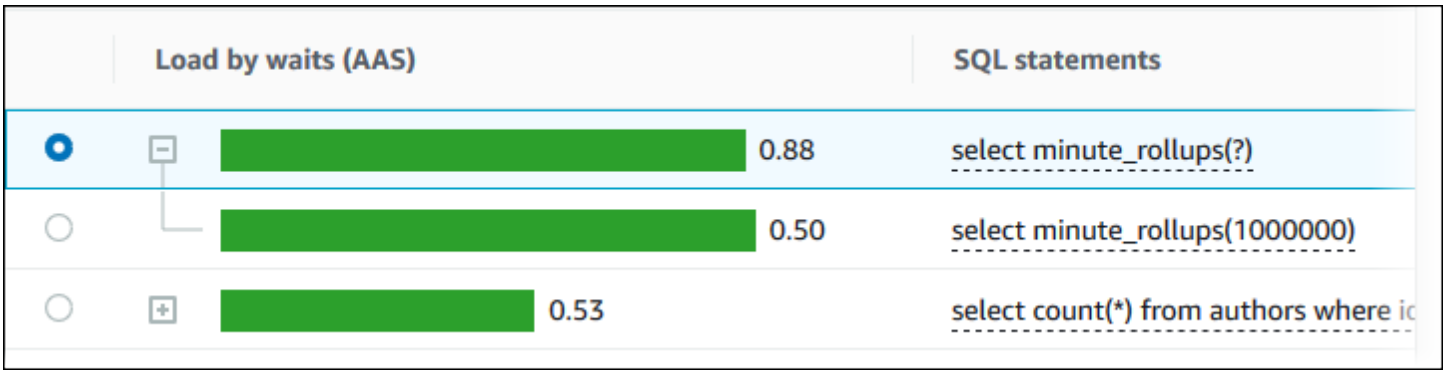


Para obtener información sobre cómo ver más de los 500 bytes predeterminados de texto SQL, consulte Acceso a más texto SQL en el panel de Performance Insights.

Un resumen de SQL es un compuesto de múltiples consultas reales que son similares en estructura, pero que pueden tener diferentes valores literales. El resumen reemplaza los valores codificados por un signo de interrogación. Por ejemplo, un resumen podría ser `SELECT * FROM emp WHERE lname= ?`. Este resumen podría incluir las siguientes consultas secundarias:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Para ver las instrucciones SQL literales de un resumen, seleccione la consulta y, a continuación, elija el símbolo más (+). En el siguiente ejemplo, la consulta seleccionada es un resumen.

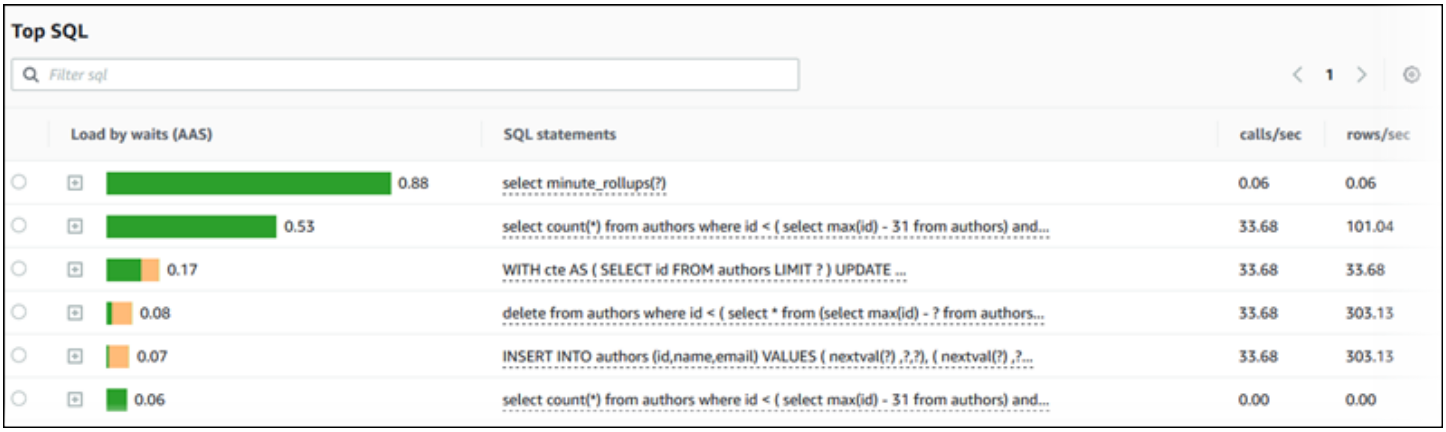


> **Note**
>
> Un resumen de SQL agrupa instrucciones SQL similares, pero no redacta información confidencial.

Estadísticas de SQL

Las estadísticas de SQL son métricas relacionadas con el rendimiento de las consultas SQL. Por ejemplo, Performance Insights podría mostrar ejecuciones por segundo o filas procesadas por segundo. Performance Insights recopila estadísticas solo para las consultas más comunes. Normalmente, coinciden con las consultas principales por carga mostradas en el panel de Performance Insights.

Todas las líneas de la tabla Top SQL (SQL principal) muestra estadísticas relevantes de la instrucción o resumen de SQL, como se muestra en el ejemplo siguiente.



Performance Insights puede informar `0.00` y `-` (desconocido) para las estadísticas de SQL. Esta situación se produce en las siguientes condiciones:

- Solo existe una muestra. Por ejemplo, Performance Insights calcula las tasas de cambio para las consultas de Aurora PostgreSQL basadas en varios ejemplos de la vista `pg_stat_statements`. Cuando una carga de trabajo se ejecuta durante un breve período de tiempo, es posible que Performance Insights solo recopile una muestra, lo que significa que no puede calcular una tasa de cambio. El valor desconocido se representa con un guion (`-`).
- Dos muestras tienen los mismos valores. Performance Insights no puede calcular una tasa de cambio porque no se ha producido ningún cambio, por lo que informa la tasa como `0.00`.
- Una instrucción de Aurora PostgreSQL carece de identificador válido. PostgreSQL crea un identificador para una instrucción solo después de analizar. Por lo tanto, puede existir una instrucción en las estructuras internas en memoria de PostgreSQL sin identificador. Dado que Performance Insights realiza muestras de estructuras internas en memoria una vez por segundo, pueden aparecer consultas de baja latencia para una sola muestra. Si el identificador de consulta no está disponible para esta muestra, Performance Insights no puede asociar esta instrucción a sus estadísticas. El valor desconocido se representa con un guion (`-`).

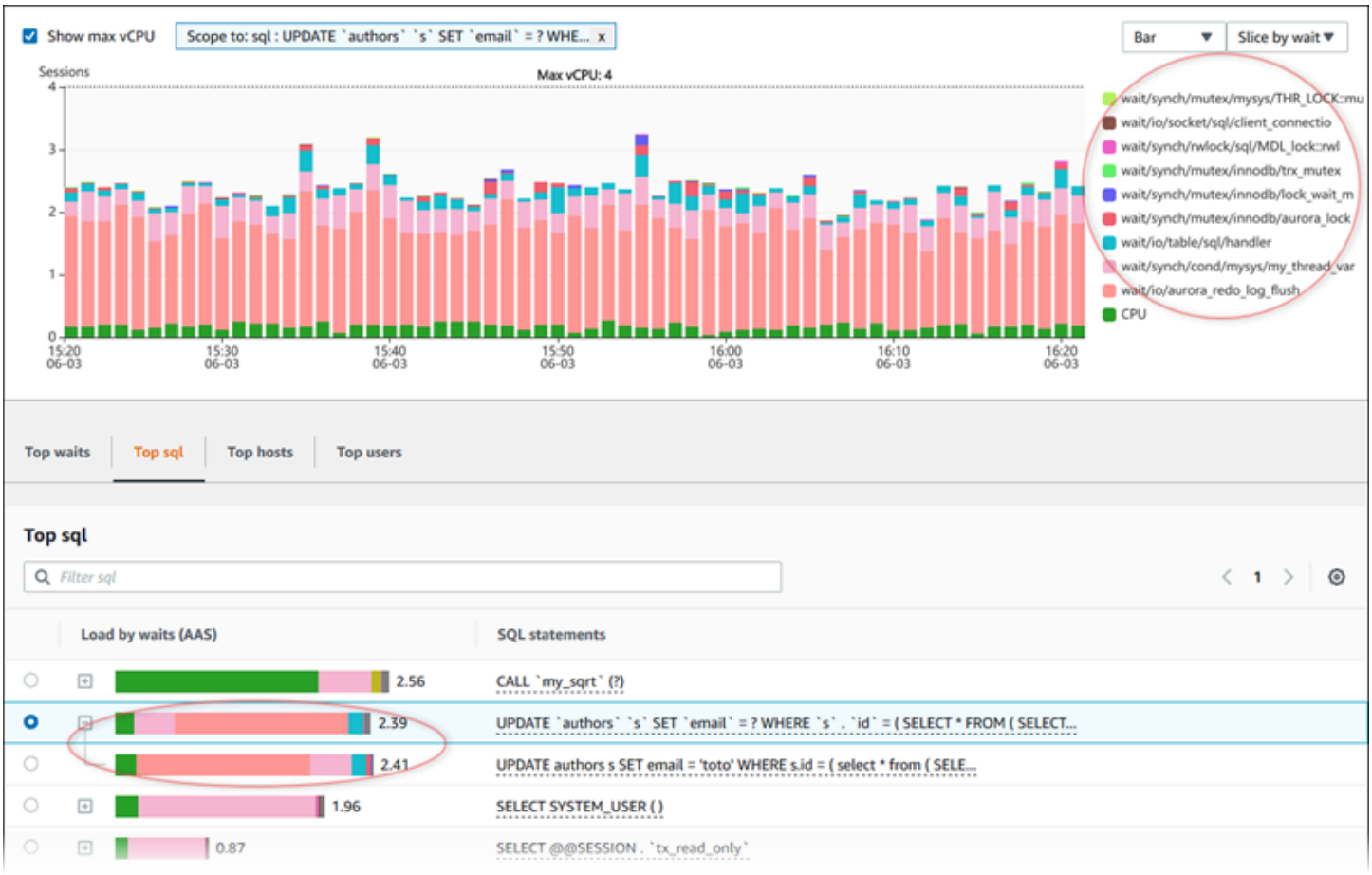
Para obtener una descripción de las estadísticas de SQL de los motores de Aurora, consulte Estadísticas de SQL para Performance Insights.

Load by waits (AAS) (Carga por esperas [AAS])

En Top SQL (SQL principal), la columna Load by waits (AAS) (Carga por espera [AAS]) ilustra el porcentaje de carga de la base de datos asociada con cada elemento de carga principal. Esta columna refleja la carga de ese elemento por cualquier agrupación que se haya seleccionado

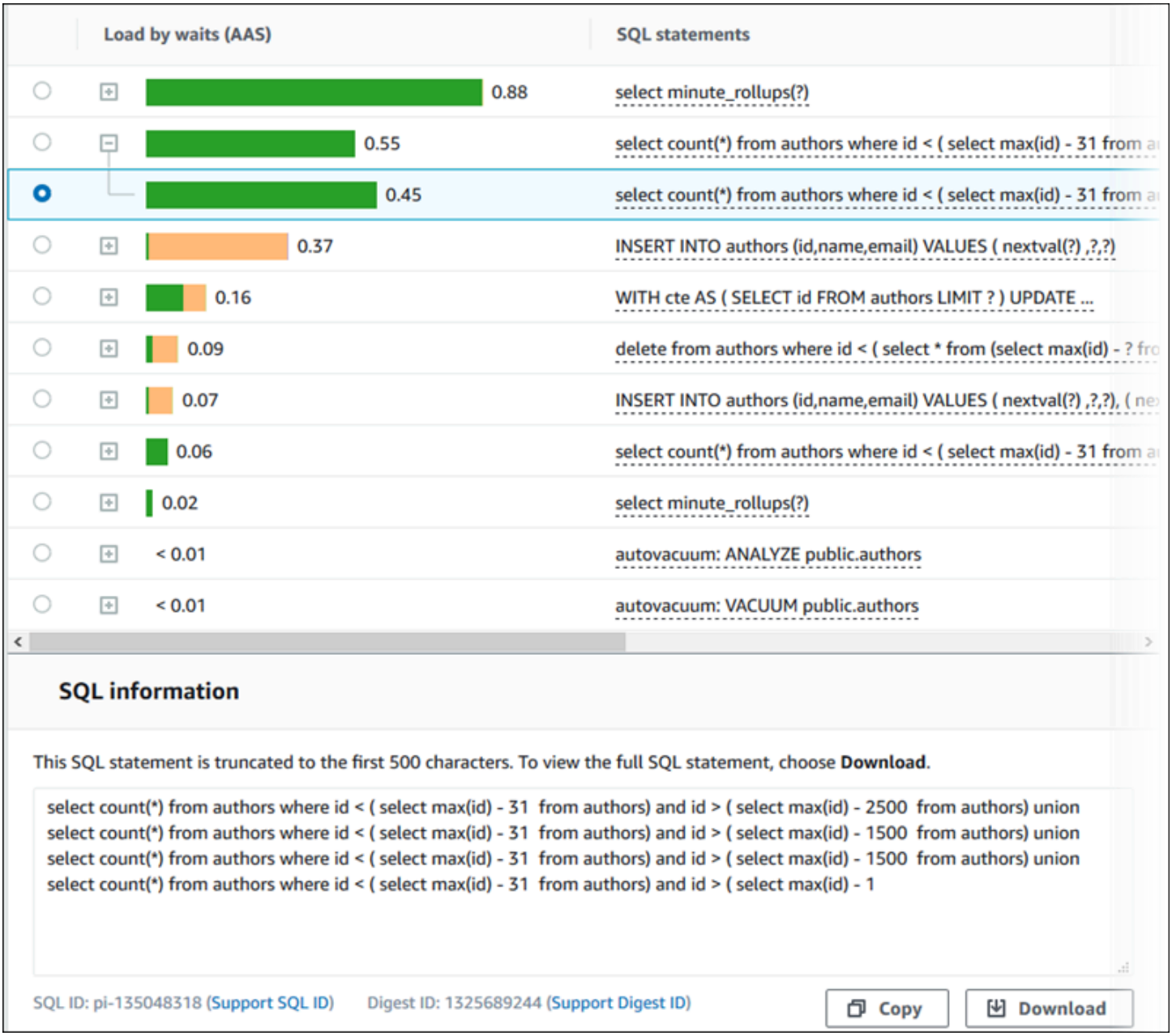
actualmente en el gráfico de carga de base de datos. Para obtener más información acerca de las sesiones activas de Average (AAS), consulte Sesiones activas promedio.

Por ejemplo, es posible que pueda agrupar el gráfico DB load (Carga de base de datos) por estados de espera. Puede examinar consultas SQL en la tabla de elementos de carga principal. En este caso, la barra DB Load by Waits (Carga de base de datos por esperas) estaría dimensionada, segmentada y dividida por colores para mostrar en qué proporción contribuye esa consulta a un estado de espera. También muestra qué estados de espera afectan a la consulta seleccionada.



Ver información de SQL

En la tabla Top SQL (SQL principal), puede abrir una instrucción para consultar su información. La información aparece en el panel inferior.

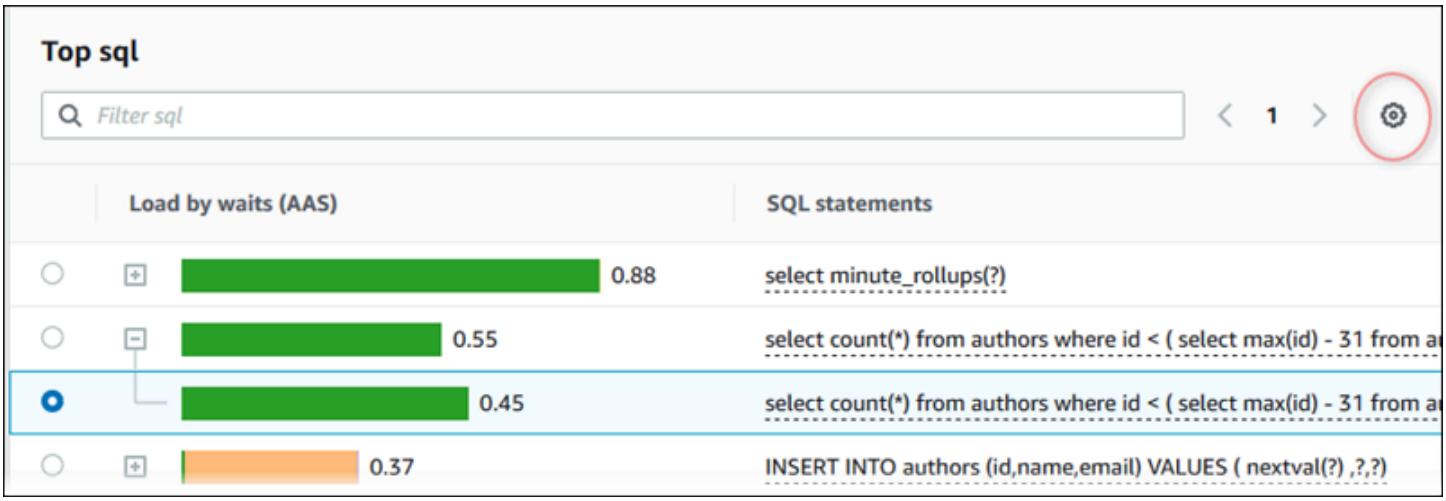


Los siguientes tipos de identificadores (ID) asociados con instrucciones SQL:

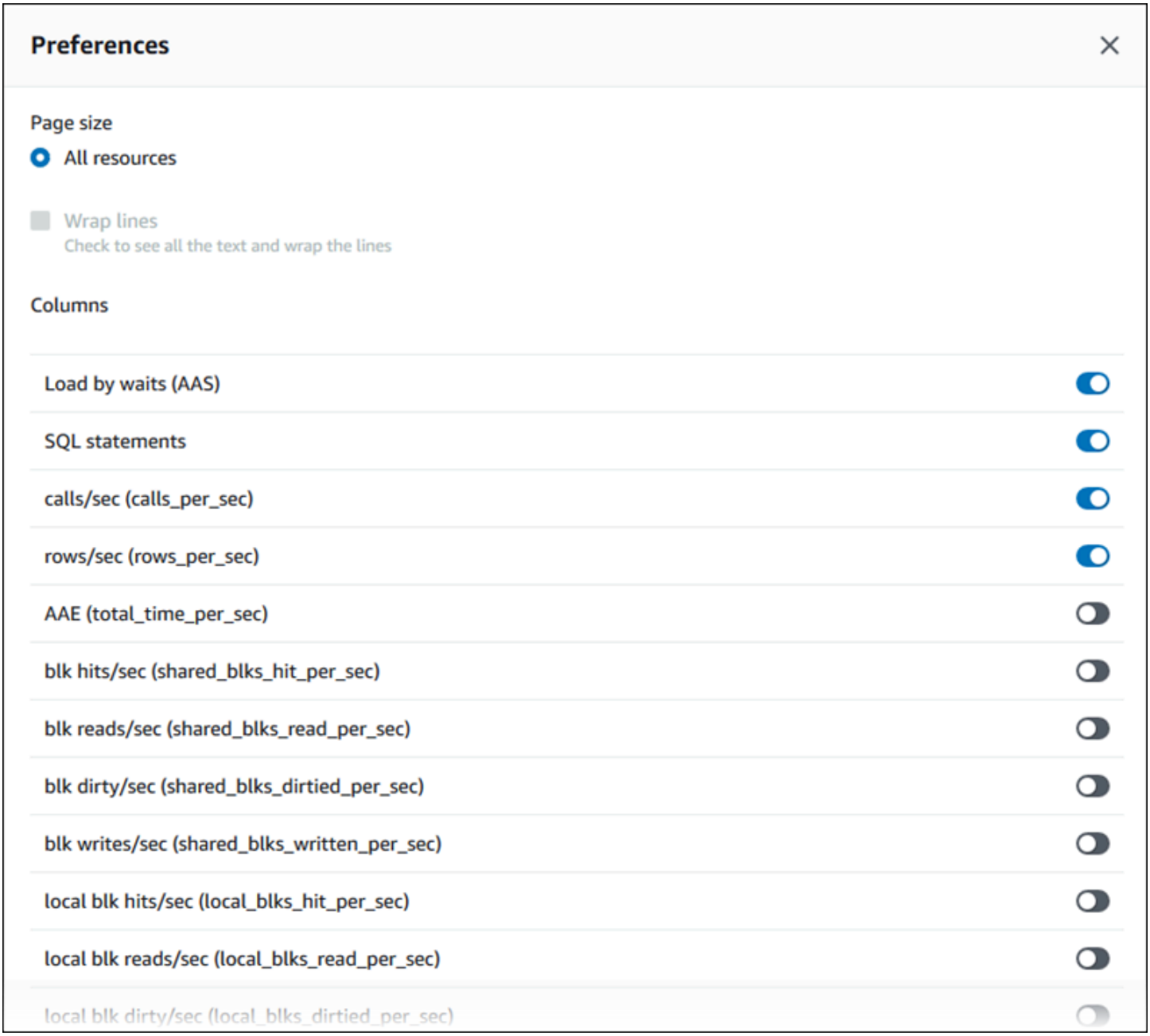
- Support SQL ID (Compatibilidad con ID SQL): un valor hash del ID de SQL. Este valor sirve solo para hacer referencia a un ID de SQL al trabajar con AWS Support. AWS Support no tiene acceso a sus ID de SQL y texto SQL reales.
- Support Digest ID (Compatibilidad con ID de resumen): un valor hash del ID de resumen. Este valor sirve solo para hacer referencia a un ID de resumen al trabajar con AWS Support. AWS Support no tiene acceso a sus ID de resumen y texto SQL reales.

Escoger preferencias de estadísticas

Para controlar las estadísticas mostradas en la pestaña Top SQL (SQL principal), puede elegir el icono Preferences (Preferencias).



Al seleccionar el icono Preferences (Preferencias), se abrirá la ventana Preferences (Preferencias). La siguiente captura de pantalla es un ejemplo de la ventana Preferences (Preferencias).



Para habilitar las estadísticas que desea que estén visibles en la pestaña Top SQL (SQL principal), utilice el ratón para desplazarse hasta la parte inferior de la ventana y, a continuación, elija Continue (Continuar).

Para obtener más información sobre las estadísticas por segundo o por llamada de los motores Aurora, consulte la sección de estadísticas SQL específicas del motor en Estadísticas de SQL para Performance Insights.

Acceso a más texto SQL en el panel de Performance Insights

De forma predeterminada, cada fila de la tabla Top SQL (SQL principal) muestra 500 bytes de texto SQL para cada instrucción SQL.

Cuando una instrucción SQL supera los 500 bytes, puede ver más texto en la sección SQL text (Texto SQL), bajo la tabla Top SQL (SQL principal). En este caso, la longitud máxima del texto que se muestra SQL text (Texto SQL) es de 4 KB. Este límite lo introduce la consola y está sujeto a los límites establecidos por el motor de base de datos. Para guardar el texto que se muestra en SQL text (Texto SQL), elija Download (Descargar).

Temas

- Límites de tamaño del texto para Aurora MySQL
- Ajuste del limite de texto SQL para las instancias de base de datos de Aurora PostgreSQL
- Ver y descargar texto SQL en el panel de Performance Insights

Límites de tamaño del texto para Aurora MySQL

Cuando se descarga un texto SQL, el motor de la base de datos determina su longitud máxima. Puede descargar texto SQL hasta los siguientes límites por motor:

Motor de base de datos	Longitud máxima del texto descargado
Aurora MySQL	La longitud está fijada en 4096 bytes.

En la sección SQL text (Texto SQL) de la consola de Performance Insights, se muestra el máximo que devuelve el motor. Por ejemplo, si Aurora MySQL devuelve como máximo 1 kB a Performance Insights, solo puede recopilar y mostrar 1 kB, incluso si la consulta original es de mayor longitud. Así, cuando se visualiza la consulta en SQL text (Texto SQL) o se descarga, Performance Insights devuelve el mismo número de bytes.

Si utiliza la AWS CLI o la API, Información de rendimiento no tiene el límite de 4 KB aplicado por la consola. `DescribeDimensionKeys` y `GetResourceMetrics` devuelven como máximo 500 bytes.

> **Note**
>
> `GetDimensionKeyDetails` devuelve la consulta completa, pero el tamaño está sujeto al límite del motor.

Ajuste del limite de texto SQL para las instancias de base de datos de Aurora PostgreSQL

Aurora PostgreSQL maneja el texto de manera diferente. Puede establecer el límite de tamaño del texto con el parámetro de instancia de base de datos `track_activity_query_size`. Este parámetro incluye las siguientes características:

Tamaño de texto predeterminado

En la versión 9.6 de Aurora PostgreSQL, la configuración predeterminada del parámetro `track_activity_query_size` es de 1024 bytes. En la versión 10 o superior de Aurora PostgreSQL, la configuración predeterminada del parámetro es de 4096 bytes.

Tamaño máximo del texto

El límite de `track_activity_query_size` para la versión 12 o inferior de Aurora PostgreSQL es de 102 400 bytes. El máximo es de 1 MB para la versión 13 y superior.

Si el motor devuelve 1 MB a Performance Insights, la consola muestra solo los primeros 4 kB. Si descarga la consulta, obtendrá 1 MB completo. En este caso, la visualización y la descarga devuelven diferentes cantidades de bytes. Para obtener más información sobre el parámetro de instancia de base de datos `track_activity_query_size`, consulte Run-time Statistics (Estadísticas de tiempo de ejecución) en la documentación de PostgreSQL.

Para aumentar el tamaño del texto SQL, aumente el límite de `track_activity_query_size`. Para modificar el parámetro, cambie el ajuste en el grupo de parámetros asociado a la instancia de base de datos de Aurora PostgreSQL.

Para cambiar la configuración cuando la instancia utiliza el grupo de parámetros predeterminado

1. Cree un nuevo grupo de parámetros de instancia de base de datos para el motor de base de datos y la versión del motor de base de datos adecuados.
2. Establezca el parámetro en el nuevo grupo de parámetros.
3. Asocie el nuevo grupo de parámetros a la instancia de base de datos.

Para obtener más información sobre configurar un parámetro de instancia de base de datos, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

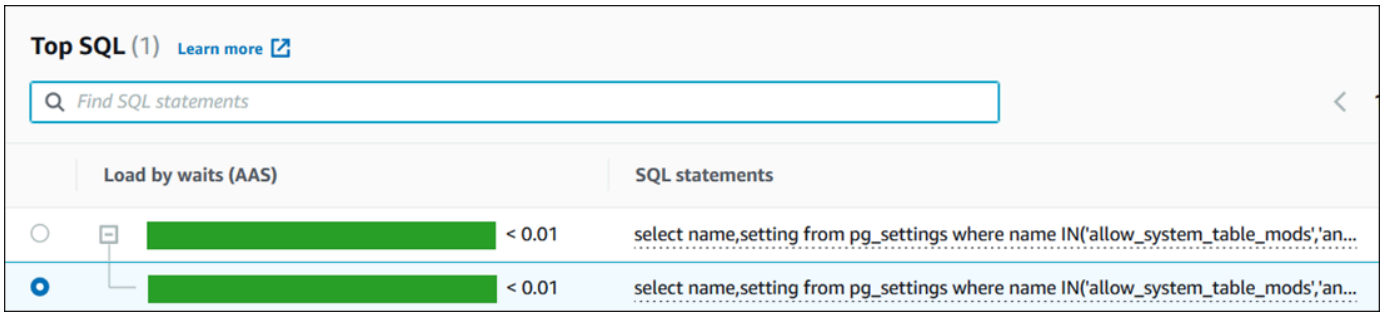
Ver y descargar texto SQL en el panel de Performance Insights

Puede ver o descargar texto SQL en el panel de Performance Insights.

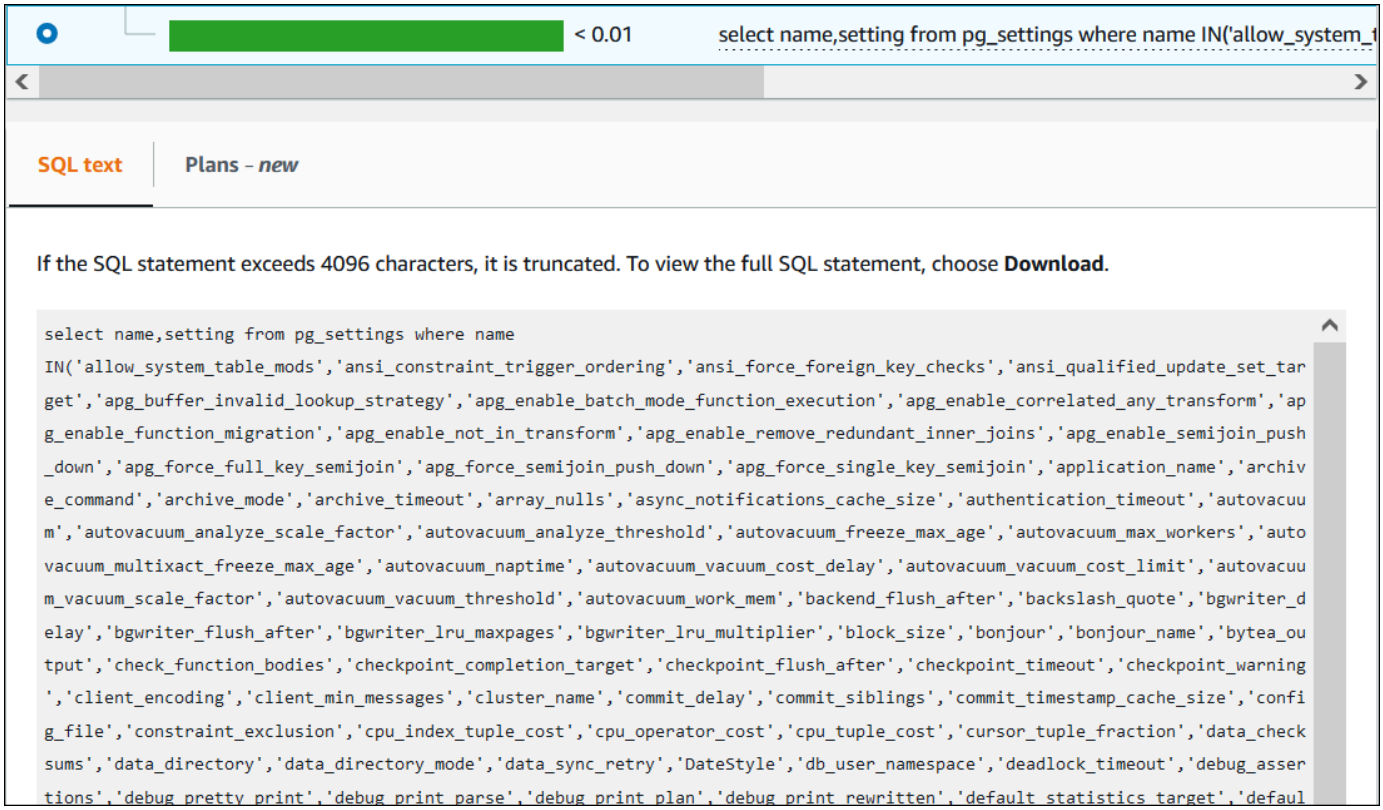
Para ver más texto SQL en el panel de Performance Insights

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Performance Insights.
3. Elija una instancia de base de datos.
4. Baje hasta la pestaña Top SQL en el panel de Información de rendimiento.
5. Elija el signo más para expandir un resumen de SQL y elija una de las consultas secundarias del resumen.

   Las instrucciones SQL con texto superior a 500 bytes son similares a las que se indican en la siguiente imagen.



6. Desplácese hasta la pestaña SQL text (Texto SQL).



El panel de Performance Insights puede mostrar hasta 4096 bytes por cada instrucción SQL.

7. (Opcional) Elija Copiar para copiar la instrucción SQL mostrada o elija Descargar para descargar la instrucción SQL para consultar el texto SQL hasta el límite del motor de base de datos.

> **Note**
>
> Para copiar o descargar la instrucción SQL, deshabilite los bloqueadores de pantallas emergentes.

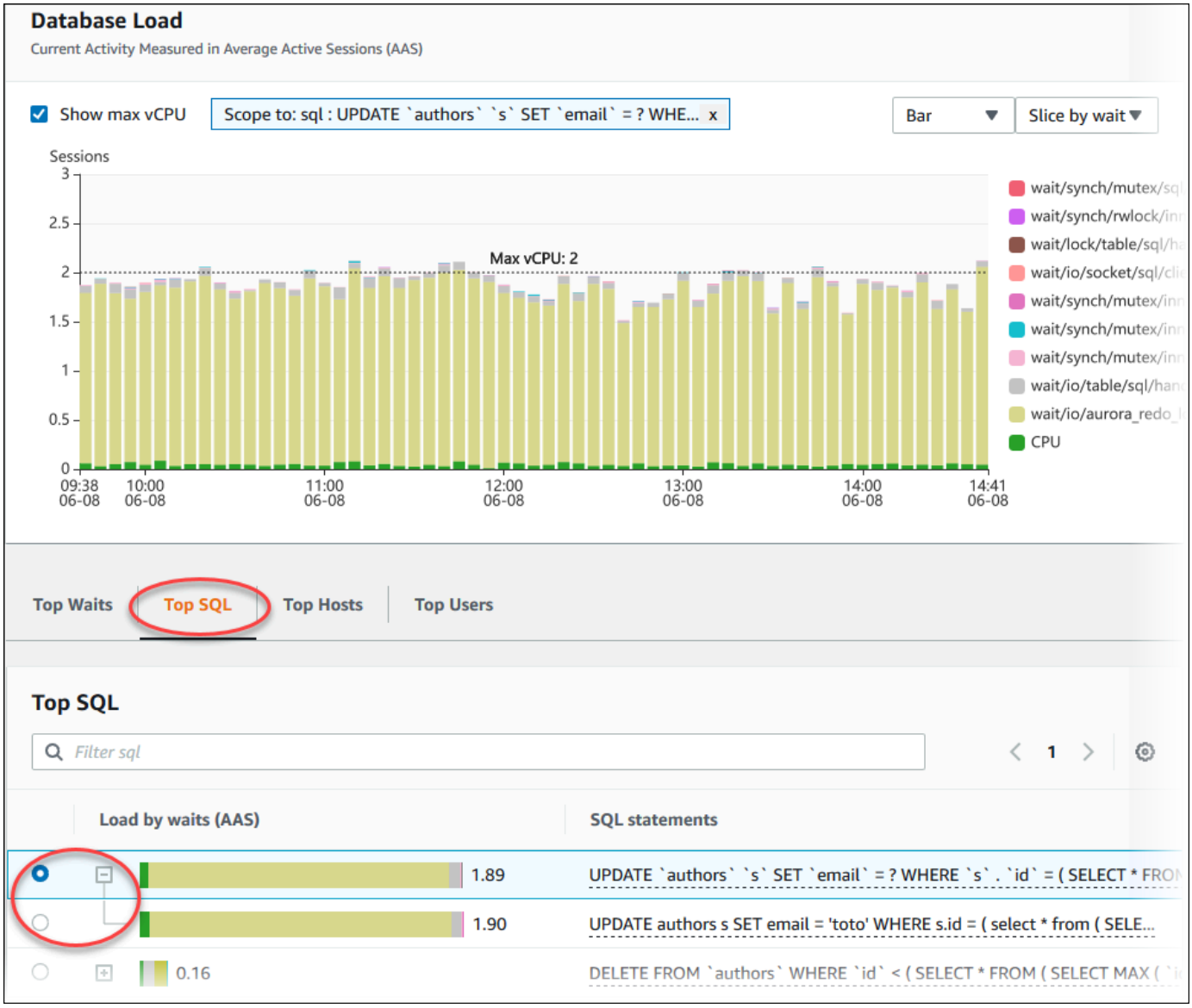
Visualización de estadísticas de SQL en el panel de Performance Insights

En el panel de Performance Insights, las estadísticas de SQL están disponibles en la pestaña Top SQL (SQL principal) del gráfico Database load (Carga de base de datos).

Para ver las estadísticas de SQL

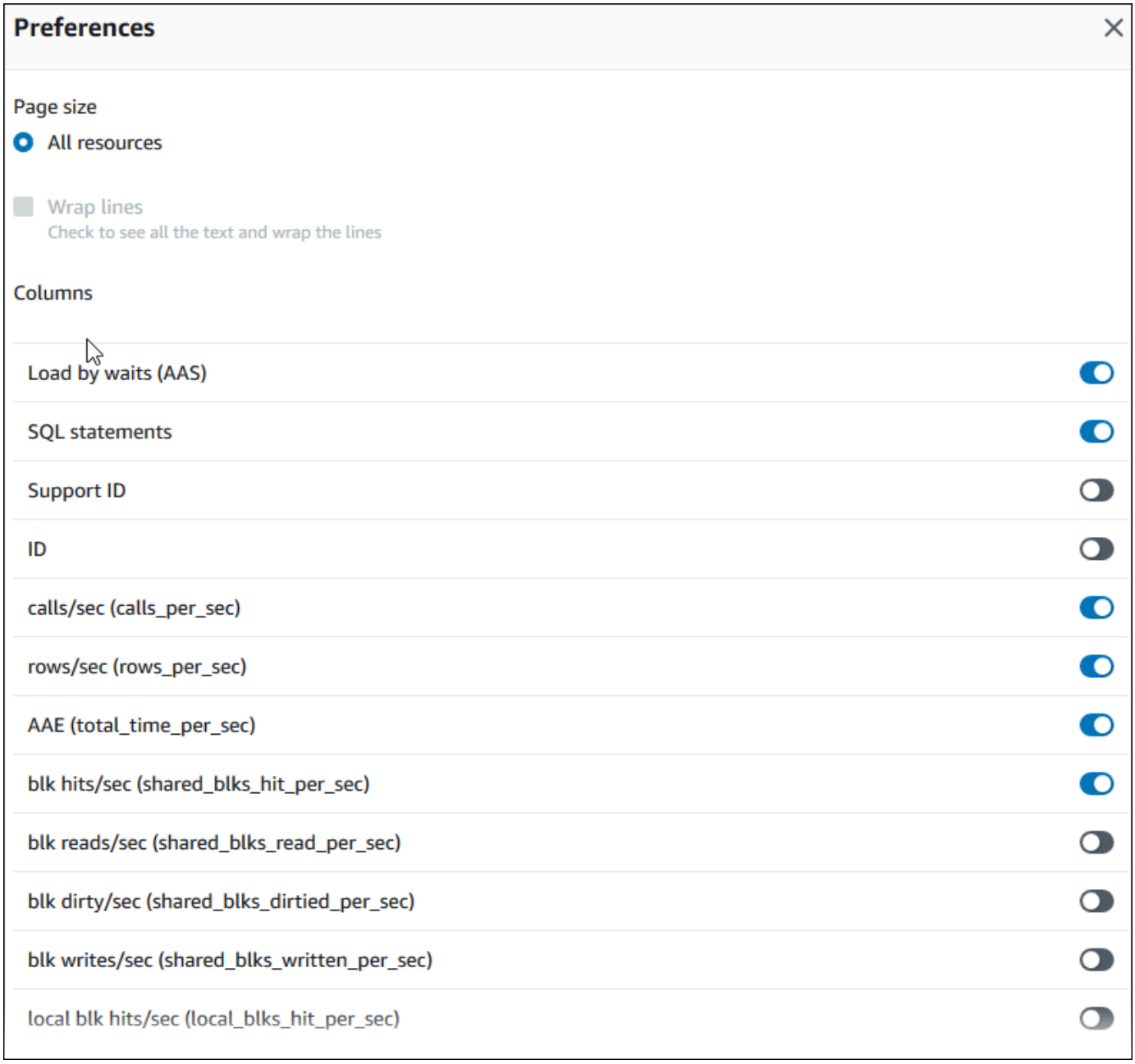
1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la izquierda, seleccione Performance Insights.

3. En la parte superior de la página, elija la base de datos cuyas estadísticas de SQL desea ver.
4. Desplácese a la parte inferior de la página y seleccione Top SQL (SQL principal).
5. Elija una instrucción individual (Solo Aurora MySQL) o consulte el resumen.

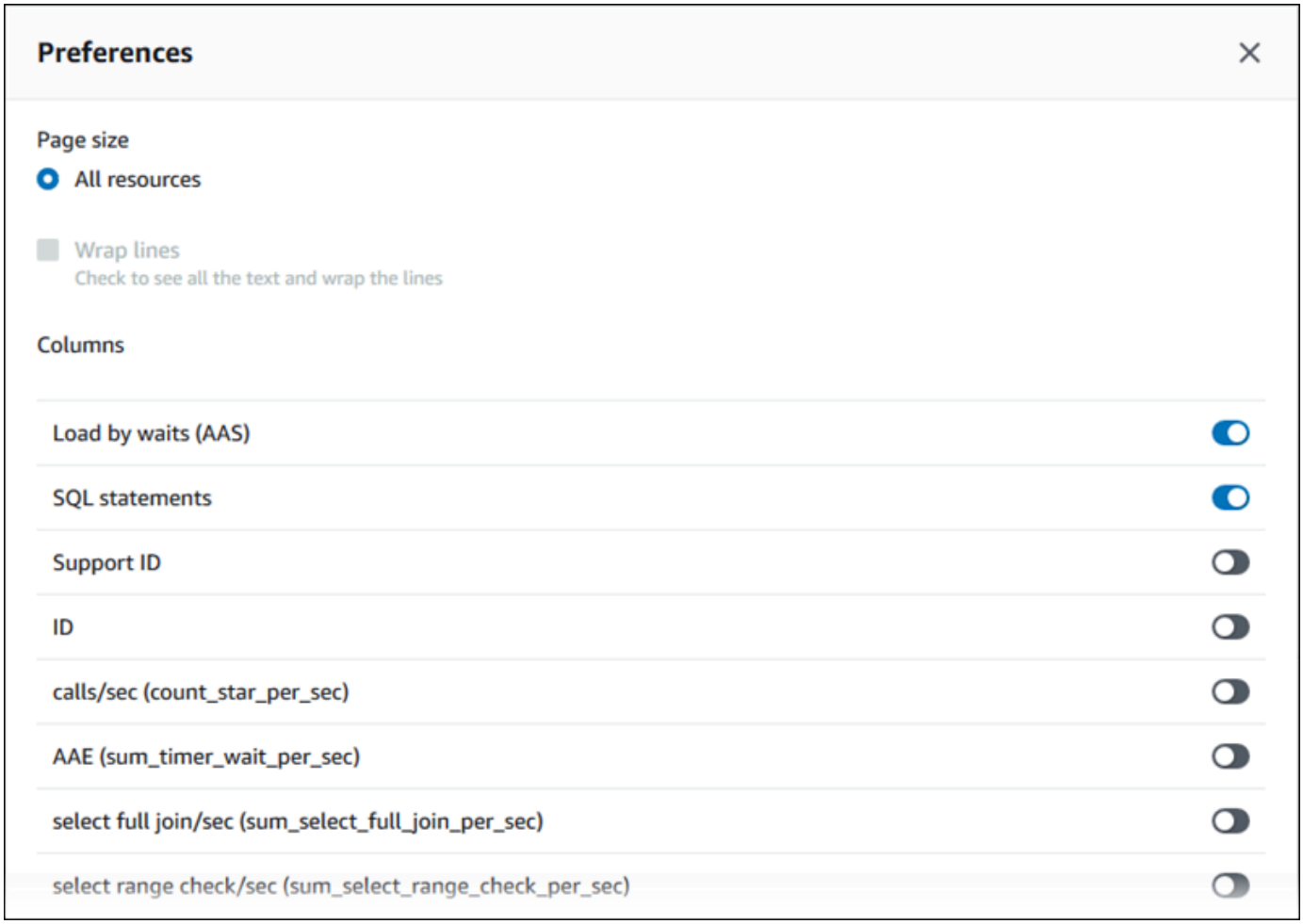


6. Seleccione qué estadísticas mostrar seleccionando el icono de engranaje de la esquina superior derecha del gráfico. Para obtener descripciones de las estadísticas de SQL de los motores de Amazon RDSAurora, consulte Estadísticas de SQL para Performance Insights.

   En el siguiente ejemplo se muestran las preferencias para Aurora PostgreSQL.



El siguiente ejemplo muestra las preferencias para las instancias de base de datos de Aurora MySQL.



7. Elija Save (Guardar) para guardar las preferencias.

   Se actualiza la tabla Top SQL (SQL principal).

Visualización de las recomendaciones proactivas de Información de rendimiento

Información de rendimiento de Amazon RDS monitoriza automáticamente métricas específicas y crea umbrales. Para ello, analiza qué niveles podrían ser potencialmente problemáticos para un recurso específico. Cuando los nuevos valores de las métricas cruzan un umbral predefinido durante un período de tiempo determinado, Información de rendimiento genera una recomendación proactiva. Esta recomendación ayuda a evitar que el rendimiento de la base de datos se vea afectado en el futuro. Para recibir estas recomendaciones proactivas, debe activar Información de rendimiento con un período de retención de nivel de pago.

Para obtener más información acerca de la activación de Información de rendimiento, consulte Activación y desactivación de Información de rendimiento de Aurora. Para obtener información sobre los precios y la retención de datos de Información de rendimiento, consulte Precios y retención de datos de Performance Insights.

Para obtener información sobre las regiones, los motores de bases de datos y las clases de instancias compatibles con las recomendaciones proactivas, consulte Compatibilidad del motor de la base de datos, la región y la clase de instancia de Amazon Aurora con características de Información de rendimiento.

Puede ver el análisis detallado y las investigaciones recomendadas de las recomendaciones proactivas en la página de detalles de las recomendaciones.

Para obtener más información sobre recomendaciones, consulte Recomendaciones para Amazon Aurora.

Para ver el análisis detallado de una recomendación proactiva

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, realice cualquiera de las siguientes acciones:

   - Elija Recomendaciones.

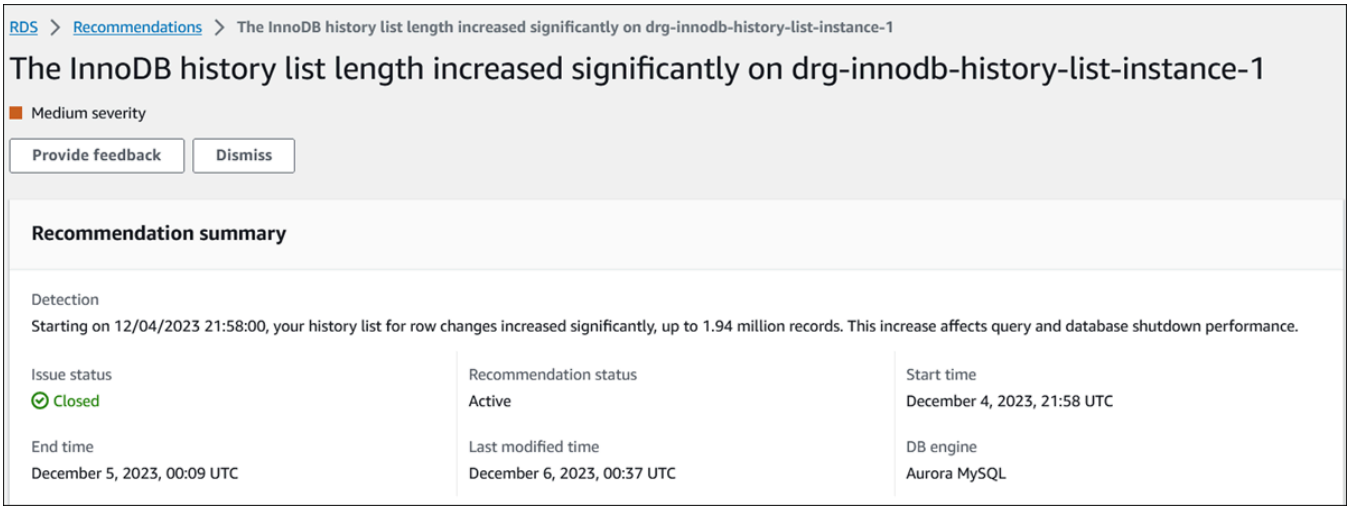
     En la página Recomendaciones, se muestra una lista de recomendaciones ordenadas por su gravedad para todos los recursos de su cuenta.
   - Elija Bases de datos y, a continuación, elija Recomendaciones para un recurso en la página de bases de datos.

     La pestaña Recomendaciones muestra las recomendaciones y sus detalles para el recurso seleccionado.
3. Busque una recomendación proactiva y elija Ver detalles.

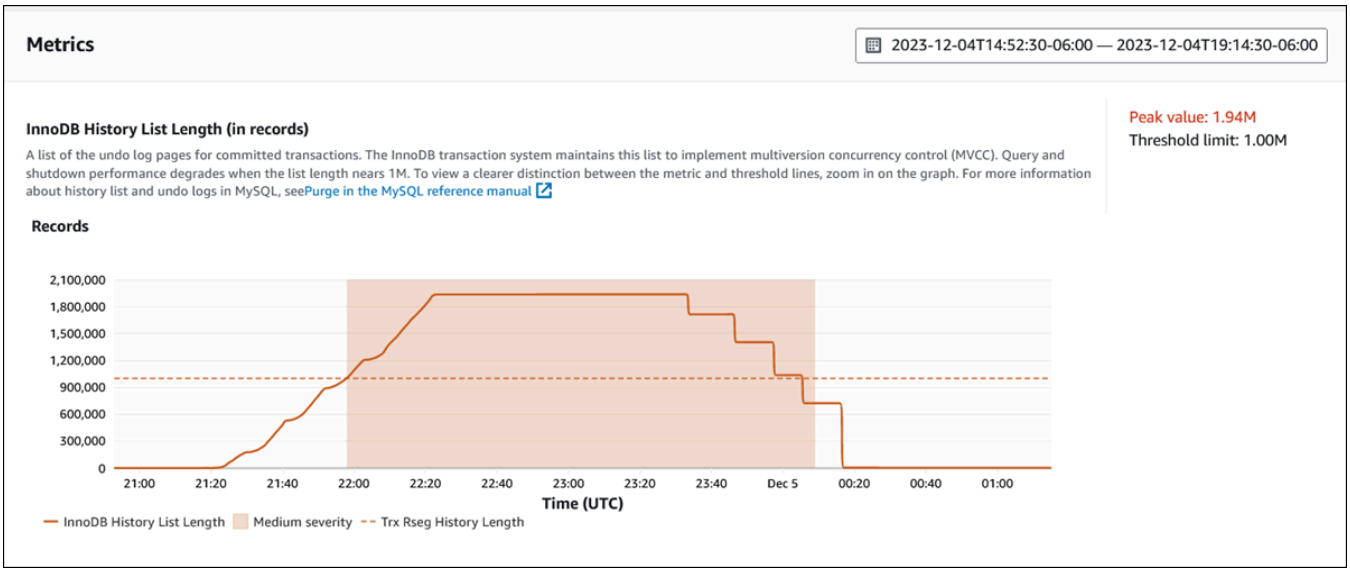
   Aparece la página de detalles de la recomendación. En el título se proporciona el nombre del recurso afectado con el problema detectado y su gravedad.

   A continuación figuran los componentes de la página de detalles de la recomendación:

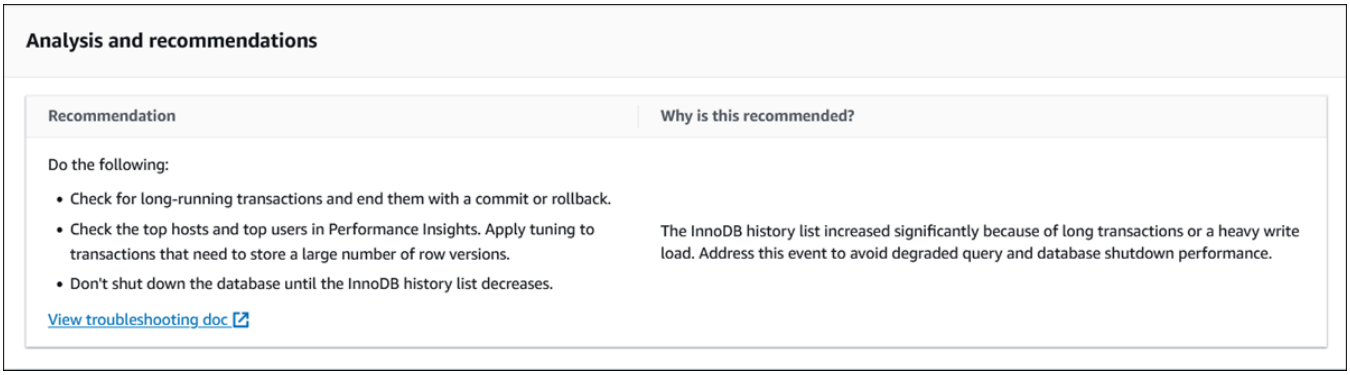
- Resumen de la recomendación: el problema detectado, el estado de la recomendación y el problema, la hora de inicio y finalización del problema, la hora de modificación de la recomendación y el tipo de motor.



- Métricas: los gráficos del problema detectado. Cada gráfico muestra un umbral determinado por el comportamiento de referencia del recurso, así como los datos de la métrica informados desde el momento de inicio de la anomalía.



- Análisis y recomendaciones: la recomendación y el motivo de la recomendación sugerida.



Puede revisar la causa del problema y, a continuación, realizar las acciones recomendadas para solucionarlo o seleccionar Descartar en la esquina superior derecha para descartar la recomendación.

Recuperación de métricas con la API de Información sobre rendimiento para Aurora

Cuando se activa la Información de rendimiento, la API proporciona visibilidad sobre el rendimiento de la instancia. registros de Amazon Cloudwatch proporciona la fuente autorizada de las métricas de monitoreo vendidas para servicios de AWS.

Con Performance Insights se ofrece una vista propia del dominio de la carga de la base de datos entendida como el promedio de sesiones activas (AAS). Esta métrica aparece para los consumidores de API como conjunto de datos de serie temporal bidimensional. La dimensión temporal de los datos ofrece datos de carga de base de datos para cada punto temporal del intervalo de tiempo consultado. Cada punto temporal descompone la carga global en relación con las dimensiones solicitadas, tales como `SQL`, `Wait-event`, `User` o `Host`, medidas en ese punto temporal.

La información sobre rendimiento de Amazon RDS monitorea el Amazon Aurora para poder analizar y solucionar los problemas de desempeño de la base de datos. Una forma de ver los datos de Performance Insights es a través de la Consola de administración de AWS. Performance Insights además ofrece una API pública, para poder consultar en sus propios datos. Puede utilizar la API para hacer lo siguiente:

- Descarga de datos en una base de datos
- Agregación de datos de Performance Insights a los paneles de monitoreo existentes
- Crear herramientas de monitoreo.

Para utilizar la API de Performance Insights, habilite Performance Insights en una de sus instancias de base de datos de Amazon RDS. Para obtener información sobre la habilitación de Performance Insights, consulte Activación y desactivación de Información de rendimiento de Aurora. Para obtener información sobre la API de Performance Insights, consulte la Referencia de la API de Performance Insights de Amazon RDS.

La API de Información sobre rendimiento proporciona las siguientes operaciones.

Acción de Performance Insights	AWS CLI command	Descripción
CreatePerformanceAnalysisReport	aws pi create-performance-analysis-report	Crea un informe de análisis de rendimiento para un período de tiempo específico para la instancia de base de datos. El resultado es AnalysisReportId que es el identificador único del informe.
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	Elimina un informe de análisis de rendimiento.
DescribeDimensionKeys	aws pi describe-dimension-keys	Recupera las principales claves de dimensión N de una métrica para un periodo de tiempo específico.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Recupera los atributos del grupo de dimension es especificado para una instancia de base de datos o un origen de datos. Por ejemplo, si especifica un ID de SQL y si los detalles de la dimensión están disponibles, GetDimensionKeyDetails recupera el texto

Acción de Performance Insights	AWS CLI command	Descripción
		completo de la dimensión `db.sql.statement` asociada a este ID. Esta operación resulta útil porque `GetResourceMetrics` y `DescribeDimensionKeys` no admiten la recuperación de texto de instrucción SQL grande.
`GetPerformanceAnalysisReport`	`aws pi get-performance-analysis-report`	Recupera el informe, incluidos los datos del informe. El resultado incluye el estado del informe, el identificador del informe, los detalles del tiempo del informe, la información y las recomendaciones.
`GetResourceMetadata`	`aws pi get-resource-metadata`	Recupere los metadatos de las distintas características. Por ejemplo, los metadatos podrían indicar que una característica está activada o desactivada en una instancia de base de datos específica.

Acción de Performance Insights	AWS CLI command	Descripción
GetResourceMetrics	aws pi get-resource-metrics	Recupera las métricas de Información sobre rendimiento de un conjunto de orígenes de datos, durante un periodo de tiempo. Puede proporcionar grupos de dimensiones y dimensiones específicas, y proporcionar criterios de agregación y filtrado para cada grupo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupere las dimensiones que se pueden consultar para cada tipo de métrica especificado en una instancia especificada.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupere todas las métricas disponibles de los tipos de métricas especificados que se pueden consultar de una instancia de base de datos especificada.
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	Recupera todos los informes de análisis disponibles para la instancia de base de datos. Los informes se enumeran en función de la hora de inicio de cada informe.

Acción de Performance Insights	AWS CLI command	Descripción
ListTagsForResource	aws pi list-tags-for-resource	Muestra todas las etiquetas de metadatos agregadas al recurso. La lista incluye el nombre y el valor de la etiqueta.
TagResource	aws pi tag-resource	Añade etiquetas de metadatos a un recurso de Amazon RDS. La etiqueta incluye un nombre y un valor.
UntagResource	aws pi untag-resource	Elimina la etiqueta de metadatos del recurso.

Para obtener más información sobre la recuperación de métricas de series temporales y ver ejemplos de la AWS CLI para Información de rendimiento, consulte los siguientes temas.

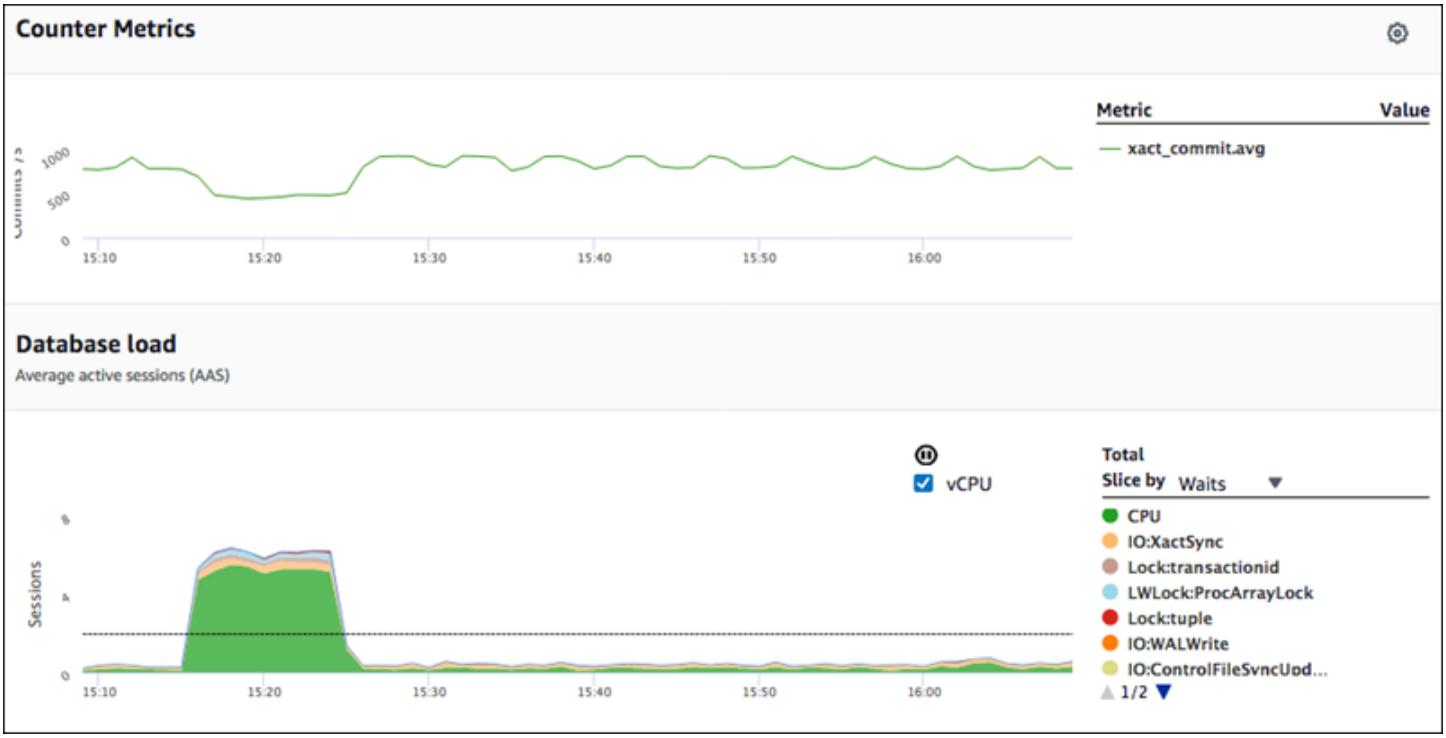
Temas

- Recuperación de métricas de series temporales para Información de rendimiento
- Ejemplos de la AWS CLI para Performance Insights

Recuperación de métricas de series temporales para Información de rendimiento

La operación `GetResourceMetrics` recupera una o más métricas de series temporales a partir de los datos de Performance Insights. `GetResourceMetrics` requiere una métrica y un periodo de tiempo y devuelve una respuesta con una lista de puntos de datos.

Por ejemplo, la Consola de administración de AWS usa `GetResourceMetrics` para completar el gráfico Counter Metrics (Métricas de contador) y el gráfico Database Load (Carga de la base de datos), como se muestra en la siguiente imagen.



Todas las métricas que devuelve `GetResourceMetrics` son métricas de series temporales estándar con la excepción de `db.load`. Esta métrica se muestra en el gráfico Database Load (Carga de base de datos). La métrica `db.load` es distinta de las demás métricas de series temporales porque puede desglosarla en subcomponentes llamados dimensiones. En la imagen anterior, `db.load` está desglosado y agrupado por los estados de espera que forman el `db.load`.

> **Note**
>
> `GetResourceMetrics` también puede devolver la métrica `db.sampleload`, pero la métrica `db.load` es apropiada en la mayoría de los casos.

Para obtener información sobre las métricas de contador devueltas por `GetResourceMetrics`, consulte Métricas de contador de Información de rendimiento.

Para las métricas se admiten los siguientes cálculos:

- Media: el valor medio de la métrica durante un período de tiempo. Añada `.avg` al nombre de la métrica.
- Mínimo: el valor mínimo de la métrica durante un período de tiempo. Añada `.min` al nombre de la métrica.

- Máximo: el valor máximo de la métrica durante un período de tiempo. Añada `.max` al nombre de la métrica.
- Suma: la suma de los valores de la métrica durante un periodo de tiempo. Añada `.sum` al nombre de la métrica.
- Número de muestras: El número de veces que se recopiló la métrica durante un período de tiempo. Añada `.sample_count` al nombre de la métrica.

Supongamos, por ejemplo, que una métrica se recopila durante 300 segundos (5 minutos) y que la métrica se recopila una vez cada minuto. Los valores para cada minuto son 1, 2, 3, 4 y 5. En este caso, se devuelven los siguientes cálculos:

- Media: 3
- Mínimo: 1
- Máximo: 5
- Suma: 15
- Número de muestras: 5

Para obtener información acerca del uso del comando `get-resource-metrics` de la AWS CLI, consulte `get-resource-metrics`.

Para la opción `--metric-queries`, especifique una o más consultas para las que desea obtener resultados. Cada consulta consta de un parámetro `Metric` obligatorio y de parámetros opcionales `GroupBy` y `Filter`. A continuación, se muestra un ejemplo de una especificación de opción `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

Ejemplos de la AWS CLI para Performance Insights

En las siguientes secciones, obtendrá más información sobre AWS Command Line Interface (AWS CLI) para Información de rendimiento, así como ejemplos de uso de la AWS CLI.

Temas

- Ayuda integrada en la AWS CLI para Información de rendimiento
- Recuperación de métricas de contador
- Recuperación del promedio de carga de base de datos para los eventos de espera principales
- Recuperación del promedio de carga de base de datos para las instrucciones SQL principales
- Recuperación del promedio de carga de base de datos filtrado por SQL
- Recuperación del texto completo de una instrucción SQL
- Creación de un informe de análisis de rendimiento para un período de tiempo
- Recuperación de un informe de análisis de rendimiento
- Enumeración de todos los informes de análisis de rendimiento de la instancia de base de datos
- Eliminación de un informe de análisis de rendimiento
- Adición de una etiqueta a un informe de análisis de rendimiento
- Enumeración de todas las etiquetas de un informe de análisis de rendimiento
- Eliminación de etiquetas de un informe de análisis de rendimiento

Ayuda integrada en la AWS CLI para Información de rendimiento

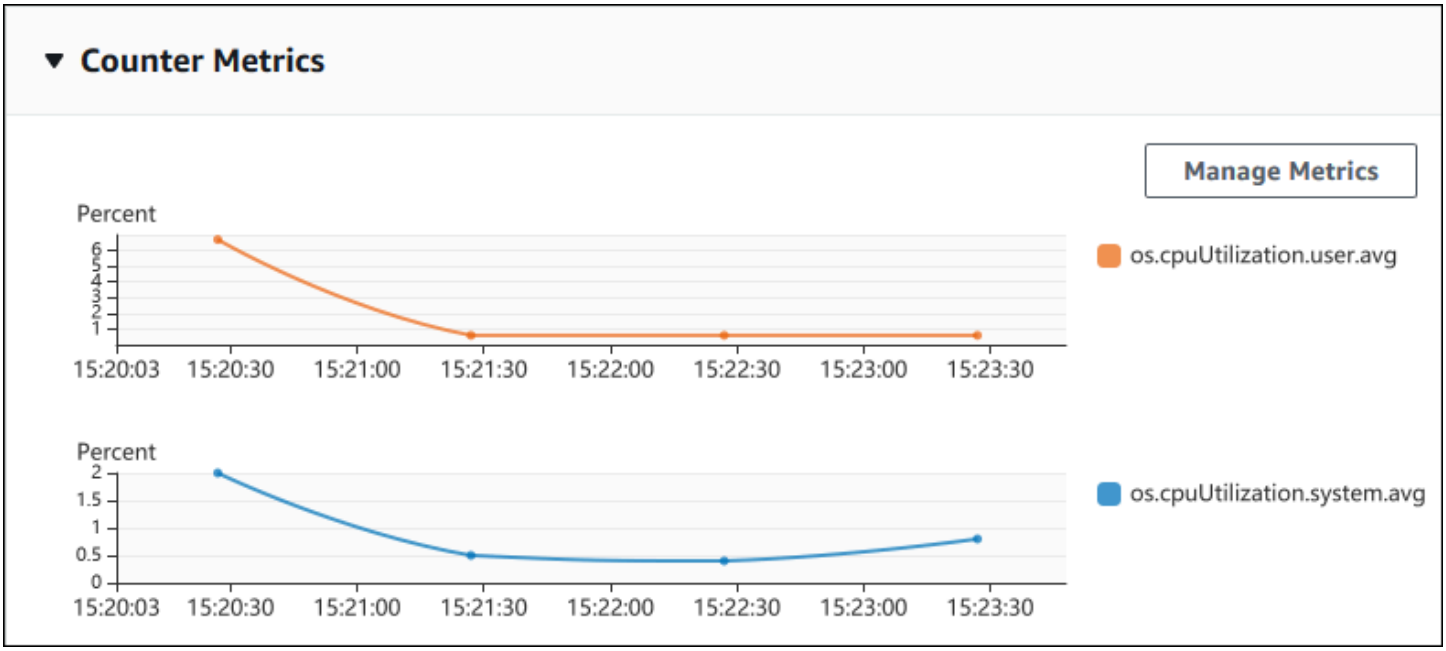
Puede ver los datos de Performance Insights a través de la AWS CLI. Puede obtener ayuda sobre los comandos de la AWS CLI de Performance Insights escribiendo lo siguiente en la línea de comandos.

```
aws pi help
```

Si no tiene instalada la AWS CLI, consulte Instalación de la AWS CLI en la Guía del usuario de la AWS CLI para obtener información sobre cómo instalarla.

Recuperación de métricas de contador

La siguiente captura de pantalla muestra dos gráficos de métricas de contador en la Consola de administración de AWS.



El siguiente ejemplo muestra cómo recopilar los mismos datos que utiliza la Consola de administración de AWS para generar los dos gráficos de métricas de contador.

Para Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

También puede hacer que un comando sea más fácil de leer especificando un archivo para la opción `--metrics-query`. El siguiente ejemplo utiliza un archivo llamado query.json para la opción. El archivo tiene el siguiente contenido.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Ejecute el siguiente comando para utilizar el archivo.

Para Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

El ejemplo anterior especifica los siguientes valores para las opciones:

- `--service-type`: RDS para Amazon RDS
- `--identifier`: el ID de recurso para la instancia de base de datos
- `--start-time` y:`--end-time` los valores ISO 8601 `DateTime` para el periodo de consulta, con varios formatos admitidos

Consulta durante un intervalo de una hora:

- `--period-in-seconds`: `60` para una consulta por minuto
- `--metric-queries`: una matriz de dos consultas, cada una para una métrica.

  El nombre de la métrica utiliza puntos para clasificar la métrica en categorías útiles y el elemento final es una función. En el ejemplo, la función es `avg` para cada consulta. Al igual que con Amazon CloudWatch, las funciones admitidas son `min`, `max`, `total` y `avg`.

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
```

```
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

La respuesta tiene `Identifier`, `AlignedStartTime` y `AlignedEndTime`. Como el valor `--period-in-seconds` era `60`, los tiempos de inicio y final se han alineado con el minuto. Si el `--period-in-seconds` fuera `3600`, los tiempos de inicio y final se habrían alineado con la hora.

La `MetricList` en la respuesta tiene una serie de entradas, cada una con una entrada `Key` y una entrada `DataPoints`. Cada `DataPoint` tiene un `Timestamp` y un `Value`. Cada lista de `Datapoints` tiene 60 puntos de datos porque las consultas son datos por minuto sobre una hora, con `Timestamp1/Minute1`, `Timestamp2/Minute2` y así sucesivamente, hasta `Timestamp60/Minute60`.

Como la consulta es para dos métricas de contador distintas, hay dos elementos en la respuesta `MetricList`.

Recuperación del promedio de carga de base de datos para los eventos de espera principales

El siguiente ejemplo es la misma consulta que utiliza la Consola de administración de AWS para generar un gráfico de línea de área apilada. Este ejemplo recupera el `db.load.avg` durante la última hora con la carga dividida según los siete eventos de espera principales. El comando es el mismo que el comando en Recuperación de métricas de contador. Sin embargo, el archivo query.json tiene los elementos indicados a continuación.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
```

```
]
```

Ejecute el comando siguiente.

Para Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

El ejemplo especifica la métrica de `db.load.avg` y un `GroupBy` de los siete eventos de espera principales. Para obtener detalles acerca de los valores válidos para este ejemplo, consulte DimensionGroup en la Referencia de la API de Performance Insights.

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
```

```
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
} //end of response
```

En esta respuesta, hay ocho entradas en la `MetricList`. Hay una entrada para el `db.load.avg` total y siete entradas para el `db.load.avg` divididas según uno de los siete eventos de espera principales. A diferencia del primer ejemplo, como había una dimensión de agrupación, debe haber una clave para cada agrupación de la métrica. No puede haber solo una clave para cada métrica, como en el caso de uso de métrica de contador básica.

Recuperación del promedio de carga de base de datos para las instrucciones SQL principales

El siguiente ejemplo agrupa `db.wait_events` por las 10 instrucciones SQL principales. Hay dos grupos distintos para instrucciones SQL:

- `db.sql` – la instrucción SQL completa, como `select * from customers where customer_id = 123`
- `db.sql_tokenized` – la instrucción SQL tokenizada, como `select * from customers where customer_id = ?`

Al analizar el desempeño de la base de datos, puede resultar útil tener en cuenta instrucciones SQL que solo se diferencien en sus parámetros como un elemento de lógica. Así pues, puede utilizar `db.sql_tokenized` al consultar. Sin embargo, sobre todo cuando le interese explicar planes, a veces es más útil examinar instrucciones SQL completas con parámetros y consultar agrupando por `db.sql`. Existe una relación principal-secundaria entre instrucciones SQL tokenizadas y completas, con varias instrucciones SQL completas (secundarias) agrupadas bajo la misma instrucción SQL tokenizada (principal).

El comando en este ejemplo es similar al comando en Recuperación del promedio de carga de base de datos para los eventos de espera principales. Sin embargo, el archivo query.json tiene los elementos indicados a continuación.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
    }
]
```

El siguiente ejemplo utiliza `db.sql_tokenized`.

Para Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-29T00:00:00Z \
   --end-time   2018-10-30T00:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-29T00:00:00Z ^
   --end-time   2018-10-30T00:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

Este ejemplo consulta durante 24 horas, con un periodo de una hora en segundos.

El ejemplo especifica la métrica de `db.load.avg` y un `GroupBy` de los siete eventos de espera principales. Para obtener detalles acerca de los valores válidos para este ejemplo, consulte DimensionGroup en la Referencia de la API de Performance Insights.

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
```

```
                },
                //... 24 datapoints
            ]
        },
        {
            "Key": { //Next key is the top tokenized SQL
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

Esta respuesta tiene 11 entradas en la `MetricList` (1 total, 10 SQL tokenizadas principales) y cada entrada tiene 24 `DataPoints` por hora.

Para consultas SQL tokenizadas, hay tres entradas en cada lista de dimensiones:

- `db.sql_tokenized.statement`: la instrucción SQL tokenizada.
- `db.sql_tokenized.db_id` : el ID de base de datos nativo utilizado para hacer referencia a SQL, o un ID sintético que genera Performance Insights para usted si no se encuentra disponible el ID de base de datos nativo. Este ejemplo devuelve el ID sintético de `pi-2372568224`.
- `db.sql_tokenized.id`: el ID de la consulta dentro del panel Performance Insights.

  En la Consola de administración de AWS, este ID se denomina ID de soporte. Se denomina así porque el ID es sobre datos que AWS Support puede examinar para ayudarle a solucionar un problema con la base de datos. AWS se toma muy en serio la seguridad y privacidad de sus datos, y casi todos los datos se almacenan encriptados con su clave AWS KMS. Por lo tanto, nadie dentro de AWS puede ver estos datos. En el ejemplo anterior, tanto `tokenized.statement` como `tokenized.db_id` se almacenan cifrados. Si tiene un problema con su base de datos, AWS Support puede ayudarle, ya que hace referencia al ID de Support.

Al realizar consultas, puede ser conveniente especificar un `Group` en `GroupBy`. Sin embargo, para un control de más precisión sobre los datos que se devuelven, especifique la lista de dimensiones. Por ejemplo, si todo lo que se necesita es `db.sql_tokenized.statement`, entonces se puede añadir un atributo `Dimensions` al archivo query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recuperación del promedio de carga de base de datos filtrado por SQL

La imagen anterior muestra que se ha seleccionado una consulta concreta y el gráfico de línea de área apilada de principales sesiones activas promedio se limita a esa consulta. Aunque se siguen consultando los siete eventos de espera generales principales, se filtra el valor de la respuesta. El filtro hace que solo tenga en cuenta las sesiones que coinciden con el filtro concreto.

La consulta de API correspondiente en este ejemplo es similar al comando en Recuperación del promedio de carga de base de datos para las instrucciones SQL principales. Sin embargo, el archivo query.json tiene los elementos indicados a continuación.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Para Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Para Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
```

```
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
```

```
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
```

```
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

En esta respuesta, todos los valores se filtran según la contribución del SQL tokenizado AKIAIOSFODNN7EXAMPLE especificado en el archivo query.json. Las claves también podrían seguir un orden distinto de una consulta sin un filtro, porque el SQL filtrado afectaba a los cinco eventos de espera principales.

Recuperación del texto completo de una instrucción SQL

En el siguiente ejemplo se recupera el texto completo de una instrucción SQL para una instancia de base de datos `db-10BCD2EFGHIJ3KL4M5NO6PQRS5`. El `--group` es `db.sql` y el `--group-identifier` es `db.sql.id`. En este ejemplo, *`my-sql-id`* representa un ID de SQL recuperado al invocar a `pi get-resource-metrics` o`pi describe-dimension-keys`.

Ejecute el siguiente comando de la .

Para Linux, macOS o Unix:

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
   --group db.sql \
```

```
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

Para Windows:

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
   --requested-dimensions statement
```

En este ejemplo, los detalles de las dimensiones están disponibles. Por lo tanto, Performance Insights recupera el texto completo de la instrucción SQL, sin truncarlo.

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

Creación de un informe de análisis de rendimiento para un período de tiempo

En el siguiente ejemplo, se crea un informe de análisis de rendimiento con la hora de inicio `1682969503` y la hora de finalización `1682979503` para la base de datos `db-loadtest-0`.

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

La respuesta es el identificador único `report-0234d3ed98e28fb17` para el informe.

```
{
```

```
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

Recuperación de un informe de análisis de rendimiento

En el siguiente ejemplo se recuperan los detalles del informe de análisis del informe `report-0d99cc91c4422ee61`.

```
aws pi get-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

La respuesta proporciona el estado del informe, el identificador, los detalles del tiempo y la información.

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

Enumeración de todos los informes de análisis de rendimiento de la instancia de base de datos

En el siguiente ejemplo se enumeran todos los informes de análisis de rendimiento disponibles para la base de datos `db-loadtest-0`.

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
--region us-west-2
```

La respuesta enumera todos los informes con el ID del informe, el estado y los detalles del período de tiempo.


```
{
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
            "StartTime": 1680583486.584,
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```


Eliminación de un informe de análisis de rendimiento

En el siguiente ejemplo, se elimina el informe de análisis de la base de datos `db-loadtest-0`.

```
aws pi delete-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
```

```
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

Adición de una etiqueta a un informe de análisis de rendimiento

En el siguiente ejemplo, se agrega una etiqueta con una clave `name` y valor `test-tag` al informe `report-01ad15f9b88bcbd56`.

```
aws pi tag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

Enumeración de todas las etiquetas de un informe de análisis de rendimiento

En el ejemplo siguiente se enumeran todas las etiquetas del informe `report-01ad15f9b88bcbd56`.

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

La respuesta enumera el valor y la clave de todas las etiquetas agregadas al informe:

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

Eliminación de etiquetas de un informe de análisis de rendimiento

En el siguiente ejemplo se elimina la etiqueta `name` de un informe `report-01ad15f9b88bcbd56`.

```
aws pi untag-resource \
```

```
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tag-keys name \
--region us-west-2
```

Después de eliminar la etiqueta, llamar a la API `list-tags-for-resource` no muestra esta etiqueta.

Registro de llamadas de Performance Insights mediante el uso de AWS CloudTrail

Performance Insights se ejecuta con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un servicio de AWS en Performance Insights. CloudTrail captura todas las llamadas a las API para Performance Insights como eventos. Esta captura incluye llamadas desde la consola de Amazon RDS y desde llamadas de código a las operaciones de la API de Performance Insights.

Si crea un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, incluidos eventos para Performance Insights. Si no configura un registro de seguimiento, puede ver los eventos más recientes en la consola de CloudTrail en el Event history (Historial de eventos). Mediante la información recopilada por CloudTrail podrá determinar ciertos detalles. Esta información incluye la solicitud que se envió a Performance Insights, la dirección IP desde la que se realizó la solicitud, quién realizó la solicitud y cuándo se realizó. También incluye detalles adicionales.

Para obtener más información sobre CloudTrail, consulte la Guía del usuario de AWS CloudTrail.

Trabajar con datos de Performance Insights en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce actividad en Performance Insights, esa actividad se registra en un evento de CloudTrail junto con otros eventos de servicios de AWS en la consola de CloudTrail en el historial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte Ver eventos con el historial de eventos de CloudTrail en la guía del usuario de AWS CloudTrail.

Para mantener un registro continuo de los eventos de su cuenta de AWS, incluidos los eventos de Performance Insights, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. De manera predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a

todas las regiones de AWS. El seguimiento registra los eventos de todas las regiones de AWS en la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte los siguientes temas en la guía del usuario de:AWS CloudTrail

- Introducción a la creación de registros de seguimiento
- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recepción de archivos de registro de CloudTrail de varias regiones y Recepción de archivos de registro de CloudTrail de varias cuentas

CloudTrail registra todas las operaciones de Performance Insights que se documentan en la Referencia de la API de Performance Insights. Por ejemplo, las llamadas a las operaciones `DescribeDimensionKeys` y `GetResourceMetrics` generan entradas en los archivos de registro de CloudTrail.

Cada entrada de registro o evento contiene información acerca de quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario de IAM.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail.

Entradas del archivo de registro de Performance Insights

Un seguimiento es una configuración que permite la entrega de eventos como archivos de registro en un bucket de Amazon S3 que especifique. Los archivos de registro de CloudTrail contienen una o varias entradas de registro. Un evento representa una única solicitud desde cualquier origen. Cada evento incluye información acerca de la operación solicitada, la fecha y la hora de la operación, los parámetros de la solicitud, etc. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas públicas a la API, por lo que estas no aparecen en ningún orden específico.

En el ejemplo que sigue se muestra una entrada de registro de CloudTrail que ilustra la operación `GetResourceMetrics`.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

API de Información de rendimiento y puntos de conexión de VPC de interfaz (AWS PrivateLink)

Puede usar AWS PrivateLink para crear una conexión privada entre la VPC e Información de rendimiento de Amazon RDS. Puede acceder a Información de rendimiento como si estuviera en su VPC, sin utilizar una puerta de enlace de Internet, un dispositivo NAT, una conexión VPN o una conexión Direct Connect. Las instancias de la VPC no necesitan direcciones IP públicas para acceder a Información de rendimiento.

Esta conexión privada se establece mediante la creación de un punto de conexión de interfaz alimentado por AWS PrivateLink. Creamos una interfaz de red de punto de conexión en cada subred habilitada para el punto de conexión de interfaz. Se trata de interfaces de red administradas por el solicitante que sirven como punto de entrada para el tráfico destinado a Información de rendimiento.

Para obtener más información, consulte Acceso a los Servicios de AWS a través de AWS PrivateLink en la Guía de AWS PrivateLink.

Consideraciones sobre Información de rendimiento

Antes de configurar un punto de conexión de interfaz para Información de rendimiento, consulte la sección Considerations en la Guía de AWS PrivateLink.

Información de rendimiento admite la realización de llamadas a todas las acciones de la API a través del punto de conexión de interfaz.

De forma predeterminada, el acceso completo a Información de rendimiento se permite a través del punto de conexión de interfaz. Para controlar el tráfico a Información de rendimiento a través del punto de conexión de interfaz, asocie un grupo de seguridad a las interfaces de red de los puntos de conexión.

Disponibilidad

La API de Información de rendimiento actualmente admite puntos de conexión de VPC en Regiones de AWS que admiten Información de rendimiento. Para obtener más información acerca de la disponibilidad de Información de rendimiento, consulte Regiones y motores de base de datos Aurora para Información sobre rendimiento.

Creación de un punto de conexión de interfaz para Información de rendimiento

Puede crear un punto de conexión de interfaz para Información de rendimiento mediante la consola de Amazon VPC o la AWS Command Line Interface (AWS CLI). Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía de AWS PrivateLink.

Cree un punto de conexión para Información de rendimiento utilizando el siguiente nombre de servicio:

Si habilita el DNS privado para el punto de conexión de interfaz, puede realizar solicitudes a la API para Información de rendimiento usando su nombre de DNS predeterminado para la región. Por ejemplo, `pi.us-east-1.amazonaws.com`.

Creación de una política de puntos de conexión de VPC para la API de Información de rendimiento

Una política de punto de conexión es un recurso de IAM que puede adjuntar a un punto de conexión de interfaz. La política de puntos de conexión predeterminada permite acceso completo a Información de rendimiento a través del punto de conexión de interfaz. Para controlar el acceso permitido a Información de rendimiento desde la VPC, adjunte una política de puntos de conexión personalizada al punto de conexión de interfaz.

Una política de punto de conexión especifica la siguiente información:

- Las entidades principales que pueden llevar a cabo acciones (Cuentas de AWS, usuarios de IAM y roles de IAM).
- Las acciones que se pueden realizar.
- El recurso en el que se pueden realizar las acciones.

Para obtener más información, consulte Control del acceso a los servicios con políticas de punto de conexión en la Guía del usuario de AWS PrivateLink.

Ejemplo: Política de punto de conexión de VPC para acciones de Información de rendimiento

El siguiente es un ejemplo de una política de un punto de conexión personalizado. Cuando se asocia con un punto de conexión, esta política concede acceso a las acciones de Información de rendimiento mostradas para todas las entidades principales en todos los recursos.

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreatePerformanceAnalysisReport",
            "rds:DeletePerformanceAnalysisReport",
            "rds:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

Ejemplo: política de punto de enlace de la VPC que deniega todo el acceso desde una cuenta de AWS especificada

La siguiente política de punto de enlace de la VPC deniega a la cuenta de AWS `123456789012` todo el acceso a los recursos mediante el punto de enlace. La política permite todas las acciones de otras cuentas.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Direcciones IP de Información de rendimiento

Las direcciones IP permiten que los recursos de la VPC se comuniquen entre sí y con otros recursos a través de Internet. Información de rendimiento admite los protocolos de direcciones IPv4 e

IPv6. De forma predeterminada, Información de rendimiento y Amazon VPC utilizan el protocolo de direccionamiento IPv4. No puedes desactivar este comportamiento. Al crear una VPC, debe especificar un bloque de CIDR IPv4 (un intervalo de direcciones IPv4 privadas).

De manera opcional, puede asociar un bloque de CIDR IPv6 a su VPC y sus subredes y asignar direcciones IPv6 de dicho bloque a recursos de RDS de su subred. La compatibilidad con el protocolo IPv6 amplía el número de direcciones IP admitidas. Al utilizar el protocolo IPv6, se asegura de tener suficientes direcciones disponibles para el futuro crecimiento de Internet. Los recursos de RDS nuevos y existentes pueden utilizar direcciones IPv4 e IPv6 dentro de su VPC. Configurar, proteger y traducir el tráfico de red entre los dos protocolos utilizados en diferentes partes de una aplicación puede provocar sobrecarga operativa. Puede estandarizar el protocolo IPv6 para los recursos de Amazon RDS para simplificar la configuración de la red. Para obtener información sobre los puntos de conexión y las cuotas de servicios, consulte Amazon Relational Database Service endpoints and quotas.

Para obtener más información sobre el direccionamiento IP de Aurora, consulte Direccionamiento IP de Aurora.

Análisis de anomalías de rendimiento de Aurora con Amazon DevOps Guru para Amazon RDS

Amazon DevOps Guru es un servicio de operaciones totalmente administrado que ayuda a los desarrolladores y operadores a mejorar el rendimiento y la disponibilidad de sus aplicaciones. DevOps Guru descarga las tareas asociadas a la identificación de problemas operativos para que pueda implementar rápidamente recomendaciones para mejorar su aplicación. Para obtener más información, consulte What is Amazon DevOps Guru? (¿Qué es Amazon DevOps Guru?) en la Guía del usuario de Amazon DevOps Guru.

DevOps Guru detecta, analiza y hace recomendaciones sobre problemas operativos existentes para todos los motores de base de datos de Amazon RDS. Para ampliar esta capacidad, DevOps Guru para RDS aplica machine learning a las métricas de Información sobre rendimiento de bases de datos Amazon Aurora . Estas funciones de monitoreo permiten a DevOps Guru for RDS detectar y diagnosticar cuellos de botella en el rendimiento y recomendar acciones correctivas específicas. DevOps Guru para RDS también puede detectar condiciones problemáticas en las bases de datos Aurora antes de que se produzcan.

Ahora puede ver estas recomendaciones en la consola de RDS. Para obtener más información, consulte Recomendaciones para Amazon Aurora.

> **Important**
>
> Amazon DevOps Guru no está disponible para clústeres de base de datos de Aurora Serverless.

El siguiente vídeo contiene información general de DevOps Guru para RDS.

Para profundizar en el tema, consulte la publicación del blog de Amazon DevOps Guru para RDS entre bastidores.

Temas

- Beneficios de DevOps Guru para RDS
- Cómo funciona DevOps Guru for RDS
- Configuración de DevOps Guru for RDS

Beneficios de DevOps Guru para RDS

Si es responsable de una base de datos Amazon Aurora, es posible que no sepa que se está produciendo un evento o una regresión que está afectando a esa base de datos. Cuando aprenda sobre el problema, es posible que no sepa por qué está ocurriendo o qué hacer al respecto. En lugar de recurrir a un administrador de base de datos (DBA) para obtener ayuda o confiar en herramientas de terceros, puede seguir las recomendaciones de DevOps Guru para RDS.

Obtiene las siguientes ventajas del análisis detallado de DevOps Guru para RDS:

Diagnóstico rápido

DevOps Guru para RDS monitorea y analiza continuamente la telemetría de bases de datos. Información sobre rendimiento, Monitorización mejorada y Amazon CloudWatch recopilan datos de telemetría de su clúster de base de datos. DevOps Guru para RDS utiliza técnicas estadísticas y de machine learning para extraer estos datos y detectar anomalías. Para obtener más información sobre los datos de telemetría, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora y Supervisión de las métricas del sistema operativo con supervisión mejorada en la Guía del usuario de Amazon Aurora .

Resolución rápida

Cada anomalía identifica el problema del rendimiento y sugiere vías de investigación o medidas correctivas. Por ejemplo, DevOps Guru para RDS podría recomendar investigar eventos de espera específicos. O podría recomendarle que ajuste la configuración del grupo de aplicaciones para limitar el número de conexiones de base de datos. Según estas recomendaciones, puede resolver los problemas de rendimiento más rápido que mediante la solución de problemas de forma manual.

Información proactiva

DevOps Guru para RDS utiliza métricas de sus recursos para detectar posibles comportamientos problemáticos antes de que se conviertan en un problema mayor. Por ejemplo, puede detectar si la base de datos utiliza un número cada vez mayor de tablas temporales en el disco, lo que podría empezar a afectar al rendimiento. A continuación, DevOps Guru ofrece recomendaciones para ayudarle a solucionar los problemas antes de que se conviertan en problemas mayores.

Conocimiento profundo de los ingenieros de Amazon y machine learning

Para detectar problemas de rendimiento y ayudarle a resolver los cuellos de botella, DevOps Guru para RDS se basa en machine learning (ML) y fórmulas matemáticas avanzadas. Los ingenieros de bases de datos de Amazon contribuyeron al desarrollo de DevOps Guru para

los resultados de RDS, que encapsulan muchos años de administración de cientos de miles de bases de datos. Al aprovechar este conocimiento colectivo, DevOps Guru for RDS puede enseñarle las mejores prácticas.

Cómo funciona DevOps Guru for RDS

DevOps Guru para RDS recopila datos sobre sus bases de datos Aurora desde Información de rendimiento de Amazon RDS. La métrica más importante es `DBLoad`. DevOps Guru for RDS consume las métricas de Información sobre rendimiento, las analiza con machine learning y publica información en el panel de control.

La información es un conjunto de anomalías relacionadas que detecta DevOps Guru.

En DevOps Guru para RDS, una anomalía es un patrón que se desvía de lo que se considera un rendimiento normal para su base de datos Amazon Aurora.

Información proactiva

La información proactiva le permite conocer el comportamiento problemático antes de que se produzca. Contiene anomalías con recomendaciones y métricas relacionadas para ayudarlo a abordar los problemas en sus bases de datos Amazon Aurora antes de que se conviertan en problemas mayores. Esta información se publica en el panel de DevOps Guru.

Por ejemplo, DevOps Guru podría detectar que su base de datos Aurora PostgreSQL está creando muchas tablas temporales en el disco. Si no se soluciona este problema, esta tendencia podría provocar problemas de rendimiento. Cada información proactiva incluye recomendaciones sobre el comportamiento correctivo y enlaces a temas relevantes en Ajuste de Aurora MySQL con información proactiva de Amazon DevOps Guru o Ajuste de Aurora PostgreSQL con información proactiva de Amazon DevOps Guru. Para obtener más información, consulte Working with insights in DevOps Guru (Trabajo con información en DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.

Información reactiva

La información reactiva identifica el comportamiento anómalo a medida que se produce. Si DevOps Guru para RDS detecta problemas de rendimiento en las instancias de bases de datos de Amazon Aurora , publica información reactiva en el panel de DevOps Guru. Para obtener más información, consulte Working with insights in DevOps Guru (Trabajo con información en DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.

Anomalías causales

Una anomalía causal es una anomalía de nivel superior dentro de la información reactiva. Database load (DB load) (Carga de base de datos) es la anomalía causal de DevOps Guru for RDS.

Para medir el impacto del rendimiento, una anomalía asigna un nivel de gravedad de Alto, Mediano o Bajo. Para obtener más información, consulte Conceptos clave de DevOps Guru for RDS en la Guía del usuario de Amazon DevOps Guru.

Si DevOps Guru detecta una anomalía actual en la instancia de base de datos, se le avisará en la página Databases (Bases de datos) de la consola de RDS. La consola también le avisa de las anomalías que se han producido en las últimas 24 horas. Para ir a la página de anomalías desde la consola de RDS, elija el enlace del mensaje de alerta. La consola de RDS también le avisa en la página del clúster de base de datos de Amazon Aurora .

Anomalías contextuales

Una anomalía contextual es un resultado dentro de la carga de base de datos que está relacionada con una información reactiva. Cada anomalía contextual describe un problema de rendimiento específico de Amazon Aurora que requiere investigación. Por ejemplo, DevOps Guru for RDS podría recomendar que considere aumentar la capacidad de la CPU o investigar los eventos de espera que contribuyen a la carga de la base de datos.

> **Important**
>
> Recomendamos que pruebe cualquier cambio en una instancia de prueba antes de modificar una instancia de producción para que pueda entender completamente el impacto de cada cambio. De esta forma, comprende el impacto del cambio.

Para obtener más información, consulte Analyzing anomalies in Amazon Aurora clusters (Análisis de anomalías en clústeres de Amazon RDS) en la Guía del usuario de Amazon DevOps Guru.

Configuración de DevOps Guru for RDS

Para permitir que DevOps Guru para Amazon RDS publique información para una base de datos de Amazon Aurora , realice las siguientes tareas.

Temas

- Configuración de las políticas de acceso de IAM para DevOps Guru para RDS

- Activación de Información sobre rendimiento para sus instancias de base de datos de Aurora
- Activación de DevOps Guru y especificación de la cobertura de recursos

Configuración de las políticas de acceso de IAM para DevOps Guru para RDS

Para ver las alertas de DevOps Guru en la consola de RDS, su usuario o rol de AWS Identity and Access Management (IAM) debe contar con alguna de las siguientes políticas:

- La política administrada de AWS `AmazonDevOpsGuruConsoleFullAccess`
- La política administrada de AWS `AmazonDevOpsGuruConsoleReadOnlyAccess` y cualquiera de las siguientes políticas:
  - La política administrada por AWS `AmazonRDSFullAccess`
  - Una política administrada por el cliente que incluya `pi:GetResourceMetrics` y `pi:DescribeDimensionKeys`

Para obtener más información, consulte Configuración de directivas de acceso para información sobre rendimiento.

Activación de Información sobre rendimiento para sus instancias de base de datos de Aurora

DevOps Guru for RDS confía en Información sobre rendimiento para sus datos. Sin Información sobre rendimiento, DevOps Guru publica anomalías, pero no incluye análisis ni recomendaciones detallados.

Al crear un clúster de base de datos de Aurora o modificar una instancia de clúster, puede activar Performance Insights. Para obtener más información, consulte Activación y desactivación de Información de rendimiento de Aurora.

Activación de DevOps Guru y especificación de la cobertura de recursos

Puede activar DevOps Guru para que supervise sus bases de datos de Amazon Aurora de cualquiera de las siguientes maneras.

Temas

- Activación de DevOps Guru en la consola de RDS
- Adición de recursos de Aurora en la consola de DevOps Guru

- Adición de recursos de Aurora mediante CloudFormation

Activación de DevOps Guru en la consola de RDS

Puede seguir varias rutas en la consola de Amazon RDS para activar DevOps Guru.

Temas

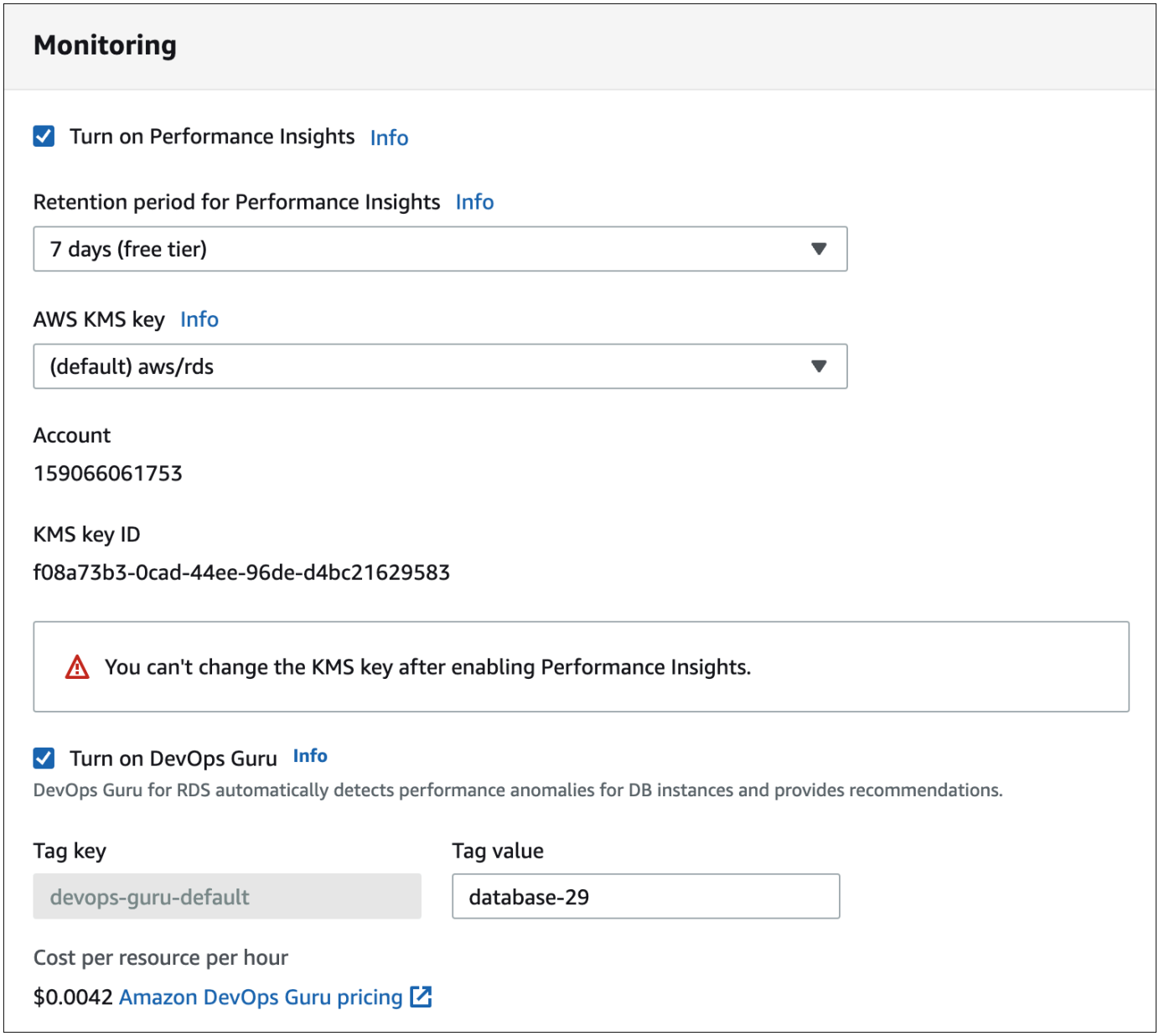
- Activación de DevOps Guru cuando crea una base de datos Aurora
- Activación de DevOps Guru desde el banner de notificación
- Respuesta a un error de permisos al activar DevOps Guru

Activación de DevOps Guru cuando crea una base de datos Aurora

El flujo de trabajo de creación incluye una configuración que activa la cobertura de DevOps Guru para su base de datos. Esta configuración se activa de forma predeterminada cuando elige la plantilla Production (Producción).

Para activar DevOps Guru cuando crea una base de datos Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Siga los pasos de Creación de un clúster de base de datos, hasta el paso en el que elige la configuración de supervisión pero sin incluirlo.
3. En Monitoring (Supervisión), elija Turn on Performance Insights (Activar Performance Insights). Para que DevOps Guru para RDS proporcione un análisis detallado de las anomalías de rendimiento, es necesario activar Performance Insights.
4. Elija Turn on DevOps Guru (Activar DevOps Guru).



5. Cree una etiqueta para la base de datos para que DevOps Guru pueda supervisarla. Haga lo siguiente:

   - En el campo de texto de Tag key (Clave de etiqueta), ingrese un nombre que comience por **`Devops-Guru-`**.
   - En el campo de texto de Tag value (Valor de etiqueta), ingrese cualquier valor. Por ejemplo, si especifica **`rds-database-1`** para el nombre de la base de datos Aurora, también puede introducir **`rds-database-1`** como el valor de etiqueta.

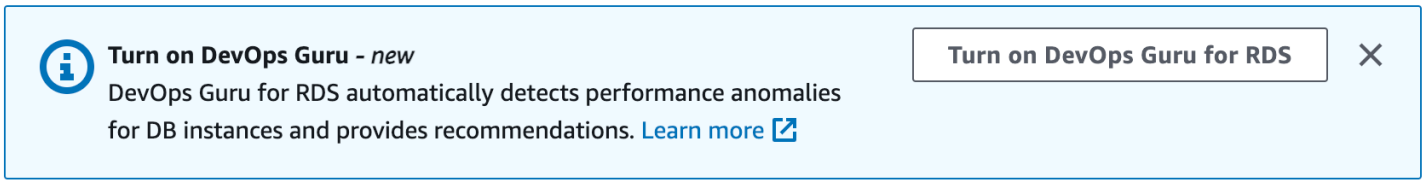
Para obtener más información sobre las etiquetas, consulte "Use tags to identify resources in your DevOps Guru applications" (Usar etiquetas para identificar los recursos en las aplicaciones de DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.

6. Complete los demás pasos proporcionados en Creación de un clúster de base de datos.

Activación de DevOps Guru desde el banner de notificación

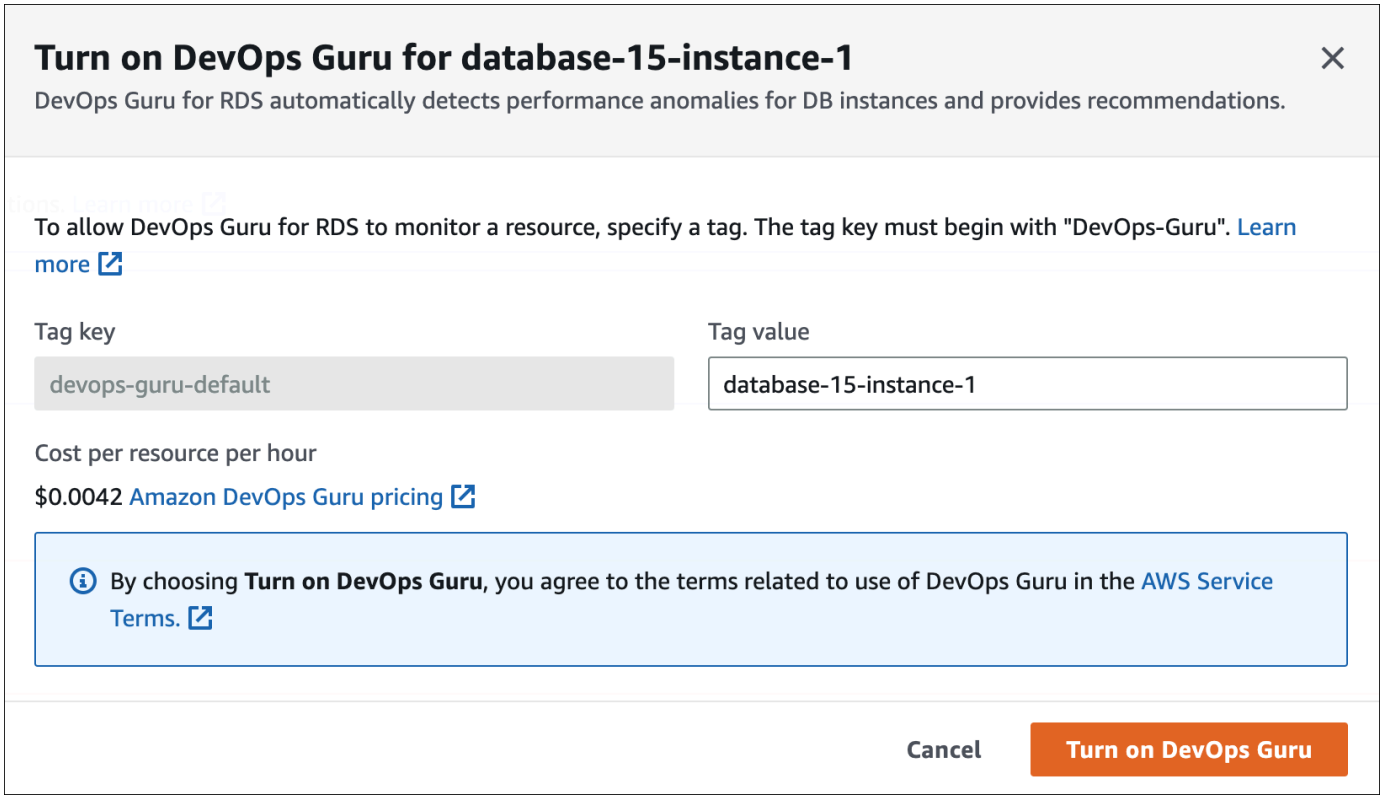
Si sus recursos no están cubiertos por DevOps Guru, Amazon RDS se lo notifica con un banner en las siguientes ubicaciones:

- La pestaña Monitoring (Supervisión) de una instancia de clúster de base de datos
- Panel de Performance Insights



Para activar DevOps Guru para su base de datos Aurora

1. En el banner, elija Turn on DevOps Guru for RDS (Activar DevOps Guru para RDS).
2. Ingrese un nombre y un valor de la clave de la etiqueta. Para obtener más información sobre las etiquetas, consulte "Use tags to identify resources in your DevOps Guru applications" (Usar etiquetas para identificar los recursos en las aplicaciones de DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.



3. Elija Turn on DevOps Guru (Activar DevOps Guru).

Respuesta a un error de permisos al activar DevOps Guru

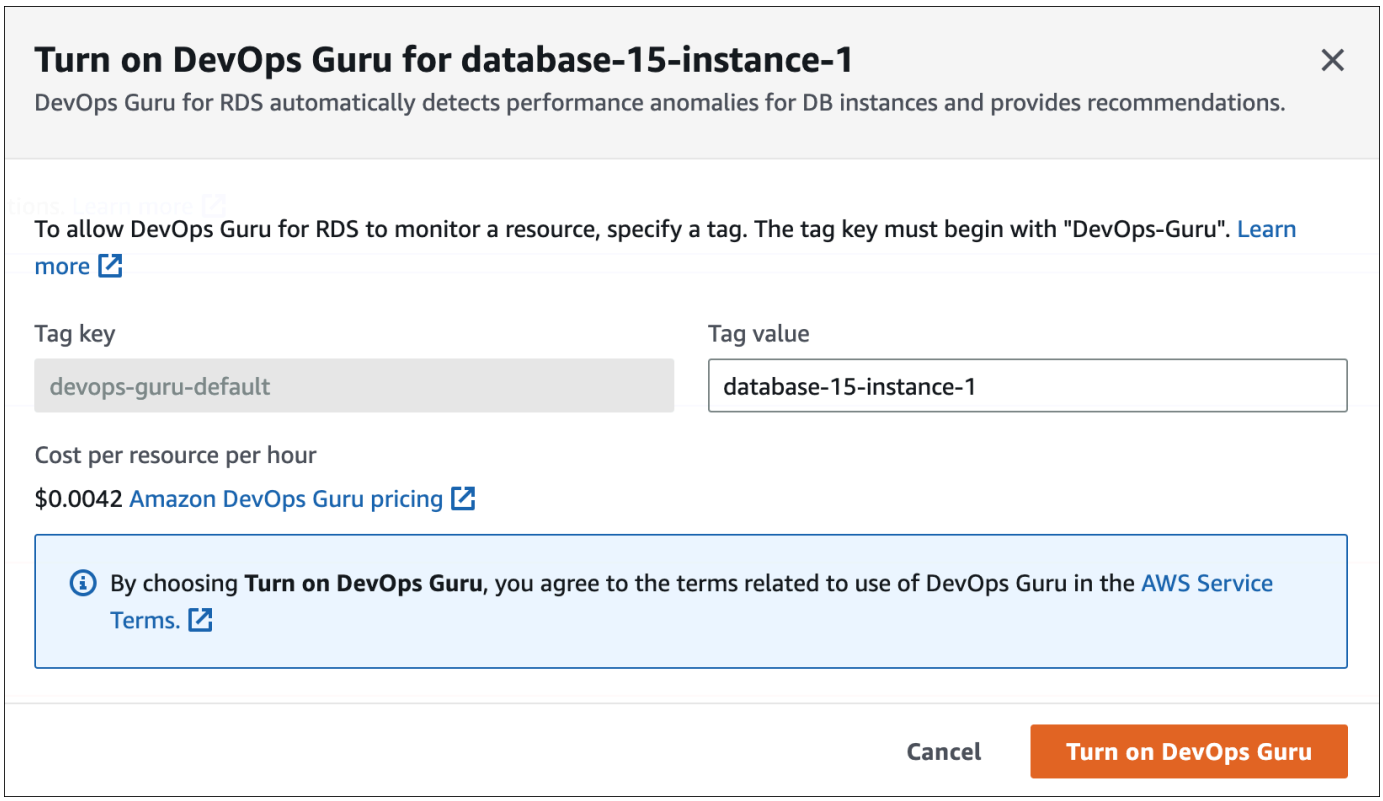
Si activa DevOps Guru desde la consola de RDS al crear una base de datos, es posible que RDS muestre el siguiente banner acerca de la ausencia de permisos.

Para responder a un error de permisos

1. Conceda a su usuario o rol de IAM el rol administrado por el usuario `AmazonDevOpsGuruConsoleFullAccess`. Para obtener más información, consulte Configuración de las políticas de acceso de IAM para DevOps Guru para RDS.
2. Abra la consola de RDS.
3. En el panel de navegación, seleccione Información de rendimiento.
4. Elija una instancia de base de datos en el clúster que acaba de crear.
5. Elija el conmutador para activar DevOps Guru para RDS.

DevOps Guru for RDS

6. Elija un valor de etiqueta. Para obtener más información, consulte "Use tags to identify resources in your DevOps Guru applications" (Usar etiquetas para identificar los recursos en las aplicaciones de DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.



7. Elija Turn on DevOps Guru (Activar DevOps Guru).

Adición de recursos de Aurora en la consola de DevOps Guru

Puede especificar su cobertura de recursos de DevOps Guru en la consola de DevOps Guru. Siga el paso descrito en Specify your DevOps Guru resource coverage (Especifique su cobertura de recursos DevOps Guru) en la Guía del usuario de Amazon DevOps Guru. Cuando edite los recursos analizados, elija una de las siguientes opciones:

- Elija Todos los recursos de la cuenta para analizar todos los recursos admitidos, como las bases de datos Aurora, en su Cuenta de AWS y región.
- Elija Pilas de CloudFormation para analizar las bases de datos Aurora que se encuentran en las pilas que usted elija. Para obtener más información, consulte el tema sobre usar pilas de AWS CloudFormation para identificar los recursos en sus aplicaciones de DevOps Guru en la Guía del usuario de Amazon DevOps Guru.

- Elija Etiquetas para analizar las bases de datos Aurora que ha etiquetado. Para obtener más información, consulte Use tags to identify resources in your DevOps Guru applications (Usar etiquetas para identificar los recursos en las aplicaciones de DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.

Para obtener más información, consulte Enable DevOps Guru (Habilitar DevOps Guru) en la Guía del usuario de Amazon DevOps Guru.

Adición de recursos de Aurora mediante CloudFormation

Puede utilizar etiquetas para añadir cobertura de sus recursos de Aurora a sus plantillas de CloudFormation. En el siguiente procedimiento, se asume que tiene una plantilla de CloudFormation tanto para la instancia de base de datos de Aurora como para la pila de DevOps Guru.

Para especificar una instancia de base de datos de Auroramediante una etiqueta de CloudFormation

1. En la plantilla de CloudFormation para su instancia de base de datos, defina una etiqueta mediante un par clave/valor.

   En el siguiente ejemplo, se asigna el valor `my-aurora-db-instance1` a `Devops-guru-cfn-default` para una instancia de base de datos de Aurora.

   ```
   MyAuroraDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBClusterIdentifier: my-aurora-db-cluster
       DBInstanceIdentifier: my-aurora-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-aurora-db-instance1
   ```

2. En la plantilla de CloudFormation de su pila de DevOps Guru, especifique la misma etiqueta en el filtro de recopilación de recursos.

   En el siguiente ejemplo, se configura DevOps Guru para proporcionar cobertura para el recurso con el valor de etiqueta `my-aurora-db-instance1`.

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
   ```

```
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "devopsguru-my-aurora-db-instance1"
```

En el siguiente ejemplo, se proporciona cobertura para todos los recursos dentro del límite de la aplicación `Devops-guru-cfn-default`.

```
DevOpsGuruResourceCollection:
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

Para obtener más información, consulte AWS::DevOpsGuru::ResourceCollection y AWS::RDS::DBInstance en la Guía del usuario de CloudFormation.

Supervisión de las métricas del sistema operativo con Supervisión mejorada

Con la supervisión mejorada, puede monitorear el sistema operativo de su instancia de base de datos en tiempo real. Cuando desea ver cómo diferentes procesos o subprocesos usan la CPU, las métricas de monitorización mejoradas son útiles.

Temas

- Descripción general de la supervisión mejorada
- Configuración y habilitación del monitoreo mejorado
- Visualización de métricas OS en la consola de RDS
- Visualización de métricas del sistema operativo mediante CloudWatch Logs

Descripción general de la supervisión mejorada

Amazon RDS proporciona métricas en tiempo real para el sistema operativo (SO) en el que se ejecuta la instancia de base de datos. Puede ver todas las métricas del sistema y la información de procesos de las instancias de base de datos de RDS en la consola. Puede administrar las métricas que desea monitorear para cada instancia y personalizar el panel de acuerdo con sus requisitos. Para ver descripciones de métricas de la supervisión mejorada, consulte Métricas del sistema operativo en Supervisión mejorada.

RDS entrega las métricas de la monitorización mejorada a su cuenta de registros de Amazon Cloudwatch. Puede crear filtros de métricas en CloudWatch desde CloudWatch Logs y mostrar los gráficos en el panel de CloudWatch. Además, puede consumir la salida JSON de monitorización mejorada desde registros de Amazon Cloudwatch en un sistema de monitoreo de su elección. Para obtener más información, consulte Monitorización mejorada en las preguntas frecuentes de Amazon RDS.

Temas

- Diferencias entre métricas de monitoreo mejorado y CloudWatch
- Retención de métricas de supervisión mejorada
- Costo de la monitorización mejorada

Diferencias entre métricas de monitoreo mejorado y CloudWatch

Un hipervisor crea y ejecuta máquinas virtuales (VM). Mediante el uso de un hipervisor, una instancia puede admitir varias máquinas virtuales invitadas al compartir virtualmente memoria y CPU. CloudWatch recopila métricas sobre el uso de la CPU del hipervisor para una instancia de base de datos. Por el contrario, la supervisión mejorada recopila sus métricas de un agente en la instancia de base de datos.

Podría encontrar diferencias entre CloudWatch y las medidas de supervisión mejorada, porque la capa del hipervisor realiza una pequeña cantidad de trabajo. Las diferencias pueden ser mayores si sus instancias de base de datos utilizan clases de instancia más pequeñas. En este escenario, es probable que la capa de hipervisor administre más máquinas virtuales (VM) en una única instancia física.

Para ver descripciones de métricas de la supervisión mejorada, consulte Métricas del sistema operativo en Supervisión mejorada. Para obtener más información sobre las métricas de CloudWatch, consulte la Guía del usuario de Amazon CloudWatch.

Retención de métricas de supervisión mejorada

Las métricas de supervisión mejorada se almacenan de forma predeterminada en CloudWatch Logs durante 30 días. Este periodo de retención es diferente de las métricas típicas de CloudWatch.

Para modificar la cantidad de tiempo que se almacenan las métricas en CloudWatch Logs, cambie la retención del grupo de registros `RDSOSMetrics` en la consola de CloudWatch. Para obtener más información, consulte Cambiar la retención de datos de registro en CloudWatch Logs en la registros de Amazon Cloudwatch User Guide.

Costo de la monitorización mejorada

Las métricas de supervisión mejorada se almacenan en CloudWatch Logs en lugar de en las métricas de Cloudwatch. El costo de la monitorización mejorada depende de los factores siguientes:

- Se le cobrará por la monitorización mejorada solo si supera el nivel gratuito que proporciona registros de Amazon Cloudwatch. Los cargos se basan en las tasas de transferencia de datos y almacenamiento de CloudWatch Logs.
- La cantidad de información transferida para una instancia de RDS es directamente proporcional a la granularidad definida para la función de monitorización mejorada. Un intervalo de monitorización más corto deriva en informes más frecuentes de métricas del SO y aumenta el costo de la

monitorización. Para administrar los costos, establezca diferentes granularidades para diferentes instancias en sus cuentas.

- Los costos de uso de la monitorización mejorada se aplican en cada instancia de base de datos para la que está habilitada dicha monitorización. Monitorizar un gran números de instancias de base de datos es más costoso que monitorizar tan solo unas cuantas.
- Las instancias de base de datos que admiten una carga de trabajo con computación más intensiva tienen más actividad de proceso de SO de la que informar y costos más elevados de monitorización mejorada.

Para obtener más información acerca de los precios, consulte Precios de Amazon CloudWatch.

Configuración y habilitación del monitoreo mejorado

Para utilizar el Monitoreo mejorado, debe crear un rol de IAM y, a continuación, habilitar el Monitoreo mejorado.

Temas

- Creación de un rol de IAM para el monitoreo mejorado
- Activación y desactivación de la supervisión mejorada
- Protección contra el problema del suplente confuso

Creación de un rol de IAM para el monitoreo mejorado

La monitorización mejorada necesita permiso para actuar en su nombre y enviar información de métrica del SO a CloudWatch Logs. Puede conceder los permisos necesarios para Enhanced Monitoring mediante un rol de AWS Identity and Access Management (IAM). Puede crear este rol al habilitar la supervisión mejorada o crearla con anticipación.

Temas

- Creación del rol de IAM cuando habilita el monitoreo mejorado
- Creación del rol de IAM antes de habilitar el monitoreo mejorado

Creación del rol de IAM cuando habilita el monitoreo mejorado

Cuando habilita el monitoreo mejorado en la consola de RDS, con Amazon RDS se puede crear el rol de IAM que usted necesite. El rol se denomina `rds-monitoring-role`. RDS utiliza este rol para la instancia de base de datos, la réplica de lectura o el clúster de base de datos Multi-AZ especificados.

Cómo crear el rol de IAM cuando se habilita el monitoreo mejorado

1. Siga los pasos de Activación y desactivación de la supervisión mejorada.
2. Establezca el Monitoring Role (Rol de monitoreo) en Default (Predeterminado) en el paso en el que elija un rol.

Creación del rol de IAM antes de habilitar el monitoreo mejorado

Puede crear el rol necesario antes de habilitar el monitoreo mejorado. Cuando habilite el monitoreo mejorado, especifique el nombre del rol nuevo. Debe crear este rol necesario si habilita la monitorización mejorada mediante la AWS CLI o la API de RDS.

Se debe conceder al usuario que habilite la monitorización mejorada el permiso `PassRole`. A fin de obtener más información, consulte el Ejemplo 2 en Concesión de permisos a un usuario para transferir un rol a un servicio de AWS en la Guía del usuario de IAM.

Para crear un rol de IAM para el monitoreo mejorado de Amazon RDS

1. Abra la consola de IAM en https://console.aws.amazon.com.
2. Seleccione Roles en el panel de navegación.
3. Elija Create role.
4. Elija la pestaña Servicio de AWS y, a continuación, elija RDS de la lista de servicios.
5. Elija RDS - Enhanced Monitoring (RDS - Supervisión mejorada) y, a continuación, elija Next (Siguiente).
6. Asegúrese de que en Permissions policies (Políticas de permisos) se muestra AmazonRDSEnhancedMonitoringRole y, a continuación, elija Next (Siguiente).
7. Escriba un nombre para el rol en Nombre de rol. Por ejemplo, escriba **`emaccess`**.

   La entidad de confianza para su rol es el servicio de AWS monitoring.rds.amazonaws.com.
8. Elija Create role (Crear rol).

Activación y desactivación de la supervisión mejorada

Puede administrar Supervisión mejorada mediante la Consola de administración de AWS, la AWS CLI o la API de RDS. Puede establecer diferentes niveles de detalle para la recopilación de métricas en cada clúster de base de datos. También puede activar Supervisión mejorada para un clúster de base de datos existente desde la consola.

Consola

Puede activar la Supervisión mejorada cuando crea un clúster de base de datos o una réplica de lectura, o cuando modifica un clúster de base de datos. Si modifica una instancia o clúster de base de datos para activar Supervisión mejorada, no necesita reiniciar la instancia de base de datos para que se efectúe el cambio.

Puede activar la Supervisión mejorada en la consola de RDS cuando realiza una de las siguientes acciones en la página Databases (Bases de datos).

- Crear un clúster de base de datos: elija Create database (Crear base de datos).
- Crear una réplica de lectura: elija Actions (Acciones) y, luego, Create read replica (Crear una réplica de lectura).
- Modificación de una instancia de base de datos o clúster de base de datos : elija la opción Modificar.

> **Note**
>
> Al habilitar Supervisión mejorada para un clúster de base de datos, no podrá administrarla para instancias de base de datos individuales dentro del clúster.

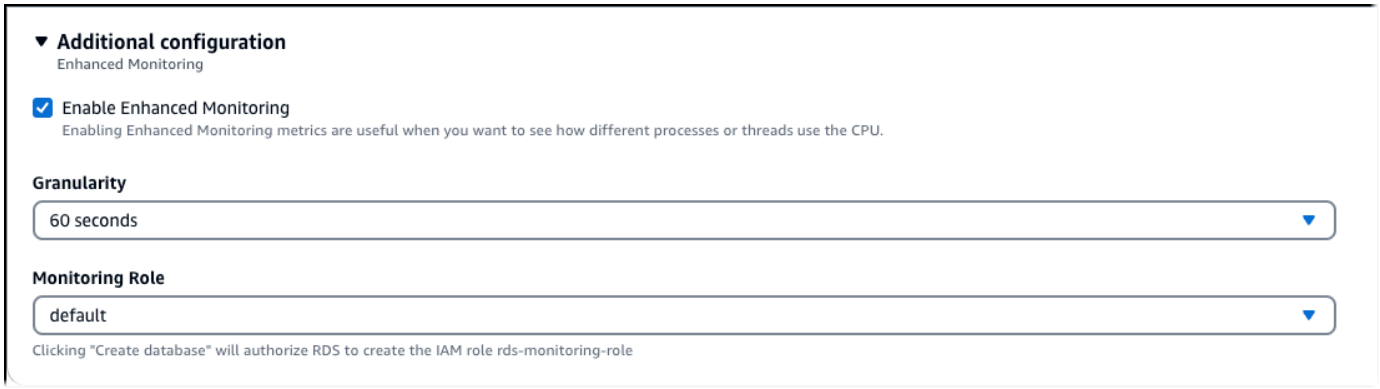
Si administra Supervisión mejorada para instancias de base de datos individuales en un clúster de base de datos y el nivel de detalle se establece en valores diferentes para distintas instancias, el clúster de base de datos será heterogéneo con respecto a Supervisión mejorada. En estos casos, no puede modificar el clúster de base de datos para administrar Supervisión mejorada en el clúster.

Para activar o desactivar la supervisión mejorada en la consola de RDS

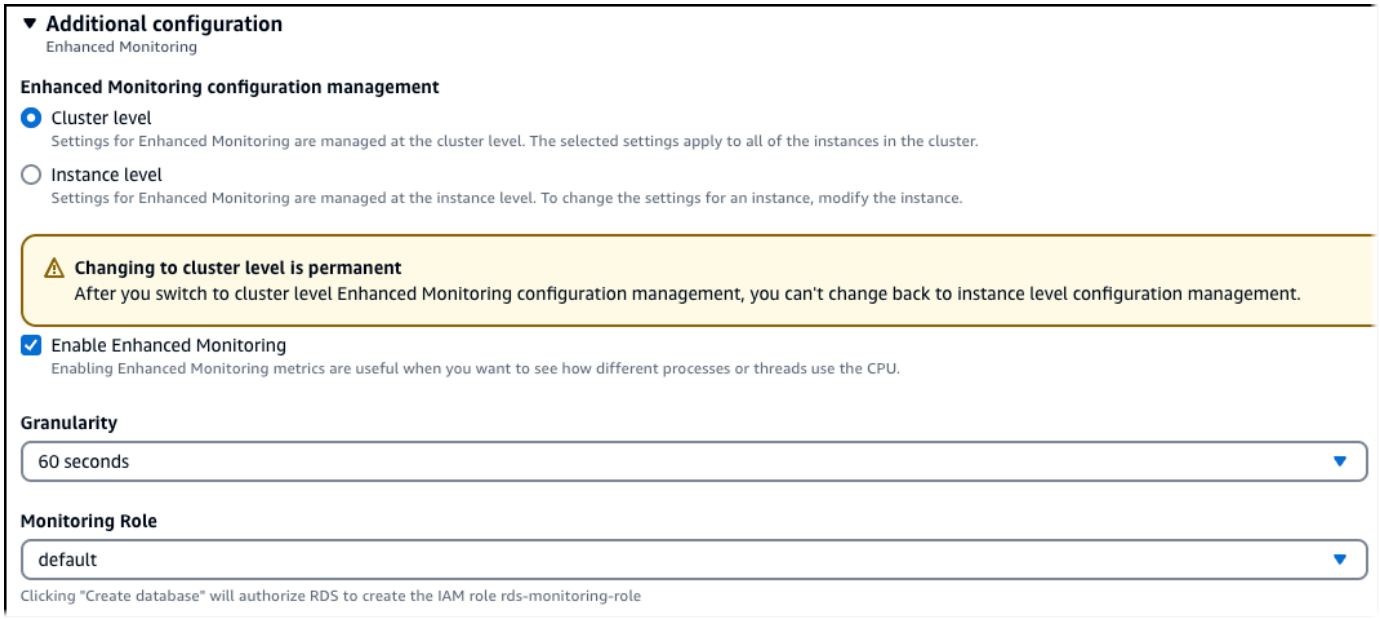
1. Desplácese a Additional configuration (Configuración adicional).

2. En Supervisión, elija Habilitar la monitorización mejorada para la clúster de base de datos o réplica de lectura. Al habilitar Supervisión mejorada en el clúster, puede administrar la configuración y las opciones de Supervisión mejorada en el clúster. La configuración del clúster se aplica a todas las instancias de base de datos del clúster.Anule la selección de la opción para deshabilitar Supervisión mejoradaen el clúster. Más adelante, podrá modificar la configuración de Supervisión mejorada para las instancias de base de datos individuales del clúster.

   En la página Crear base de datos, puede seleccionar la opción de activar Supervisión mejorada en el clúster.



   Si no habilita Supervisión mejorada al crear un clúster, puede modificarlo en la página Modificar clúster de base de datos.



> **Note**
>
> No puede administrar Supervisión mejorada para una instancia de base de datos individual en un clúster de base de datos que ya tenga esta opción administrada en el clúster.

3. No tiene la opción de elegir administrar Supervisión mejorada en el clúster si el clúster es heterogéneo con respecto a Supervisión mejorada. Para administrar Supervisión mejorada en el clúster, cambie la configuración de Supervisión mejorada de cada instancia para que sea igual en todas. Ahora puede optar por administrar Supervisión mejorada en el clúster al modificarlo.
4. Establezca la propiedad Monitoring Role (Rol de monitorización) en el rol de IAM que ha creado para permitir que Amazon RDS se comunique con registros de Amazon Cloudwatch, o bien elija Default (Predeterminado) para que RDS cree un rol denominado `rds-monitoring-role`.
5. Establezca la propiedad Nivel de detalle en el intervalo (en segundos) entre puntos cuando se recopilan métricas para la instancia de base de datos, clúster de base datos o réplica de lectura. La propiedad Granularity puede establecerse en uno de los siguientes valores: `1`, `5`, `10`, `15`, `30` o `60`.

   La velocidad más rápida a la que la consola de RDS se actualiza es cada 5 segundos. Si establece la granularidad en 1 segundo en la consola de RDS, seguirá viendo métricas actualizadas solo cada 5 segundos. Puede recuperar actualizaciones de métricas de 1 segundo mediante CloudWatch Logs.

AWS CLI

Para activar la Supervisión mejorada mediante la AWS CLI, en los siguientes comandos, establezca la opción `--monitoring-interval` en un valor distinto de `0` y establezca la opción `--monitoring-role-arn` en el rol que creó en Creación de un rol de IAM para el monitoreo mejorado.

- create-db-cluster
- modify-db-cluster
- create-db-instance
- create-db-instance-read-replica
- modify-db-instance

La opción `--monitoring-interval` especifica el intervalo (en segundos) entre puntos cuando se recopilan métricas de monitoreo mejorado. Los valores válidos para la opción son `0`, `1`, `5`, `10`, `15`, `30` y `60`.

Para desactivar la supervisión mejorada mediante AWS CLI, establezca la opción `--monitoring-interval` en `0` en los siguientes comandos.

Example

El siguiente ejemplo activa la Supervisión mejorada para una instancia de base de datos:

Para Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

En el siguiente ejemplo, se activa Supervisión mejorada para un clúster de base de datos:

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --monitoring-interval 30 ^
```

```
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

El siguiente ejemplo activa la Supervisión mejorada para una instancia de base de datos Multi-AZ:

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

API de RDS

Para activar la supervisión mejorada mediante la API de RDS, establezca el parámetro `MonitoringInterval` en un valor distinto de `0` y establezca el parámetro `MonitoringRoleArn` en el rol que creó en Creación de un rol de IAM para el monitoreo mejorado. Establezca estos parámetros en las siguientes acciones:

- CreateDBCluster
- ModifyDBCluster
- CreateDBInstance
- CreateDBInstanceReadReplica
- ModifyDBInstance

El parámetro `MonitoringInterval` especifica el intervalo (en segundos) entre puntos cuando se recopilan métricas de monitoreo mejorado. Los valores válidos son `0`, `1`, `5`, `10`, `15`, `30` y `60`.

Para desactivar la supervisión mejorada mediante la API de RDS, establezca `MonitoringInterval` en `0`.

Protección contra el problema del suplente confuso

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación entre servicios puede dar lugar al problema de la sustitución confusa. La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puedes manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta. Para obtener más información, consulte El problema del suplente confuso.

Para limitar los permisos al recurso que Amazon RDS puede dar a otro servicio, se recomienda utilizar las claves de contexto de condición global de `aws:SourceArn` y `aws:SourceAccount` en una política de confianza para el rol de supervisión mejorada. Si utiliza ambas claves de contexto de condición global, deben usar el mismo ID de cuenta.

La forma más eficaz de protegerse contra el problema del suplente confuso es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Para Amazon RDS, establezca `aws:SourceArn` en `arn:aws:rds:`*`Region`*`:`*`my-account-id`*`:db:`*`dbname`*.

En los siguientes ejemplos se utilizan las claves de contexto de condición global de `aws:SourceArn` y `aws:SourceAccount` en una política de confianza para evitar el problema del suplente confuso.

JSON

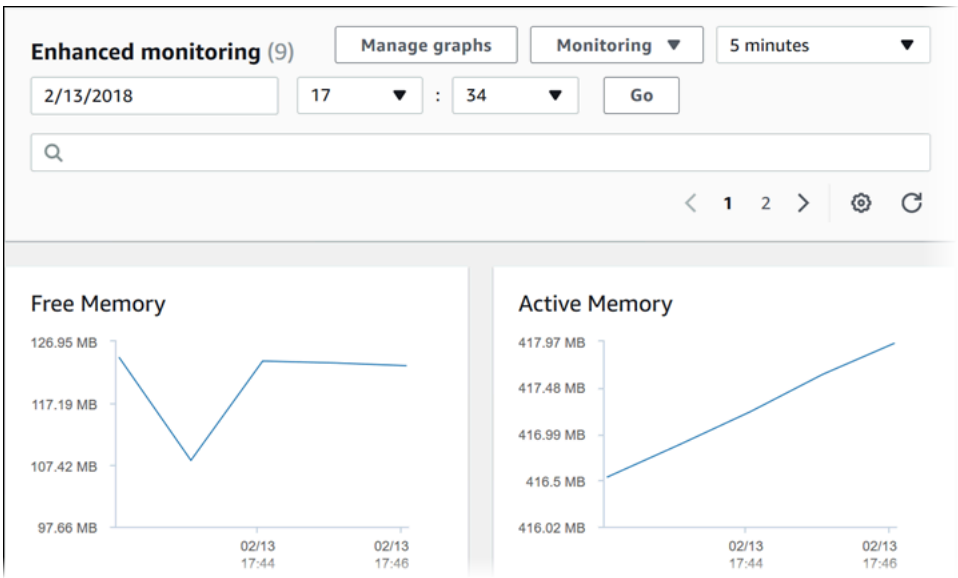
```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-account-id:db:dbname"
        },
```

```
        "StringEquals": {
          "aws:SourceAccount": "my-account-id"
        }
      }
    }
  ]
}
```

Visualización de métricas OS en la consola de RDS

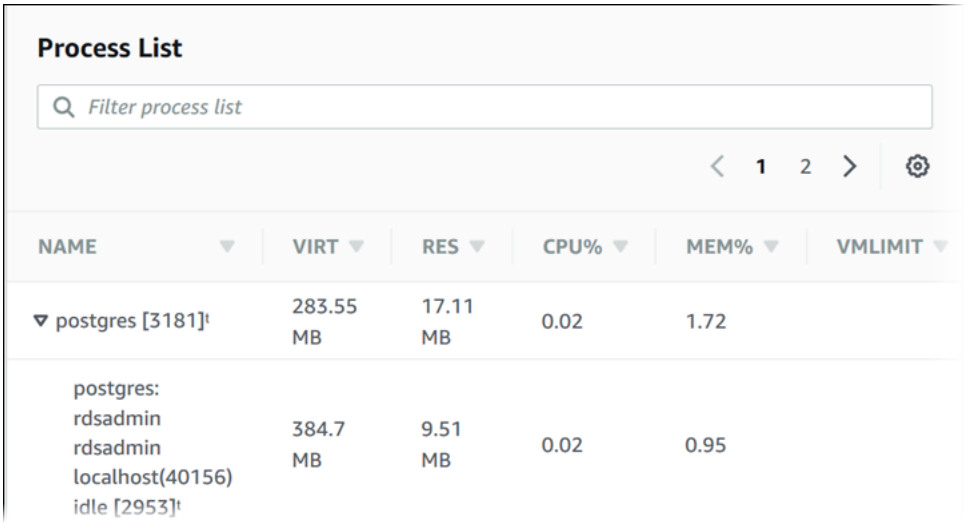
Puede ver las métricas del SO informadas por la monitorización mejorada en la consola de RDS si elige Enhanced monitoring (Monitorización mejorada) para Monitoring (Monitorización).

El siguiente ejemplo muestra la página Supervisión mejorada. Para ver descripciones de métricas de la supervisión mejorada, consulte Métricas del sistema operativo en Supervisión mejorada.



Si desea ver detalles para los procesos que se ejecutan en su instancia de base de datos, elija OS process list (Lista de procesos del SO) para Monitoring (Monitorización).

A continuación, se muestra la vista Process List (Lista de procesos).



La métrica de monitorización mejorada que se muestra en la vista Process list (Lista de procesos) se organiza de la siguiente manera:

- RDS child processes (Procesos secundarios de RDS): muestra un resumen de los procesos de RDS que admiten la instancia de base de datos, por ejemplo `aurora` para clústeres de base de datos de Amazon Aurora. Los subprocesos aparecen anidados debajo del proceso principal. Los subprocesos solo muestran el uso de la CPU, ya que otras métricas son iguales para todos los subprocesos. La consola muestra un máximo de 100 procesos y subprocesos. Los resultados son una combinación de los principales procesos y subprocesos que consumen memoria y CPU. Si hay más de 50 procesos y más de 50 subprocesos, la consola muestra los 50 consumidores principales de cada categoría. Esta pantalla le ayuda a identificar qué procesos están teniendo mayor impacto en el desempeño.
- Procesos de RDS: muestra un resumen de los recursos utilizados por el agente de administración de RDS, los procesos de monitoreo de diagnóstico y otros procesos de AWS que son necesarios para admitir instancias de base de datos de RDS.
- OS processes (Procesos de SO)–: muestra un resumen del kernel y de los procesos del sistema, que por lo general tienen un impacto mínimo en el rendimiento.

Los elementos enumerados para cada proceso son los siguientes:

- VIRT–: muestra el tamaño virtual del proceso.
- RES–: muestra la memoria física real que utiliza el proceso.
- CPU% (% de CPU): muestra el porcentaje de ancho de banda de CPU total que consume el proceso.

- MEM% (% de memoria): muestra el porcentaje de memoria total que utiliza el proceso.

Los datos de monitorización que se muestran en la consola de RDS se obtienen de registros de Amazon Cloudwatch. También puede obtener la métrica para una instancia de base de datos en forma de flujo de registro de CloudWatch Logs. Para obtener más información, consulte Visualización de métricas del sistema operativo mediante CloudWatch Logs.

La métrica de monitorización mejorada no se devuelve durante las siguientes situaciones:

- Una conmutación por error de la instancia de base de datos.
- Cambio de la clase de una instancia de base de datos (escalado del cómputo).

La métrica de monitorización mejorada se devuelve al reiniciar una instancia de base de datos porque solo se reinicia el motor de la base de datos. Se sigue informando de la métrica correspondiente al sistema operativo.

Visualización de métricas del sistema operativo mediante CloudWatch Logs

Una vez habilitada la Supervisión mejorada para la clúster, puede ver las métricas mediante CloudWatch Logs, con cada flujo de registro que representa una sola instancia de base de datos o un clúster de base de datos bajo supervisión. El identificador de flujo de registros es el identificador de recursos (`DbiResourceId`) para la instancia de base de datos o el clúster de base de datos.

Para ver los datos de registro de la Supervisión mejorada

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Si es necesario, elija la Región de AWS en la que se encuentra su clúster de base de datos. Para obtener más información, consulte Regiones y puntos de enlace en la Referencia general de Amazon Web Services.
3. En el panel de navegación, elija Logs.
4. Elija RDSOSMetrics en la lista de grupos de registro.
5. Elija el flujo de registro que desea ver en la lista de flujos de registro.

Referencia de métricas para Amazon Aurora

En esta referencia, encontrará descripciones de las métricas de Amazon Aurora para Amazon CloudWatch, Información sobre rendimiento y Supervisión mejorada.

Temas

- Métricas de Amazon CloudWatch para Amazon Aurora
- Dimensiones de Amazon CloudWatch para Aurora.
- Disponibilidad de métricas de Aurora en la consola de Amazon RDS.
- Métricas de Amazon CloudWatch para Información de rendimiento de Amazon RDS
- Métricas de contador de Información de rendimiento
- Estadísticas de SQL para Performance Insights
- Métricas del sistema operativo en Supervisión mejorada

Métricas de Amazon CloudWatch para Amazon Aurora

El espacio de nombres `AWS/RDS` incluye las siguientes métricas que se aplican a entidades de base de datos que se ejecutan en Amazon Aurora. Algunas métricas se aplican a Aurora MySQL, Aurora PostgreSQL o ambos. Además, algunas métricas son específicas de un clúster, instancia principal, instancia de réplica o todas las instancias de base de datos.

Para consultar las métricas de base de datos globales de Aurora, consulte Métricas de Amazon CloudWatch para el reenvío de escritura en Aurora MySQL y Métricas de Amazon CloudWatch para el reenvío de escritura en Aurora PostgreSQL. Para obtener métricas de consulta paralelas de Aurora, consulte Monitorización de consultas paralelas para Aurora MySQL.

Temas

- Métricas de nivel de clúster para Amazon Aurora
- Métricas de nivel de instancia para Amazon Aurora
- Métricas de uso de Amazon CloudWatch para Amazon Aurora

Métricas de nivel de clúster para Amazon Aurora

En la siguiente tabla se describen las métricas específicas de los clústeres de Aurora.

Métrica	Descripción	Se aplica a	Unidades
`AuroraGlobalDBDataTransferBytes`	En una base de datos global de Aurora, la cantidad de datos de registros REDO transferida desde la región AWS de origen a una región de AWS secundaria. **Note** Esta métrica solo está disponible en una secundaria Regiones de AWS.	Aurora MySQL y Aurora PostgreSQL	Bytes
`AuroraGlobalDBProgressLag`	En una base de datos global de Aurora, la medida de hasta qué punto el clúster secundario está detrás del clúster principal tanto para las transacciones de usuario como para las transacciones del sistema. **Note** Esta métrica solo está disponible en una secundaria Regiones de AWS.	Aurora MySQL y Aurora PostgreSQL	Milisegundos
`AuroraGlobalDBReplicatedWriteIO`	En una base de datos global de Aurora, el número de operaciones de E/S de escritura replicadas desde la	Aurora MySQL y Aurora PostgreSQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
	región de AWS principal al volumen de clúster en una región de AWS secundaria. Los cálculos de facturación para las regiones de AWS secundarias en una base de datos global utilizan `VolumeWriteIOPs` para tener en cuenta las escrituras realizadas dentro del clúster. Los cálculos de facturación para la región de AWS principal en una base de datos global utilizan `VolumeWriteIOPs` para tener en cuenta la actividad de escritura dentro de dicho clúster y `AuroraGlobalDBReplicatedWriteIO` para tener en cuenta la replicación entre regiones dentro de la base de datos global. **Note** Esta métrica solo está disponible en una secundaria Regiones de AWS.		

Métrica	Descripción	Se aplica a	Unidades
`AuroraGlobalDBReplicationLag`	Para una base de datos global de Aurora, el retardo que se da cuando la replicación actualiza desde la región de AWS principal. **Note** Esta métrica solo está disponible en una secundaria Regiones de AWS.	Aurora MySQL y Aurora PostgreSQL	Milisegundos
`AuroraGlobalDBRPOLag`	En una base de datos global de Aurora, el retardo del objetivo del punto de recuperación (RPO). Esta métrica mide hasta qué punto está el clúster secundario detrás del clúster principal para las transacciones de usuario. **Note** Esta métrica solo está disponible en una secundaria Regiones de AWS.	Aurora MySQL y Aurora PostgreSQL	Milisegundos

Métrica	Descripción	Se aplica a	Unidades
`AuroraVolumeBytesLeftTotal`	El espacio disponible restante para el volumen del clúster. A medida que crece el volumen del clúster, este valor disminuye. Si llega a cero, el clúster notifica un error de falta de espacio. Si desea detectar si su clúster de Aurora MySQL se acerca al límite de tamaño de 128 tebibytes (TiB), este valor es más simple y de más confianza para monitorear que `VolumeBytesUsed` . `AuroraVolumeBytesLeftTotal` tiene en cuenta el almacenamiento utilizado para la organización interna y otras asignaciones que no afectan a la facturación del almacenamiento.	Aurora MySQL	Bytes
`BacktrackChangeRecordsCreationRate`	El número de registros de cambios en el seguimiento de datos anteriores que se crean a lo largo de cinco minutos en su clúster de base de datos.	Aurora MySQL	Recuento cada 5 minutos
`BacktrackChangeRecordsStored`	El número de registros de cambios de datos anteriores que se han utilizado en el clúster de base de datos.	Aurora MySQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
`BackupRetentionPeriodStorageUsed`	La cantidad total de almacenamiento de copias de seguridad utilizada para permitir la característica de restauración a un momento dado dentro del periodo de retención de copias de seguridad del clúster de base de datos de Aurora. Esta cantidad se incluye en el total notificado por la métrica `TotalBackupStorageBilled` . Se calcula de forma independiente para cada clúster de Aurora. Para obtener instrucciones, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora.	Aurora MySQL y Aurora PostgreSQL	Bytes
`ServerlessDatabaseCapacity`	La capacidad actual de un clúster de bases de datos Aurora Serverless.	Aurora MySQL y Aurora PostgreSQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
`SnapshotStorageUsed`	La cantidad total de almacenamiento de copias de seguridad consumida por todas las instantán eas de Aurora para un clúster de base de datos de Aurora fuera de su periodo de retención de copia de seguridad. Esta cantidad se incluye en el total notificado por la métrica `TotalBack upStorageBilled` . Se calcula de forma independi ente para cada clúster de Aurora. Para obtener instrucciones, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora.	Aurora MySQL y Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
`TotalBackupStorageBilled`	La cantidad total de almacenamiento de copias de seguridad en bytes facturada para un clúster de base de datos de Aurora determinado. La métrica incluye el almacenamiento de copias de seguridad medido por las métricas `BackupRetentionPeriodStorageUsed` y `SnapshotStorageUsed`. Esta métrica se calcula por separado para cada clúster de Aurora. Para obtener instrucciones, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora.	Aurora MySQL y Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
`VolumeBytesUsed`	La cantidad de almacenamiento que utiliza su clúster de base de datos de Aurora. Este valor afecta al costo del clúster de base de datos de Aurora (para ver la información de precios, consulte la página de precios de Amazon RDS). Este valor no refleja algunas asignaciones de almacenamiento interno que no afectan a la facturación de almacenamiento. Para Aurora MySQL, puede prever problemas de agotamiento de espacio con mayor precisión probando si `AuroraVolumeBytesLeftTotal` se acerca a cero en lugar de comparar `VolumeBytesUsed` con el límite de almacenamiento de 128 TiB. Para los clústeres que son clones, el valor de esta métrica depende de la cantidad de datos agregados o modificados en el clon. La métrica también puede aumentar o disminuir cuando se elimina	Aurora MySQL y Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
	el clúster original o cuando se agregan nuevos clones o se eliminan. Para obtener más información, consulte Eliminación de un volumen del clúster de origen Tenga en cuenta que no tiene sentido elegir un valor `--period` que sea pequeño, porque Amazon RDS recopila esta métrica a intervalos, no de forma continua.		

Métrica	Descripción	Se aplica a	Unidades
`VolumeReadIOPs`	El número de operaciones de E/S de lectura facturadas desde un volumen de clúster en un intervalo de 5 minutos. Las operaciones de lectura facturadas se calculan en el nivel del volumen de clúster, se agrupan desde todas las instancias del clúster de bases de datos de Aurora y se notifican a continuación a intervalos de 5 minutos. El valor se calcula tomando el valor de la métrica Read operations (Operaciones de lectura) a lo largo de un periodo de 5 minutos. Puede determinar la cantidad de operaciones de lectura facturadas por segundo tomando el valor de la métrica Billed read operations (Operaciones de lectura facturadas) y dividiendo por 300 segundos. Por ejemplo, si Billed read operations (Operaciones de lectura facturadas) devuelve 13 686, las operaciones de lectura facturadas por segundo serán 45 (13 686/300 = 45,62).	Aurora MySQL y Aurora PostgreSQL	Recuento cada 5 minutos

Métrica	Descripción	Se aplica a	Unidades
	Las operaciones de lectura facturadas se acumulan para las consultas que solicitan páginas de la base de datos que no están en la caché del búfer y que se deben cargar desde el almacenamiento. Es posible que aparezcan picos en las operaciones de lectura facturadas, ya que los resultados de la consulta se leen desde el almacenam iento y se cargan en la caché del búfer. Tenga en cuenta que no tiene sentido elegir un valor `--period` que sea pequeño, porque Amazon RDS recopila esta métrica a intervalos, no de forma continua. **Tip** Si su clúster de Aurora MySQL utiliza una consulta en paralelo, es posible que vea un aumento en los valores `VolumeReadIOPS` . Las consultas en		

Métrica	Descripción	Se aplica a	Unidades
	paralelo no utilizan el grupo de búfer. Por lo tanto, si bien las consultas son rápidas, este procesamiento optimizado puede dar como resultado un aumento de las operaciones de lectura y los cargos asociados.		
`VolumeWriteIOPs`	Número de operaciones de E/S de escritura en el disco en el volumen de clúster indicadas a intervalos de 5 minutos. Para obtener una descripci ón detallada de cómo se calculan las operaciones de escritura facturada, consulte `VolumeReadIOPs` . Tenga en cuenta que no tiene sentido elegir un valor `--period` que sea pequeño, porque Amazon RDS recopila esta métrica a intervalos, no de forma continua.	Aurora MySQL y Aurora PostgreSQL	Recuento cada 5 minutos

Métricas de nivel de instancia para Amazon Aurora

Las siguientes métricas de Amazon CloudWatch específicas de instancias se aplican a todas las instancias de Aurora MySQL y Aurora PostgreSQL, a menos que se indique lo contrario.

Métrica	Descripción	Se aplica a	Unidades
`AbortedClients`	El número de conexiones de cliente que no se han cerrado correctamente.	Aurora MySQL	Recuento
`ActiveTransactions`	Número medio de transacciones que se ejecutan en una instancia de base de datos de Aurora por segundo. De forma predeterminada, Aurora no habilita esta métrica. Para comenzar a medir este valor, establezca `innodb_monitor_enable='all'` en el grupo de parámetros de base de datos para una instancia de base de datos específica.	Aurora MySQL	Recuento por segundo
`ACUUtilization`	Es el valor de la métrica `ServerlessDatabaseCapacity` dividido por el valor máximo de ACU del clúster de base de datos. Esta métrica es aplicable solo para la versión 2 de Aurora sin servidor.	Aurora MySQL y Aurora PostgreSQL	Percentage

Métrica	Descripción	Se aplica a	Unidades
`AuroraBinlogReplicaLag`	Cantidad de tiempo de retraso de un clúster de bases de datos de réplica de registro binario en Aurora MySQL con respecto al origen de replicación de registro binario. Un retraso significa que el origen está generando registros más rápido de lo que la réplica puede aplicarlos. Esta métrica informa diferentes valores dependiendo de la versión del motor: Aurora MySQL versión 2 El campo `Seconds_Behind_Master` de MySQL `SHOW SLAVE STATUS` Aurora MySQL versión 3 `SHOW REPLICA STATUS` Puede utilizar esta métrica para supervisar los errores y el retraso de réplica en un clúster que actúa como réplica de registro binario. El valor de métrica indica lo siguiente:	Principal para Aurora MySQL	Segundos

Métrica	Descripción	Se aplica a	Unidades
	Un gran valor La réplica está retrasando el origen de replicación. **`0`** o un valor cercano a **`0`** El proceso de réplica está activo y actualizado. **`-1`** Aurora no puede determinar el retraso, que puede ocurrir durante la configuración de la réplica o cuando la réplica está en un estado de error. Dado que la replicación de registros binarios solo se produce en la instancia de escritor del clúster, se recomienda utilizar la versión de esta métrica asociada al rol WRITER. Para obtener más informaci ón acerca del uso de la replicación, consulte Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS. Para obtener más informaci ón acerca de la solución de problemas, consulte		

Métrica	Descripción	Se aplica a	Unidades
	Problemas de replicación de Amazon Aurora MySQL.		
`AuroraDMLRejectedMasterFull`	El número de consultas reenviadas que se han rechazado porque la sesión en la instancia de base de datos de escritura está completa.	Principal para Aurora MySQL, versión 2	Recuento
`AuroraDMLRejectedWriterFull`	El número de consultas reenviadas que se han rechazado porque la sesión en la instancia de base de datos de escritura está completa.	Principal para Aurora MySQL, versión 3	Recuento
`AuroraEstimatedSharedMemoryBytes`	La cantidad estimada de memoria del búfer compartido o del grupo de búfer que se utilizó activamente durante el último intervalo de sondeo configurado.	Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
`AuroraMemoryHealthState`	Indica el estado de la memoria. Un valor `0` equivale a `NORMAL`. Un valor de `10` equivale a `RESERVED`, lo que significa que el servidor se acerca a un nivel crítico de uso de memoria. Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.	Aurora MySQL versión y posteriores	Calibre
`AuroraMemoryNumDeclinedSqlTotal`	Es el número incremental de consultas rechazadas como parte de las estrategias para evitar el problema de memoria insuficiente (OOM). Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.	Aurora MySQL versión y posteriores	Recuento

Métrica	Descripción	Se aplica a	Unidades
`AuroraMemoryNumKilConnTotal`	Número incremental de conexiones cerradas como parte de las estrategias para evitar el problema de memoria insuficiente (OOM). Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.	Aurora MySQL versión y posteriores	Recuento
`AuroraMemoryNumKilQueryTotal`	Número incremental de consultas finalizadas como parte de las estrategias para evitar el problema de memoria insuficiente (OOM). Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.	Aurora MySQL versión y posteriores	Recuento
`AuroraMillisecondsSpentInOomRecovery`	El tiempo transcurrido desde que el estado de la memoria se situara por debajo del estado normal. Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.	Aurora MySQL versión 3.08.0 y posteriores	Milisegundos

Métrica	Descripción	Se aplica a	Unidades
`AuroraNumOomRecoveryTriggered`	El número de veces que el estado de la memoria se ha situado por debajo del estado normal. Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.	Aurora MySQL versión 3.08.0 y posteriores	Recuento

Métrica	Descripción	Se aplica a	Unidades
`AuroraOptimizedReadsCacheHitRatio`	Porcentaje de solicitudes atendidas por la caché de lecturas optimizadas. El valor se calcula con la siguiente fórmula: `orcache_blks_hit/ (orcache_blks_hit + storage_blks_read)` Si `AuroraOptimizedReadsCacheHitRatio` es 100 %, significa que se han leído todas las páginas de la caché de lecturas optimizadas. Si `AuroraOptimizedReadsCacheHitRatio` es `0`, significa que no se ha leído ninguna página de la caché de lecturas optimizadas.	Principal para Aurora PostgreSQL	Porcentaje
`AuroraReplicaLag`	Para una réplica de Aurora, el retardo en milisegundos que se da cuando la replicación se actualiza desde la instancia principal.	Réplica para Aurora MySQL y Aurora PostgreSQL	Milisegundos

Métrica	Descripción	Se aplica a	Unidades
`AuroraReplicaLagMaximum`	El retardo máximo entre la instancia principal y cualquiera de las instancias de base de datos de Aurora del clúster de base de datos. Cuando las réplicas de lectura se eliminan o se les cambia el nombre, puede producirse un aumento temporal en el retraso de la replicación a medida que el recurso antiguo se somete a un proceso de reciclaje. Para obtener una representación precisa del retraso de la replicación durante ese periodo, le recomendamos que supervise la métrica `AuroraReplicaLag` de cada instancia de réplica de lectura.	Principal para Aurora MySQL y Aurora PostgreSQL	Milisegundos
`AuroraReplicaLagMinimum`	El retardo mínimo entre la instancia principal y cualquiera de las instancias de base de datos de Aurora del clúster de base de datos.	Principal para Aurora MySQL y Aurora PostgreSQL	Milisegundos

Métrica	Descripción	Se aplica a	Unidades
`AuroraSlowConnectionHandleCount`	Número de conexiones que han esperado dos segundos o más para iniciar el protocolo de enlace. Esta métrica solo se aplica a Aurora MySQL versión 3.	Aurora MySQL	Recuento
`AuroraSlowHandshakeCount`	Número de conexiones que han tardado 50 milisegundos o más en finalizar el protocolo de enlace. Esta métrica solo se aplica a Aurora MySQL versión 3.	Aurora MySQL	Recuento
`BacktrackWindowActual`	Diferencia entre el intervalo de destino para el seguimiento de datos anteriores y el intervalo real.	Principal para Aurora MySQL	Minutos
`BacktrackWindowAlert`	Número de veces que el intervalo real para el seguimiento de datos anteriores es inferior al de destino en un determinado periodo.	Principal para Aurora MySQL	Recuento
`BlockedTransactions`	Número medio de transacciones de la base de datos que se bloquean cada segundo.	Aurora MySQL	Recuento por segundo
`BufferCacheHitRatio`	Porcentaje de solicitudes que se responden desde la caché de búfer.	Aurora MySQL y Aurora PostgreSQL	Percentage

Métrica	Descripción	Se aplica a	Unidades
`CommitLatency`	El tiempo medio que tarda el motor y el almacenamiento en completar las operaciones de confirmación.	Aurora MySQL y Aurora PostgreSQL	Milisegundos
`CommitThroughput`	Número medio de operacion es de confirmación por segundo.	Aurora MySQL y Aurora PostgreSQL	Recuento por segundo
`ConnectionAttempts`	Número de intentos de conexión a una instancia , tanto si se han realizado correctamente como si no.	Aurora MySQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
`CPUCreditBalance`	El número de créditos de CPU que una instancia ha acumulado, notificados en intervalos de 5 minutos. Puede utilizar esta métrica para determinar cuánto tiempo puede una instancia de base de datos de sobrepasar temporalmente su nivel de rendimiento de referencia a una velocidad dada. Esta métrica se aplica únicamente a estas clases de instancias: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` y `db.t4g` • Aurora PostgreSQL: `db.t3` y `db.t4g` **Note** Recomendamos que se usen solo las clases de instancia de base de datos T para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la	Aurora MySQL y Aurora PostgreSQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
	producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos. Los créditos de lanzamiento funcionan de la misma manera en Amazon RDS que en Amazon EC2. Para obtener más información, consulte Créditos de lanzamiento en la Guía del usuario de Amazon Elastic Compute Cloud para instancias de Linux.		

Métrica	Descripción	Se aplica a	Unidades
`CPUCreditUsage`	El número de créditos de CPU consumidos durante el período especificado, notificados en intervalos de 5 minutos. Esta métrica mide la cantidad de tiempo durante la que las CPU físicas se han usado para procesar instrucciones de las CPU virtuales asignadas a la instancia de base de datos. Esta métrica se aplica únicamente a estas clases de instancias: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` y `db.t4g` • Aurora PostgreSQL: `db.t3` y `db.t4g` **Note** Recomendamos que se usen solo las clases de instancia de base de datos T para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para	Aurora MySQL y Aurora PostgreSQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
	obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.		
`CPUSurplusCreditBalance`	La cantidad de créditos sobrantes que ha gastado una instancia ilimitada cuando su valor `CPUCreditBalance` es igual a cero. El valor de `CPUSurplusCreditBalance` se compensa con los créditos de CPU obtenidos. Si el número de créditos sobrantes supera el número máximo de créditos que la instancia puede ganar en un periodo de 24 horas, los créditos sobrantes gastados por encima del máximo implican un cargo adicional. Las métricas de créditos de CPU solo se encuentran disponibles cada 5 minutos.	Aurora MySQL y Aurora PostgreSQL	Créditos (vCPU/minutos)

Métrica	Descripción	Se aplica a	Unidades
`CPUSurplusCreditsCharged`	La cantidad de créditos sobrantes gastados que no se han compensado con créditos de CPU obtenido y, por lo tanto, implican un cargo adicional. Los créditos sobrantes gastados se cobran cuando se da alguno de los casos siguientes: • Los créditos sobrantes gastados superan el número máximo de créditos que la instancia puede obtener en un periodo de 24 horas. Los créditos sobrantes gastados por encima de la cantidad máxima se cobran al final de la hora. • La instancia se detiene o se termina. • La instancia se cambia de `unlimited` a `standard`. Las métricas de créditos de CPU solo se encuentran disponibles cada 5 minutos.	Aurora MySQL y Aurora PostgreSQL	Créditos (vCPU/minutos)

Métrica	Descripción	Se aplica a	Unidades
`CPUUtilization`	Porcentaje de CPU usado por una instancia de base de datos de Aurora.	Aurora MySQL y Aurora PostgreSQL	Percentage
`DatabaseConnections`	El número de conexiones de red de cliente a la instancia de base de datos. El número de sesiones de base de datos puede ser superior al valor de métrica porque el valor de métrica no incluye lo siguiente: • Sesiones que ya no tienen conexión de red, pero que la base de datos no ha limpiado. • Sesiones creadas por el motor de base de datos para sus propios fines. • Sesiones creadas por las capacidades de ejecución paralela del motor de base de datos. • Sesiones creadas por el programador de trabajos del motor de base de datos. • Conexiones de Amazon Aurora	Aurora MySQL y Aurora PostgreSQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
DDLLatency	Duración promedio de solicitudes como, por ejemplo, las solicitudes para crear, alterar y eliminar.	Aurora MySQL	Milisegundos
DDLThroughput	Número medio de solicitudes DDL por segundo.	Aurora MySQL	Recuento por segundo
Deadlocks	Número medio de interbloq ueos en la base de datos por segundo.	Aurora MySQL y Aurora PostgreSQL	Recuento por segundo
DeleteLatency	Duración promedio de las operaciones de eliminación.	Aurora MySQL	Milisegundos
DeleteThroughput	Número medio de consultas de eliminación por segundo.	Aurora MySQL	Recuento por segundo
DiskQueueDepth	El número de solicitudes de lectura/escritura pendientes a la espera de obtener acceso al disco.	Aurora MySQL y Aurora PostgreSQL	Recuento
DMLLatency	Duración promedio de inserciones, actualizaciones y eliminaciones.	Aurora MySQL	Milisegundos
DMLThroughput	Número medio de insercion es, actualizaciones y eliminaciones por segundo.	Aurora MySQL	Recuento por segundo
EngineUptime	La cantidad de tiempo que la instancia lleva en ejecución.	Aurora MySQL y Aurora PostgreSQL	Segundos

Métrica	Descripción	Se aplica a	Unidades
`FreeableMemory`	La cantidad de memoria de acceso aleatorio disponible. En el caso de las bases de datos de Aurora MySQL y Aurora PostgreSQL, esta métrica informa del valor del campo `MemAvailable` de `/proc/meminfo` .	Aurora MySQL y Aurora PostgreSQL	Bytes
`FreeEphemeralStorage`	La cantidad de almacenamiento de NVMe efímero disponible.	Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
`FreeLocalStorage`	La cantidad de almacenamiento local disponible. A diferencia de otros motores de base de datos, en las instancias de base de datos de Aurora esta métrica indica la cantidad de almacenamiento disponible en cada instancia de base de datos. Este valor depende de la clase de la instancia de base de datos (para ver la información de precios, consulte la página de precios de Amazon RDS). Puede aumentar la cantidad de espacio de almacenamiento libre para una instancia eligiendo una clase de instancia de base de datos más grande para ella. (Esto no se aplica a). Aurora Serverless v2.)	Aurora MySQL y Aurora PostgreSQL	Bytes
`InsertLatency`	Duración promedio de las operaciones de inserción.	Aurora MySQL	Milisegundos
`InsertThroughput`	Número medio de operaciones de inserción por segundo.	Aurora MySQL	Recuento por segundo
`LoginFailures`	Número medio de intentos de inicio de sesión con error por segundo.	Aurora MySQL	Recuento por segundo

Métrica	Descripción	Se aplica a	Unidades
`MaximumUsedTransactionIDs`	La antigüedad del ID de transacción sin vaciar más antiguo, en transacciones. Si este valor alcanza 2 146 483 648 (2^31 - 1 000 000), la base de datos se fuerza en modo de solo lectura, a fin de evitar el reinicio de los ID de transacción. Para obtener más información, consulte Prevención de fallos en la identificación de las transacciones en la documentación de PostgreSQL.	Aurora PostgreSQL	Recuento
`NetworkReceiveThroughput`	La cantidad de rendimiento de red recibida de los clientes por cada instancia en el clúster de bases de datos de Aurora. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de Aurora y el volumen de clúster.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo (la consola muestra megabytes por segundo)

Métrica	Descripción	Se aplica a	Unidades
`NetworkThroughput`	La cantidad de rendimiento de red recibida de los clientes y transmitida a ellos por cada instancia en el clúster de bases de datos de Aurora. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de Aurora y el volumen de clúster.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo
`NetworkTransmitThroughput`	El rendimiento de red enviado a los clientes por cada instancia del clúster de base de datos de Aurora. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de y el volumen de clúster.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo (la consola muestra megabytes por segundo)
`NumBinaryLogFiles`	El número de archivos binlog generados.	Aurora MySQL	Recuento
`OldestReplicationSlotLag`	El tamaño de retardo de la réplica con mayor retardo en cuanto a los datos de registro de escritura anticipada (WAL) recibidos.	Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
PurgeBoundary	Número de transacción hasta el que se permite la depuración de InnoDB. Si esta métrica no aumenta durante periodos de tiempo prolongados, es una buena indicación de que las transacciones de larga duración bloquean la depuración de InnoDB. Para investigar, compruebe las transacciones activas en el clúster de base de datos de Aurora MySQL.	Aurora MySQL versión 2.11 y posteriores Aurora MySQL versión 3, versiones 3.08 y posteriores	Recuento
PurgeFinishedPoint	Número de transacción hasta el que se realiza la depuración de InnoDB. Esta métrica puede ser de utilidad para evaluar la rapidez con la que avanza la depuración de InnoDB.	Aurora MySQL versión 2.11 y posteriores Aurora MySQL versión 3, versiones 3.08 y posteriores	Recuento
Queries	Número medio de consultas ejecutadas por segundo.	Aurora MySQL	Recuento por segundo
RDSToAuroraPostgreSQLReplicaLag	El retardo cuando se replican actualizaciones desde la instancia de principal de RDS PostgreSQL a otros nodos del clúster.	Réplica para Aurora PostgreSQL	Segundos

Métrica	Descripción	Se aplica a	Unidades
`ReadIOPS`	Es el número medio de operaciones de E/S en disco por segundo, pero los informes leen y escriben por separado y en intervalos de un minuto.	Aurora MySQL y Aurora PostgreSQL	Recuento por segundo
`ReadIOPSEphemeralStorage`	Número promedio de operaciones de E/S de lectura en disco al almacenamiento NVMe efímero.	Aurora PostgreSQL	Recuento por segundo
`ReadLatency`	Tiempo medio de cada operación de E/S en el disco.	Aurora MySQL y Aurora PostgreSQL	Segundos
`ReadLatencyEphemeralStorage`	Tiempo promedio que toma cada operación de E/S de lectura en disco para el almacenamiento NVMe efímero.	Aurora PostgreSQL	Segundos
`ReadThroughput`	El número medio de bytes leídos del disco por segundo.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo
`ReadThroughputEphemeralStorage`	Número promedio de bytes leídos en el disco por segundo para el almacenamiento NVMe efímero.	Aurora PostgreSQL	Bytes por segundo

Métrica	Descripción	Se aplica a	Unidades
`ReplicationSlotDiskUsage`	La cantidad de espacio en disco consumida por los archivos de ranura de la replicación.	Aurora PostgreSQL	Bytes
`ResultSetCacheHitRatio`	Porcentaje de solicitudes servidas por la caché Resultset.	Aurora MySQL versión	Porcentaje
`RollbackSegmentHistoryListLength`	Los registros de deshacer que registran transacciones confirmadas con registros marcados para eliminación. Estos registros están programados para ser procesados por la operación de depuración de InnoDB.	Aurora MySQL	Recuento
`RowLockTime`	El tiempo total dedicado a adquirir bloqueos de fila para tablas de InnoDB.	Aurora MySQL	Milisegundos
`SelectLatency`	La cantidad promedio de tiempo para las operaciones de selección.	Aurora MySQL	Milisegundos
`SelectThroughput`	Número medio de consultas de selección por segundo.	Aurora MySQL	Recuento por segundo
`ServerlessDatabaseCapacity`	La capacidad actual de un clúster de bases de datos Aurora Serverless.	Aurora MySQL y Aurora PostgreSQL	Recuento

Métrica	Descripción	Se aplica a	Unidades
`StorageNetworkReceiveThroughput`	El rendimiento de red recibido del subsistema de almacenamiento de Aurora por cada instancia del clúster de bases de datos.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo
`StorageNetworkThroughput`	La cantidad de rendimiento de red recibida del subsistema de almacenamiento de Aurora y enviada al mismo por cada instancia en el clúster de bases de datos de Aurora.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo
`StorageNetworkTransmitThroughput`	La cantidad de rendimiento de red enviada al subsistema de almacenamiento de Aurora por cada instancia en el clúster de bases de datos de Aurora.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo
`SumBinaryLogSize`	El tamaño total de los archivos binlog.	Aurora MySQL	Bytes
`SwapUsage`	La cantidad de espacio de intercambio utilizada. Esta métrica no está disponible para las siguientes clases de instancia de base de datos: • db.r3.*, db.r4.* y db.r7g.* (Aurora MySQL) • db.r7g.* (Aurora PostgreSQL)	Aurora MySQL y Aurora PostgreSQL	Bytes

Métrica	Descripción	Se aplica a	Unidades
`TempStorageIOPS`	Es el número de IOPS para lectura y escritura en el almacenamiento local adjunto a la instancia de base de datos. Esta métrica represent a un recuento y se mide una vez por segundo. Esta métrica es aplicable solo para la versión 2 de Aurora sin servidor.	Aurora MySQL y Aurora PostgreSQL	Recuento por segundo
`TempStorageThroughput`	Es la cantidad de datos transferidos desde y hacia el almacenamiento local asociado a la instancia de base de datos. Esta métrica representa un número de bytes y se mide una vez por segundo. Esta métrica es aplicable solo para la versión 2 de Aurora sin servidor.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo
`TransactionAgeMaximum`	Es la antigüedad de la transacción en ejecución más antigua.	Aurora MySQL versión 3, versiones 3.08 y posteriores	Segundos

Métrica	Descripción	Se aplica a	Unidades
`TransactionLogsDiskUsage`	La cantidad de espacio en disco consumida por los registros de transacciones en la instancia de base de datos de Aurora PostgreSQL. Esta métrica solo se genera cuando Aurora PostgreSQL utiliza la replicación lógica o AWS Database Migration Service. De forma predeterminada, Aurora PostgreSQL utiliza registros, no registros de transacciones. Cuando los registros de transacciones no están en uso, el valor de esta métrica es `-1`.	Principal para Aurora PostgreSQL	Bytes
`TruncateFinishedPoint`	Identificador de transacción hasta el que se realiza la operación de deshacer el truncado.	Aurora MySQL versión 2.11 y posteriores Aurora MySQL versión 3, versiones 3.08 y posteriores	Recuento
`UpdateLatency`	La cantidad promedio de tiempo para las operaciones de actualización.	Aurora MySQL	Milisegundos
`UpdateThroughput`	Número medio de actualizaciones por segundo.	Aurora MySQL	Recuento por segundo

Métrica	Descripción	Se aplica a	Unidades
`WriteIOPS`	El número de registros de escritura de almacenamiento de Aurora generados por segundo. Es más o menos el número de registros generados por la base de datos. Estos no corresponden a escrituras de páginas de 8K y no corresponden a paquetes de red enviados.	Aurora MySQL y Aurora PostgreSQL	Recuento por segundo
`WriteIOPSEphemeralStorage`	Número promedio de operaciones de E/S de escritura en disco en el almacenamiento NVMe efímero.	Aurora PostgreSQL	Recuento por segundo
`WriteLatency`	Tiempo medio de cada operación de E/S en el disco.	Aurora MySQL y Aurora PostgreSQL	Segundos
`WriteLatencyEphemeralStorage`	Tiempo promedio que toma cada operación de E/S de escritura en disco para el almacenamiento NVMe efímero.	Aurora PostgreSQL	Segundos
`WriteThroughput`	Número medio de bytes escritos en almacenamiento persistente cada segundo.	Aurora MySQL y Aurora PostgreSQL	Bytes por segundo

Métrica	Descripción	Se aplica a	Unidades
`WriteThroughputEphemeralStorage`	Número promedio de bytes escritos en el disco por segundo para el almacenamiento NVMe efímero.	Aurora PostgreSQL	Bytes por segundo

Métricas de uso de Amazon CloudWatch para Amazon Aurora

El espacio de nombres `AWS/Usage` de Amazon CloudWatch incluye métricas de uso en el nivel de cuenta para sus cuotas de servicio de Amazon RDS. CloudWatch recopila métricas de uso automáticamente para todas las Regiones de AWS.

Para obtener más información, consulte Uso de métricas de Amazon CloudWatch en la Guía del usuario de Amazon CloudWatch. Para obtener más información acerca de las cuotas, consulte Cuotas y restricciones para Amazon Aurora y Solicitud de un aumento de cuota en la Guía del usuario de Service Quotas.

Métrica	Descripción	Unidades*
`AuthorizationsPerDBSecurityGroup`	Es el número de reglas de entrada por grupo de seguridad de base de datos de su Cuenta de AWS. El valor utilizado es el número más alto de reglas de entrada en un grupo de seguridad de base de datos de la cuenta. Es posible que otros grupos de seguridad de base de datos de la cuenta tengan un número menor de reglas de entrada.	Recuento
`CustomEndpointsPerDBCluster`	Es el número de puntos de conexión personalizados por clúster de base de datos de su Cuenta de AWS. El valor utilizado es el número más alto de puntos de conexión personalizados en un clúster de base de datos de la cuenta. Es posible que otros clústeres de bases de datos de la cuenta tengan un número inferior de puntos de conexión personalizados.	Recuento

Métrica	Descripción	Unidades*
DBCluster ParameterGroups	El número de grupos de parámetros del clúster de base de datos en la Cuenta de AWS. El recuento excluye los grupos de parámetros predeterminados.	Recuento
DBClusterRoles	Es el número de roles de AWS Identity and Access Management (IAM) asociados por clúster de base de datos en su Cuenta de AWS. El valor utilizado es el número máximo de roles de IAM asociados a un clúster de base de datos de la cuenta. Es posible que otros clústeres de bases de datos de la cuenta tengan un número inferior de roles de IAM asociados.	Recuento
DBClusters	El número de clústeres de base de datos de Amazon Aurora en la Cuenta de AWS.	Recuento
DBInstanceRoles	Es el número de roles de AWS Identity and Access Management (IAM) asociados por instancia de base de datos en su Cuenta de AWS. El valor utilizado es el número máximo de roles de IAM asociados a una instancia de base de datos de la cuenta. Es posible que otras instancias de bases de datos de la cuenta tengan un número inferior de roles de IAM asociados.	Recuento
DBInstances	El número de instancias de base de datos de la Cuenta de AWS.	Recuento
DBParamet erGroups	El número de grupos de parámetros de base de datos de la Cuenta de AWS. El recuento excluye los grupos de parámetros de base de datos predeterminados.	Recuento
DBSubnetGroups	El número de grupos de subredes de base de datos de la Cuenta de AWS. El recuento excluye el grupo de subred predeterminado.	Recuento
EventSubs criptions	El número de suscripciones de notificación de eventos en la Cuenta de AWS.	Recuento

Métrica	Descripción	Unidades*
`Integrations`	El número de integraciones sin ETL con Amazon Redshift en la Cuenta de AWS.	Recuento
`ManualClusterSnapshots`	El número de instantáneas de clúster de base de datos creadas manualmente en la Cuenta de AWS. El recuento excluye las instantáneas no válidas.	Recuento
`OptionGroups`	El número de grupos de opciones de la Cuenta de AWS. El recuento excluye los grupos de opciones predeterminados.	Recuento
`Proxies`	Es el número de proxies de RDS en su cuenta de AWS.	Recuento
`ReadReplicasPerMaster`	Es el número de réplicas de lectura por instancia de base de datos en la cuenta. El valor utilizado es el número máximo de réplicas de lectura para una instancia de base de datos de la cuenta. Es posible que otras instancias de bases de datos de la cuenta tengan un número inferior de réplicas de lectura.	Recuento
`ReservedDBInstances`	El número de instancias de base de datos reservadas en la Cuenta de AWS. El recuento excluye las instancias retiradas o rechazadas.	Recuento
`SubnetsPerDBSubnetGroup`	Número de subredes por grupo de subredes de base de datos en su Cuenta de AWS. Es el número más alto de subredes de un grupo de subredes de base de datos de la cuenta. Es posible que otros grupos de subredes de base de datos de la cuenta tengan un número menor de subredes.	Recuento

> **Note**
>
> Amazon RDS no publica unidades para métricas de uso en CloudWatch. Las unidades solo aparecen en la documentación.

Dimensiones de Amazon CloudWatch para Aurora.

Los datos de las métricas de Aurora se pueden filtrar usando cualesquiera de las dimensiones de la tabla siguiente:

Dimensión	Filtrar los datos solicitados por…
`DBInstanceIdentifier`	Una instancia de base de datos específica.
`DBClusterIdentifier`	Un clúster de de base de datos de Aurora específico.
`DBClusterIdentifier, Role`	Un clúster de Aurora base de datos específico y agregado a la métrica por rol de instancia (WRITER/READER). Por ejemplo, puede agregar métricas para todas las instancias READER que pertenezcan a un clúster.
`DbClusterIdentifier, EngineName`	Una combinación específica de clúster de base de datos de Aurora y nombre de motor Por ejemplo, puede ver la métrica de `VolumeReadIOPs` para el clúster `ams1` y el motor `aurora`.
`DatabaseClass`	Todas las instancias de una clase de base de datos. Por ejemplo, puede agregar métricas para todas las instancias que pertenezcan a la clase de base de datos `db.r5.large` .
`EngineName`	Sólo el nombre del motor identificado. Por ejemplo, puede agregar métricas para todas las instancias que tengan el nombre de motor `aurora-postgresql` .
`SourceRegion`	Use la región especificada. Por ejemplo, puede agregar métricas para todas las instancias de bases de datos de la `us-east-1` región.

Disponibilidad de métricas de Aurora en la consola de Amazon RDS.

No todas las métricas proporcionadas por Amazon Aurora están disponibles en la consola de Amazon RDS. Sin embargo, puede verlas mediante otras herramientas, como la AWS CLI y la API de CloudWatch. Además, algunas de las métricas que están disponibles en la consola de Amazon

RDS solo se muestran para clases de instancias concretas o con nombres distintos y con unidades de medida diferentes.

Temas

- Métricas de Aurora disponibles en la vista de última hora
- Métricas de Aurora disponibles en casos específicos
- Métricas de Aurora que no están disponibles en la consola

Métricas de Aurora disponibles en la vista de última hora

Puede ver un subconjunto de métricas de Aurora clasificadas en la vista de última hora de la consola de Amazon RDS. En la siguiente tabla se muestran las categorías y las métricas asociadas que se muestran en la consola de Amazon RDS para una instancia de Aurora.

Categoría	Métricas

SQL	`ActiveTransactions`
	`BlockedTransactions`
	`BufferCacheHitRatio`
	`CommitLatency`
	`CommitThroughput`
	`DatabaseConnections`
	`DDLLatency`
	`DDLThroughput`
	`Deadlocks`
	`DMLLatency`
	`DMLThroughput`
	`LoginFailures`

Categoría	Métricas

	`ResultSetCacheHitRatio`
	`SelectLatency`
	`SelectThroughput`

System (Sistema)	`AuroraReplicaLag`
	`AuroraReplicaLagMaximum`
	`AuroraReplicaLagMinimum`
	`CPUCreditBalance`
	`CPUCreditUsage`
	`CPUUtilization`
	`FreeableMemory`
	`FreeLocalStorage` (Esto no se aplica a Aurora Serverless v2).
	`NetworkReceiveThroughput`

Implementación	`AuroraReplicaLag`
	`BufferCacheHitRatio`
	`ResultSetCacheHitRatio`
	`SelectThroughput`

Métricas de Aurora disponibles en casos específicos

Además, algunas métricas de Aurora solo se muestran para clases de instancias concretas, para instancias de base de datos o con nombres distintos y con unidades de medida diferentes:

- Las métricas `CPUCreditBalance` y `CPUCreditUsage` se muestran solo para las clases de instancia de Aurora MySQL `db.t2` y para las clases de instancia de Aurora PostgreSQL `db.t3`.

- Las siguientes métricas se muestran con nombres diferentes, como se indica en la lista:

Métrica	Nombre que mostrar
`AuroraReplicaLagMaximum`	Replica lag maximum
`AuroraReplicaLagMinimum`	Replica lag minimum
`DDLThroughput`	DDL
`NetworkReceiveThroughput`	Network throughput
`VolumeBytesUsed`	Bytes de volumen [facturados] usados
`VolumeReadIOPs`	IOPS de lectura de volumen [facturadas]
`VolumeWriteIOPs`	IOPS de escritura de volumen [facturadas]

- Las siguientes métricas se aplican a un clúster de bases de datos Aurora completo, pero se muestran solo al ver instancias de base de datos para un clúster de bases de datos Aurora en la consola de Amazon RDS:
  - `VolumeBytesUsed`
  - `VolumeReadIOPs`
  - `VolumeWriteIOPs`
- Las siguientes métricas se muestran en megabytes y no en bytes en la consola de Amazon RDS:
  - `FreeableMemory`
  - `FreeLocalStorage`
  - `NetworkReceiveThroughput`
  - `NetworkTransmitThroughput`
- Las siguientes métricas se aplican a un clúster de base de datos de Aurora PostgreSQL con lecturas optimizadas de Aurora:
  - `AuroraOptimizedReadsCacheHitRatio`
  - `FreeEphemeralStorage`
  - `ReadIOPSEphemeralStorage`
  - `ReadLatencyEphemeralStorage`
  - `ReadThroughputEphemeralStorage`

- `WriteIOPSEphemeralStorage`
- `WriteLatencyEphemeralStorage`
- `WriteThroughputEphemeralStorage`

Métricas de Aurora que no están disponibles en la consola

Las siguientes métricas de Aurora no están disponibles en la consola de Amazon RDS:

- `AuroraBinlogReplicaLag`
- `DeleteLatency`
- `DeleteThroughput`
- `EngineUptime`
- `InsertLatency`
- `InsertThroughput`
- `NetworkThroughput`
- `Queries`
- `UpdateLatency`
- `UpdateThroughput`

Métricas de Amazon CloudWatch para Información de rendimiento de Amazon RDS

Performance Insights publica automáticamente algunas métricas en Amazon CloudWatch. Se pueden consultar los mismos datos en Performance Insights, pero al contar con las métricas en CloudWatch es sencillo añadir alarmas de CloudWatch. También resulta fácil añadir las métricas a paneles de CloudWatch existentes.

Métrica	Descripción
DBLoad	El número de sesiones activas de la base de datos. Normalmente, necesita los datos del número promedio de sesiones activas. En Performance Insights, estos datos se consultan como `db.load.avg` .

Métrica	Descripción
DBLoadCPU	El número de sesiones activas cuyo tipo de evento de espera es CPU. En Performance Insights, estos datos se consultan como `db.load.avg` , filtrados por el tipo de evento de espera `CPU`.
DBLoadNonCPU	Promedio de sesiones activas cuyo tipo de evento de espera no es CPU.
DBLoadRelativeToNumVCPUs	La relación entre la carga de base de datos y el número de CPU virtuales para la base de datos.

> **Note**
>
> Estas métricas se publican en CloudWatch solo si hay una carga en la instancia de base de datos.

Puede examinar estas métricas mediante la consola de CloudWatch, la AWS CLI o la API de CloudWatch. También puede examinar otras métricas de contador de Performance Insights mediante una función matemática métrica especial. Para obtener más información, consulte Consulta de otras métricas de contador de Performance Insights en CloudWatch.

Por ejemplo, puede obtener las estadísticas para la métrica `DBLoad` ejecutando el comando get-metric-statistics.

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

Este ejemplo genera un resultado similar al siguiente.


```
{
  "Datapoints": [
  {
  "Timestamp": "2021-07-19T21:30:00Z",
  "Unit": "None",
  "Average": 2.1
  },
  {
  "Timestamp": "2021-07-19T21:34:00Z",
  "Unit": "None",
  "Average": 1.7
  },
  {
  "Timestamp": "2021-07-19T21:35:00Z",
  "Unit": "None",
  "Average": 2.8
  },
  {
  "Timestamp": "2021-07-19T21:31:00Z",
  "Unit": "None",
  "Average": 1.5
  },
  {
  "Timestamp": "2021-07-19T21:32:00Z",
  "Unit": "None",
  "Average": 1.8
  },
  {
  "Timestamp": "2021-07-19T21:29:00Z",
  "Unit": "None",
  "Average": 3.0
  },
  {
  "Timestamp": "2021-07-19T21:33:00Z",
  "Unit": "None",
  "Average": 2.4
  }
  ],
  "Label": "DBLoad"
  }
```

Para obtener más información acerca de CloudWatch, consulte ¿Qué es Amazon CloudWatch? en la Guía del usuario de Amazon CloudWatch.

Consulta de otras métricas de contador de Performance Insights en CloudWatch

> **Note**
>
> Si habilita el modo avanzado de Información sobre las bases de datos, Amazon RDS publica las métricas del contador de Información de rendimiento en Amazon CloudWatch. Con Información sobre las bases de datos, no necesita utilizar la función matemática de métrica `DB_PERF_INSIGHTS`. Puede utilizar el panel de Información sobre las bases de datos de CloudWatch para buscar, consultar y configurar alarmas para las métricas de los contadores de Información de rendimiento.

Puede realizar consultas, generar alarmas y crear gráficos en las métricas de Performance Insights de RDS desde CloudWatch. Puede acceder a la información sobre su clúster de base de datos mediante la función matemática de la métrica `DB_PERF_INSIGHTS` para CloudWatch. Esta función le permite utilizar las métricas de Performance Insights que no se notifican directamente a CloudWatch para crear una nueva serie temporal.

Para utilizar la nueva función Metric Math, haga clic en el menú desplegable Agregar matemática, en la pantalla Seleccionar una métrica de la consola de CloudWatch. Puede usarla para crear alarmas y gráficos en métricas de Performance Insights o en combinaciones de métricas de CloudWatch y Performance Insights, lo que incluye alarmas de alta resolución para métricas de menos de un minuto. También puede utilizar la función mediante programación al incluir la expresión de Metric Math en una solicitud `get-metric-data`. Para obtener más información, consulte Sintaxis de matemáticas en las métricas y funciones y Crear una alarma en las métricas del contador de Performance Insights desde una base de datos AWS.

Métricas de contador de Información de rendimiento

Las métricas de contador son métricas de rendimiento de sistemas operativos y bases de datos en el panel de control de Información de rendimiento. Para ayudar a identificar y analizar los problemas de rendimiento, puede correlacionar las métricas de contador con la carga de base de datos. Debe anexar una función estadística a la métrica para obtener los valores de la métrica. Por ejemplo, las funciones compatibles con `os.memory.active`, las métricas son `.avg`, `.min`, `.max`, `.sum` y `.sample_count`.

Las métricas del contador se recopilan una vez por minuto. La recopilación de métricas del sistema operativo depende de si la monitorización mejorada está activada o desactivada. Si la monitorización mejorada está desactivada, las métricas del sistema operativo se recopilan una vez por minuto. Si la monitorización mejorada está activada, las métricas del sistema operativo se recopilan durante el período de tiempo seleccionado. Para obtener más información acerca de si activar o desactivar la monitorización mejorada, consulte Activación y desactivación de la supervisión mejorada.

Temas

- Contadores de sistemas operativos de Información de rendimiento
- Contadores de Información de rendimiento para Aurora MySQL
- Contadores de Información sobre rendimiento para Aurora PostgreSQL

Contadores de sistemas operativos de Información de rendimiento

Los siguientes contadores de sistemas operativos, que llevan el prefijo `os`, están disponibles con Información sobre rendimiento para Aurora PostgreSQL y Aurora MySQL.

Puede utilizar la API `ListAvailableResourceMetrics` para obtener la lista de métricas de contador disponibles para su instancia de base de datos. Para obtener más información, consulte ListAvailableResourceMetrics en la guía de referencia de la API Información de rendimiento de Amazon RDS.

Contador	Tipo	Unidad	Métrica	Descripción
Activa	Memoria	Kilobytes	os.memory .active	La cantidad de memoria asignada, en kilobytes.
Búferes	Memoria	Kilobytes	os.memory .buffers	La cantidad de memoria utilizada para almacenar en búfer solicitud es de E/S antes de escribir en el dispositivo de

Contador	Tipo	Unidad	Métrica	Descripción
				almacenamiento, en kilobytes.
Cached	Memoria	Kilobytes	os.memory.cached	La cantidad de memoria utilizada para almacenar en la caché las E/S basadas en el sistema de archivos, en kilobytes.
DB Cache	Memoria	Bytes	os.memory.db.cache	La cantidad de memoria utilizada para la caché de páginas por proceso de base de datos, incluido tmpfs (shmem), en bytes.
DB Resident Set Size	Memoria	Bytes	os.memory.db.residentSetSize	La cantidad de memoria utilizada para la caché anónima y de intercambio por proceso de base de datos, sin incluir tmpfs (shmem), en bytes.

Contador	Tipo	Unidad	Métrica	Descripción
DB Swap	Memoria	Bytes	os.memory .db.swap	La cantidad de memoria utilizada para el intercambio por proceso de base de datos, en bytes.
Dirty	Memoria	Kilobytes	os.memory.dirty	La cantidad de páginas de memoria en la RAM que se han modificado, pero no escrito, en su bloque de datos relacionado en el almacenamiento, en kilobytes.
Free	Memoria	Kilobytes	os.memory.free	La cantidad de memoria no asignada, en kilobytes.
Huge Pages Free	Memoria	Páginas	os.memory .hugePagesFree	El número de páginas de gran tamaño libres. Las páginas de gran tamaño son una caracterí stica del kernel de Linux.

Contador	Tipo	Unidad	Métrica	Descripción
Huge Pages Rsvd	Memoria	Páginas	os.memory .hugePagesRsvd	El número de páginas de gran tamaño confirmadas.
Huge Pages Size	Memoria	Kilobytes	os.memory .hugePagesSize	El tamaño de cada unidad de páginas de gran tamaño, en kilobytes.
Huge Pages Surp	Memoria	Páginas	os.memory .hugePagesSurp	El número de páginas de gran tamaño sobrantes disponibles con respecto al total.
Huge Pages Total	Memoria	Páginas	os.memory .hugePagesTotal	El número total de páginas enormes.
Inactive	Memoria	Kilobytes	os.memory .inactive	La cantidad de páginas de memoria utilizadas con menor frecuencia, en kilobytes.

Contador	Tipo	Unidad	Métrica	Descripción
Mapped	Memoria	Kilobytes	os.memory .mapped	La cantidad total de contenido del sistema de archivos mapeado a la memoria dentro de un espacio de direcciones de proceso, en kilobytes.
Out of Memory Kill Count	Memoria	Eliminaciones	os.memory .outOfMem oryKillCount	El número de terminaciones de OOM que se produjeron durante el último intervalo de recopilación.
Page Tables	Memoria	Kilobytes	os.memory .pageTables	La cantidad de memoria utilizada por tablas de página, en kilobytes.
Slab	Memoria	Kilobytes	os.memory.slab	La cantidad de estructuras de datos de kernel reutilizables, en kilobytes.
Total	Memoria	Kilobytes	os.memory.total	La cantidad total de memoria, en kilobytes.

Contador	Tipo	Unidad	Métrica	Descripción
Writeback	Memoria	Kilobytes	os.memory .writeback	La cantidad de páginas desfasadas en la RAM que se siguen escribiendo en el almacenamiento de respaldo, en kilobytes.
Guest	Utilización de la CPU	Porcentaje	os.cpuUtilization. guest	El porcentaje de CPU utilizado por programas invitados.
Idle	Utilización de la CPU	Porcentaje	os.cpuUtilization. idle	El porcentaje inactivo de CPU.
Irq	Utilización de la CPU	Porcentaje	os.cpuUtilization. irq	El porcentaje de CPU utilizado por interrupciones de software.
Nice	Utilización de la CPU	Porcentaje	os.cpuUtilization. nice	El porcentaje de CPU utilizado por programas que se ejecutan con la prioridad más baja.
Steal	Utilización de la CPU	Porcentaje	os.cpuUtilization. steal	El porcentaje de CPU utilizado por otras máquinas virtuales.

Contador	Tipo	Unidad	Métrica	Descripción
System	Utilización de la CPU	Porcentaje	os.cpuUtilization.system	El porcentaje de CPU utilizado por el kernel.
Total	Utilización de la CPU	Porcentaje	os.cpuUtilization.total	El porcentaje total de CPU utilizado. Este valor incluye el valor nice.
User	Utilización de la CPU	Porcentaje	os.cpuUtilization.user	El porcentaje de CPU utilizado por programas de usuario.
Wait	Utilización de la CPU	Porcentaje	os.cpuUtilization.wait	El porcentaje de CPU sin utilizar mientras se espera el acceso de E/S.
Aurora Storage Aurora Storage Bytes Rx	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.auroraStorageBytesRx	El número de bytes recibidos para almacenamiento de Aurora por segundo.
Aurora Storage Aurora Storage Bytes Tx	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.auroraStorageBytesTx	El número de bytes cargados para almacenamiento en Aurora por segundo.

Contador	Tipo	Unidad	Métrica	Descripción
Aurora Storage Disk Queue Depth	E/S de disco	Solicitudes	os.diskIO.auroraStorage.diskQueueDepth	La longitud de la cola del disco de almacenamiento de Aurora.
Aurora Storage Read IOs PS	E/S de disco	Solicitudes por segundo	os.diskIO.auroraStorage.readIOsPS	El número de operaciones de lectura por segundo.
Aurora Storage Read Latency	E/S de disco	Milisegundos	os.diskIO.auroraStorage.readLatency	La latencia media de una solicitud de E/S de lectura al almacenamiento de Aurora, en milisegundos.
Aurora Storage Read Throughput	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.readThroughput	La cantidad de rendimiento de red utilizada por las solicitudes al clúster de bases de datos, en bytes por segundo.
Aurora Storage Write IOs PS	E/S de disco	Solicitudes por segundo	os.diskIO.auroraStorage.writeIOsPS	El número de operaciones de escritura por segundo.

Contador	Tipo	Unidad	Métrica	Descripción
Aurora Storage Write Latency	E/S de disco	Milisegundos	os.diskIO.auroraStorage.writeLatency	La latencia media de una solicitud de E/S de escritura al almacenamiento de Aurora, en milisegundos.
Aurora Storage Write Throughput	E/S de disco	Bytes por segundo	os.diskIO.auroraStorage.writeThroughput	La cantidad de rendimiento de red utilizada por las respuestas del clúster de bases de datos, en bytes por segundo.
Rdstemp Avg Queue Len	E/S de disco	Solicitudes	OS.diskio.rdstemp.avgQueuelen	El número de solicitudes que espera en la cola del dispositivo de E/S.
Rdstemp Avg Req Sz	E/S de disco	Solicitudes	os.diskIO.rdstemp.avgReqSz	El número de solicitudes que espera en la cola del dispositivo de E/S.

Contador	Tipo	Unidad	Métrica	Descripción
Rdstemp Await	E/S de disco	Milisegundos	os.diskIO .rdstemp.await	El número de milisegundos necesarios para responder a las solicitudes, incluido el tiempo de cola y el tiempo de servicio.
Rdstemp Read IOs PS	E/S de disco	Solicitudes	os.diskIO .rdstemp. readIOsPS	El número de operaciones de lectura por segundo.
Rdstemp Read KB	E/S de disco	Kilobytes	os.diskIO .rdstemp.readKb	El número total de kilobytes leídos.
Rdstemp Read KB PS	E/S de disco	Kilobytes por segundo	os.diskIO .rdstemp. readKbPS	El número de kilobytes leídos por segundo.
Rdstemp Rrqm PS	E/S de disco	Solicitudes por segundo	os.diskIO .rdstemp.rrqmPS	El número de solicitudes leídas fusionadas en cola por segundo.
Rdstemp TPS	E/S de disco	Transacciones por segundo	os.diskIO .rdstemp.tps	El número de transacciones de E/S por segundo.

Contador	Tipo	Unidad	Métrica	Descripción
Rdstemp Util	E/S de disco	Porcentaje	os.diskIO .rdstemp.util	El porcentaje de tiempo de CPU durante el cual se emitieron las solicitudes.
Rdstemp Write IOs PS	E/S de disco	Solicitudes por segundo	os.diskIO .rdstemp. writeIOsPS	El número de operaciones de escritura por segundo.
Rdstemp Write KB	E/S de disco	Kilobytes	os.diskIO .rdstemp.writeKb	El número total de kilobytes escritos.
Rdstemp Write KB PS	E/S de disco	Kilobytes por segundo	os.diskIO .rdstemp. writeKbPS	El número de kilobytes escritos por segundo.
Rdstemp Wrqm PS	E/S de disco	Solicitudes por segundo	os.diskIO .rdstemp. wrqmPS	El número de solicitudes de escritura fusionadas en cola por segundo.
Blocked	Tareas	Tareas	os.tasks.blocked	El número de tareas que están bloqueadas.
Running	Tareas	Tareas	os.tasks.running	El número de tareas que están en ejecución.

Contador	Tipo	Unidad	Métrica	Descripción
Sleeping	Tareas	Tareas	os.tasks.sleeping	El número de tareas que están inactivas.
Stopped	Tareas	Tareas	os.tasks.stopped	El número de tareas que se han detenido.
Total	Tareas	Tareas	os.tasks.total	El número total de tareas.
Zombie	Tareas	Tareas	os.tasks.zombie	El número de tareas secundarias inactivas con una tarea principal activa.
One	Promedio de carga por minuto	Processes	os.loadAverageMinute.one	El número de procesos que solicitan tiempo de la CPU en el último minuto.
Fifteen	Promedio de carga por minuto	Processes	os.loadAverageMinute.fifteen	El número de procesos que solicitan tiempo de la CPU en los últimos 15 minutos.
Five	Promedio de carga por minuto	Processes	os.loadAverageMinute.five	El número de procesos que solicitan tiempo de la CPU en los últimos 5 minutos.

Contador	Tipo	Unidad	Métrica	Descripción
Cached	Intercambio	Kilobytes	os.swap.cached	La cantidad de memoria de intercambio, en kilobytes, utilizada como memoria caché.
Free	Intercambio	Kilobytes	os.swap.free	La cantidad de memoria de intercambio no asignada, en kilobytes.
In	Intercambio	Kilobytes	os.swap.in	La cantidad de memoria, en kilobytes, intercambiada desde disco.
Out	Intercambio	Kilobytes	os.swap.out	La cantidad de memoria, en kilobytes, intercambiada del disco.
Total	Intercambio	Kilobytes	os.swap.total	La cantidad de memoria de intercambio disponible, en kilobytes.

Contador	Tipo	Unidad	Métrica	Descripción
Max Files	Sistema de archivos	Archivos	os.fileSys.maxFiles	El número máximo de archivos que se pueden crear para el sistema de archivos en todos los volúmenes de almacenamiento.
Used Files	Sistema de archivos	Archivos	os.fileSys.usedFiles	El número de archivos en el sistema de archivos en todos los volúmenes de almacenamiento.
Used File Percent	Sistema de archivos	Archivos	os.fileSys.usedFilePercent	El porcentaje de archivos disponibles en uso en todos los volúmenes de almacenamiento.
Used Percent	Sistema de archivos	Porcentaje	os.fileSys.usedPercent	El porcentaje de espacio en disco del sistema de archivos que está en uso en todos los volúmenes de almacenamiento.

Contador	Tipo	Unidad	Métrica	Descripción
Used	Sistema de archivos	Kilobytes	os.fileSys.used	La cantidad de espacio en disco utilizado por los archivos en el sistema de archivos en todos los volúmenes de almacenamiento, en kilobytes.
Total	File Sys	Kilobytes	os.fileSys.total	El espacio en disco total disponible para el sistema de archivos, en todos los volúmenes de almacenamiento en kilobytes.
Max Files	Sistema de archivos	Archivos	os.fileSys.<volume Name>.maxFiles	El número máximo de archivos que se pueden crear para el volumen de almacenamiento.
Used Files	Sistema de archivos	Archivos	os.fileSys.<volume Name>.usedFiles	El número de archivos del volumen de almacenamiento.

Contador	Tipo	Unidad	Métrica	Descripción
Used File Percent	Sistema de archivos	Archivos	os.fileSys.<volumeName>.usedFilePercent	El porcentaje de archivos disponibles en uso en el volumen de almacenamiento.
Used Percent	Sistema de archivos	Porcentaje	os.fileSys.<volumeName>.usedPercent	El porcentaje de espacio en disco de volumen de almacenamiento que está en uso.
Used	Sistema de archivos	Kilobytes	os.fileSys.<volumeName>.used	La cantidad de espacio en disco utilizado por los archivos en el volumen de almacenamiento, en kilobytes.
Total	File Sys	Kilobytes	os.fileSys.<volumeName>.total	El espacio en disco total en el volumen de almacenamiento, en kilobytes.
Rx	Network	Bytes por segundo	os.network.rx	El número de bytes recibidos por segundo.
Tx	Network	Bytes por segundo	os.network.tx	El número de bytes cargados por segundo.

Contador	Tipo	Unidad	Métrica	Descripción
Acu Utilization	General	Porcentaje	os.general.acuUtilization	El porcentaje de la capacidad actual fuera de la capacidad máxima configurada.
Max Configured Acu	General	ACU	os.general.maxConfiguredAcu	La capacidad máxima configurada por el usuario, en unidades de capacidad de Aurora (ACU).
Min Configured Acu	General	ACU	os.general.minConfiguredAcu	La capacidad mínima configurada por el usuario, en ACU.
Num VCPUs	General	vCPU	os.general.numVCPUs	El número de CPU virtuales (vCPU) para la instancia de base de datos.
Serverless Database Capacity	General	ACU	os.general.serverlessDatabaseCapacity	La capacidad actual de la instancia, en ACU.

Contadores de Información de rendimiento para Aurora MySQL

Los siguientes contadores de base de datos están disponibles con Performance Insights para Aurora MySQL.

Temas

- Contadores nativos para Aurora MySQL
- Contadores no nativos para Aurora MySQL

Contadores nativos para Aurora MySQL

Las métricas nativas las define el motor de base de datos y no Amazon Aurora. Puede encontrar las definiciones de estas métricas nativas en Server Status Variables (Variables de estado de servidor) en la documentación de MySQL.

Contador	Tipo	Unidad	Métrica
Com_analyze	SQL	Consultas por segundo	db.SQL.Com_analyze
Com_optimize	SQL	Consultas por segundo	db.SQL.Com_optimize
Com_select	SQL	Consultas por segundo	db.SQL.Com_select
Innodb_rows_deleted	SQL	Filas por segundo	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	Filas por segundo	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	Filas por segundo	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	Filas por segundo	db.SQL.Innodb_rows_updated

Contador	Tipo	Unidad	Métrica
Consultas	SQL	Consultas por segundo	db.SQL.Queries
Preguntas	SQL	Consultas por segundo	db.SQL.Questions
Select_full_join	SQL	Consultas por segundo	db.SQL.Select_full_join
Select_full_range_join	SQL	Consultas por segundo	db.SQL.Select_full_range_join
Select_range	SQL	Consultas por segundo	db.SQL.Select_range
Select_range_check	SQL	Consultas por segundo	db.SQL.Select_range_check
Select_scan	SQL	Consultas por segundo	db.SQL.Select_scan
Slow_queries	SQL	Consultas por segundo	db.SQL.Slow_queries
Sort_merge_passes	SQL	Consultas por segundo	db.SQL.Sort_merge_passes

Contador	Tipo	Unidad	Métrica
Sort_range	SQL	Consultas por segundo	db.SQL.Sort_range
Sort_rows	SQL	Consultas por segundo	db.SQL.Sort_rows
Sort_scan	SQL	Consultas por segundo	db.SQL.Sort_scan
Total_query_time	SQL	Milisegundos	db.SQL.Total_query_time
Table_locks_immediate	Bloqueos	Solicitudes por segundo	db.Locks.Table_locks_immediate
Table_locks_waited	Bloqueos	Solicitudes por segundo	db.Locks.Table_locks_waited
Innodb_row_lock_time	Bloqueos	Milisegundos (promedio)	db.Locks.Innodb_row_lock_time
Aborted_clients	Usuarios	Conexiones	db.Users.Aborted_clients
Aborted_connects	Usuarios	Conexiones	db.Users.Aborted_connects
Connections	Usuarios	Connections	db.Users.Connections

Contador	Tipo	Unidad	Métrica
External_threads_connected	Usuarios	Connections	db.Users.External_threads_connected
max_connections	Usuarios	Connections	db.Users.max_connections
Threads_connected	Usuarios	Connections	db.Users.Threads_connected
Threads_created	Usuarios	Conexiones	db.Users.Threads_created
Threads_running	Usuarios	Conexiones	db.Users.Threads_running
Created_tmp_disk_tables	Temp	Tablas por segundo	db.Temp.Created_tmp_disk_tables
Created_tmp_tables	Temp	Tablas por segundo	db.Temp.Created_tmp_tables
Innodb_buffer_pool_pages_data	Caché	Páginas	db.Cache.Innodb_buffer_pool_pages_data
Innodb_buffer_pool_pages_total	Caché	Páginas	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Caché	Páginas por segundo	db.Cache.Innodb_buffer_pool_read_requests
Innodb_buffer_pool_reads	Caché	Páginas por segundo	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Caché	Tablas	db.Cache.Opened_tables
Opened_table_definitions	Caché	Tablas	db.Cache.Opened_table_definitions

Contador	Tipo	Unidad	Métrica
Qcache_hits	Caché	Consultas	db.Cache.Qcache_hits

Contadores no nativos para Aurora MySQL

Las métricas de contadores no nativos se definen mediante Amazon RDS. Una métrica no nativa puede ser una métrica que obtiene con una consulta concreta. Una métrica no nativa también puede ser una métrica derivada, en la que se utilicen dos o más contadores nativos en cálculos para proporciones, aciertos o latencias.

Contador	Tipo	Unidad	Métrica	Descripción	Definición
active_transactions	Transa ones	db.Trar ctions.a tive_tra saction	Las transacci ones activas totales.	`SELECT COUNT(1) AS active_tr ansactions FROM INFOR MATION_SC HEMA.INNO DB_TRX`	
innodb_buffer_pool _hit_rate	Caché	db.Cac innoDB ffer_poo _hit_rat	El porcentaj e de lecturas que InnoDB podría cumplir del grupo de búferes.	`100 * innodb_bu ffer_pool _read_req uests / (innodb_ buffer_po ol_read_r equests + innodb_bu ffer_pool _reads)`	
innodb_buffer_pool _hits	Caché	Página por segund	db.Cache. innoDB_bu ffer_pool _hits	El número de lecturas que InnoDB podría cumplir del grupo de búferes.	`innodb_bu ffer_pool _read_requests - innodb_bu`

Contador	Tipo	Unidad	Métrica	Descripción	Definición
					`ffer_pool _reads`
innodb_buffer_pool _usage	Caché	Porcen e	db.Cache. innoDB_bu ffer_pool _usage	El porcentaje del grupo de búferes de InnoDB que contiene datos (páginas). **Note** Al usar tablas comprimid as, este valor puede variar. Para obtener más información, consulte la información acerca de `Innodb_bu ffer_pool _pages_da ta` y `Innodb_bu ffer_pool _pages_to tal` en Server Status Variables en la documenta ción de MySQL.	`Innodb_bu ffer_pool _pages_da ta / Innodb_bu ffer_pool _pages_total * 100.0`

Contador	Tipo	Unidad	Métrica	Descripción	Definición
innodb_deadlocks	Bloque	db.Lock innodb_ adlocks	El número total de interbloqueos.	`SELECT COUNT AS innodb_de adlocks FROM INFORMATION_SC HEMA.INNO DB_METRICS WHERE NAME= 'lock_dea dlocks'`	
innodb_lock_timeouts	Bloque	db.Lock innodb_ ck_time ts	El número total de interbloqueos que agotaron el tiempo de espera.	`SELECT COUNT AS innodb_lo ck_timeouts FROM INFOR MATION_SC HEMA.INNO DB_METRICS WHERE NAME= 'lock_tim eouts'`	
innodb_row_lock_waits	Bloque	db.Lock innodb_ w_lock_ its	El número total de bloqueos que resultaron en una espera.	`SELECT COUNT AS innodb_ro w_lock_waits FROM INFOR MATION_SC HEMA.INNO DB_METRICS WHERE NAME ='lock_ro w_lock_waits'`	

Contador	Tipo	Unidad	Métrica	Descripción	Definición
innodb_rows_changed	SQL	db.SQL.Innodb_rows_changed	Las operaciones de filas de InnoDB totales.	db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated	
query_cache_hit_rate	Caché	Porcentaje	db.Cache.query_cache_hit_rate	La proporción de aciertos de caché de conjunto de resultados MySQL (caché de consultas).	Qcache_hits / (QCache_hits + Com_select) * 100
temp_disk_tables_percent	Temp	db.Temp.Created_tmp_disk_tables_percent	El porcentaje de tablas temporales que el servidor crea en el disco al ejecutar instrucciones.	(db.Temp.Created_tmp_disk_tables / db.Temp.Created_tmp_tables) * 100	

Contador	Tipo	Unidad	Métrica	Descripción	Definición
trx_rseg_history_len	Transa ones	Ningun	db.Transactions.trx_rseg_history_len	Una lista de las páginas de registro de deshacer de las transacciones confirmadas que mantiene el sistema de transacciones InnoDB para implementar el control de concurrencia de varias versiones. Para obtener más información acerca de los detalles de los registros de deshacer, consultehttps://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html en la documentación de MySQL.	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

Contadores de Información sobre rendimiento para Aurora PostgreSQL

Los siguientes contadores de base de datos están disponibles con Performance Insights para Aurora PostgreSQL.

Temas

- Contadores nativos para Aurora PostgreSQL
- Contadores no nativos para Aurora PostgreSQL

Contadores nativos para Aurora PostgreSQL

Las métricas nativas las define el motor de base de datos y no Amazon Aurora. Puede encontrar definiciones para estas métricas en Ver estadísticas en la documentación de PostgreSQL.

Contador	Tipo	Unidad	Métrica
tup_deleted	SQL	Tuplas por segundo	db.SQL.tup_deleted
tup_fetched	SQL	Tuplas por segundo	db.SQL.tup_fetched
tup_inserted	SQL	Tuplas por segundo	db.SQL.tup_inserted
tup_returned	SQL	Tuplas por segundo	db.SQL.tup_returned
tup_updated	SQL	Tuplas por segundo	db.SQL.tup_updated
blks_hit	Caché	Bloques por segundo	db.Cache.blks_hit
buffers_alloc	Caché	Bloques por segundo	db.Cache.buffers_alloc
buffers_checkpoint	Punto de comprobación	Bloques por segundo	db.Checkpoint.buffers_checkpoint
checkpoints_req	Punto de comprobación	Puntos de comprobación por minuto	db.Checkpoint.checkpoints_req
checkpoint_sync_time	Punto de comprobación	Milisegundos por punto de comprobación	db.Checkpoint.checkpoint_sync_time
checkpoints_timed	Punto de comprobación	Puntos de comprobación por minuto	db.Checkpoint.checkpoints_timed
checkpoint_write_time	Punto de comprobación	Milisegundos por punto de comprobación	db.Checkpoint.checkpoint_write_time

Contador	Tipo	Unidad	Métrica
maxwritten_clean	Punto de comprobación	Paradas de eliminación de Bgwriter por minuto	db.Checkpoint.maxwritten_clean
deadlocks	Simultaneidad	Interbloqueos por minuto	db.Concurrency.deadlocks
blk_read_time	I/O	Milisegundos	db.IO.blk_read_time
blks_read	I/O	Bloques por segundo	db.IO.blks_read
buffers_backend	I/O	Bloques por segundo	db.IO.buffers_backend
buffers_backend_fsync	I/O	Bloques por segundo	db.IO.buffers_backend_fsync
buffers_clean	I/O	Bloques por segundo	db.IO.buffers_clean
temp_bytes	Temp	Bytes por segundo	db.Temp.temp_bytes
temp_files	Temp	Archivos por minuto	db.Temp.temp_files
xact_commit	Transacciones	Confirmaciones por segundo	db.Transactions.xact_commit
xact_rollback	Transacciones	Restauraciones por segundo	db.Transactions.xact_rollback
numbackends	Usuario	Conexiones	db.User.numbackends
archived_count	WAL	Archivos por minuto	db.WAL.archived_count

Contadores no nativos para Aurora PostgreSQL

Las métricas de contadores no nativos se definen mediante Amazon Aurora. Una métrica no nativa puede ser una métrica que obtiene con una consulta concreta. Una métrica no nativa también puede ser una métrica derivada, en la que se utilicen dos o más contadores nativos en cálculos para proporciones, aciertos o latencias.

Contador	Tipo	Unidad	Métrica	Descripción	Definición
checkpoint_sync_latency	Punto de comprobación	Milisegundos	db.Checkpoint.checkpoint_sync_latency	La cantidad de tiempo invertido en la parte del procesamiento del punto de comprobación en la que los archivos se han sincronizado en el disco.	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	Punto de comprobación	Milisegundos	db.Checkpoint.checkpoint_write_latency	La cantidad de tiempo invertido en la parte del procesamiento del punto de comprobación en la que los archivos se han escrito en el disco.	`checkpoint_write_time / (checkpoints_timed + checkpoints_req)`
local_blks_read	E/S	Bloques	db.IO.local_blks_read	Número total de bloques locales leídos.	No aplicable
local_blk_read_time	E/S	Milisegundos	db.IO.local_blk_read_time	Si `track_io_timing` está activado, registra el tiempo total dedicado a leer bloques de archivos de datos locales, en milisegundos; de lo contrario,	No aplicable

Contador	Tipo	Unidad	Métrica	Descripción	Definición
				el valor es cero. Para obtener más información, consulta track_io_timing.	
num_blocked_sessions	Bloqueos	db.Locks.num_blocked_sessions	El número de sesiones bloqueadas.	–	
orcache_blks_hit	E/S	Consultas	db.IO.orcache_blks_hit	Número total de aciertos de bloques compartidos desde la caché de lecturas optimizadas.	No aplicable
orcache_blk_read_time	E/S	Milisegundos	db.IO.orcache_blk_read_time	Si `track_io_timing` está activado, registra el tiempo total dedicado a leer bloques de archivos de datos desde la caché de lecturas optimizadas (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.	No aplicable

Contador	Tipo	Unidad	Métrica	Descripción	Definición
read_latency	E/S	Milisegundos	db.IO.read_latency	El tiempo invertido leyendo bloques de archivos de datos por backends en esta instancia.	`blk_read_time / blks_read`
storage_blks_read	E/S	Bloques	db.IO.storage_blks_read	Número total de bloques compartidos leídos desde el almacenamiento de Aurora.	No aplicable
storage_blk_read_time	E/S	Milisegundos	db.IO.storage_blk_read_time	Si `track_io_timing` está activado, registra el tiempo total dedicado a leer bloques de archivos de datos desde el almacenamiento de Aurora (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.	No aplicable
num_blocked_sessions	Bloqueos	db.Locks.num_blocked_sessions	El número de sesiones bloqueadas.	–	

Contador	Tipo	Unidad	Métrica	Descripción	Definición
active_count	Estado	Sesiones	db.state.active_count	El número de sesiones en el estado `active`.	No aplicable
idle_count	Estado	Sesiones	db.state.idle_count	El número de sesiones en el estado `idle`.	No aplicable
idle_in_transaction_aborted_count	Estado	Sesiones	db.state.idle_in_transaction_aborted_count	El número de sesiones en el estado `idle in transaction (aborted)` .	No aplicable
idle_in_transaction_count	Estado	Sesiones	db.state.idle_in_transaction_count	El número de sesiones en el estado `idle in transaction` .	No aplicable
idle_in_transaction_max_time	Estado	Segundos	db.state.idle_in_transaction_max_time	La duración de la transacción de mayor duración en el estado `idle in transaction` , en segundos.	No aplicable
logical_reads	SQL	Bloques	db.SQL.logical_reads	El número total de bloques marcados y leídos.	`blks_hit + blks_read`
queries_started	SQL	Consultas	db.SQL.queries	El número de consultas iniciadas.	No aplicable
queries_finished	SQL	Consultas	db.SQL.queries	El número de consultas finalizadas.	No aplicable

Contador	Tipo	Unidad	Métrica	Descripción	Definición
total_query_time	SQL	Milisegundos	db.SQL.total_query_time	El tiempo total empleado en ejecutar instrucciones, en milisegundos.	No aplicable
active_transactions	Transacciones	Transacciones	db.Transactions.active_transactions	El número de transacciones activas.	No aplicable
blocked_transactions	Transacciones	Transacciones	db.Transactions.blocked_transactions	El número de transacciones bloqueadas.	No aplicable
commit_latency	Transacciones	Microsegundos	db.Transactions.commit_latency	Duración promedio de las operaciones de confirmación.	`db.Transactions.duration_commits / db.Transactions.xact_commit`
duration_commits	Transacciones	Milisegundos	db.Transactions.duration_commits	El tiempo total de transacción empleado en el último minuto, en milisegundos.	No aplicable
max_used_xact_ids	Transacciones	Transacciones	db.Transactions.max_used_xact_ids	El número de transacciones que no se han vaciado.	No aplicable

Contador	Tipo	Unidad	Métrica	Descripción	Definición
oldest_inactive_logical_replication_slot_xid_age	Transacciones	Longitud	db.Transactions.oldest_inactive_logical_replication_slot_xid_age	La antigüedad de la transacción más antigua en una ranura de replicación lógica inactiva.	No aplicable
oldest_active_logical_replication_slot_xid_age	Transacciones	Longitud	db.Transactions.oldest_active_logical_replication_slot_xid_age	La antigüedad de la transacción más antigua en una ranura de replicación lógica activa.	No aplicable
oldest_reader_feedback_xid_age	Transacciones	Longitud	db.Transactions.oldest_reader_feedback_xid_age	La antigüedad de la transacción más antigua de una transacción de larga duración en una instancia de lector de Aurora o una instancia de lector de base de datos global de Aurora.	No aplicable
oldest_prepared_transaction_xid_age	Transacciones	Longitud	db.Transactions.oldest_prepared_transaction_xid_age	La antigüedad de la transacción preparada más antigua.	No aplicable

Contador	Tipo	Unidad	Métrica	Descripción	Definición
oldest_running_transaction_xid_age	Transacciones	Longitud	db.Transactions.oldest_running_transaction_xid_age	La antigüedad de la transacción en ejecución más antigua.	No aplicable
max_connections	Usuarios	Usuarios	db.User.max_connections	El número máximo de conexiones permitidas para una base de datos, según lo configurado en el parámetro `max_connections` .	No aplicable
total_auth_attempts	Usuarios	Usuarios	db.User.total_auth_attempts	El número de intentos de conexión a esta instancia.	No aplicable
archive_failed_count	WAL	Archivos por minuto	db.WAL.archive_failed_count	El número de intentos fallidos de archivado de archivos WAL, en archivos por minuto.	No aplicable

Estadísticas de SQL para Performance Insights

Las estadísticas de SQL son métricas relacionadas con el rendimiento de las consultas SQL que recopila Performance Insights. Performance Insights recopila estadísticas de cada segundo que se ejecuta una consulta y para cada llamada SQL. Las estadísticas de SQL son un promedio del intervalo de tiempo seleccionado.

Un resumen de SQL es un conjunto de todas las consultas que tienen un patrón dado, pero no necesariamente tienen los mismos valores literales. El resumen reemplaza los valores literales por un signo de interrogación. Por ejemplo, `SELECT * FROM emp WHERE lname= ?`. Este resumen podría incluir las siguientes consultas secundarias:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Todos los motores admiten estadísticas de SQL para las consultas de resumen.

Para obtener información sobre la compatibilidad de esta característica por región, motor de base de datos y clase de instancia, consulte Compatibilidad del motor de la base de datos, la región y la clase de instancia de Amazon Aurora con características de Información de rendimiento.

Temas

- Estadísticas de SQL de Aurora MySQL
- Estadísticas de SQL de Aurora PostgreSQL

Estadísticas de SQL de Aurora MySQL

Aurora MySQL recopila estadísticas de SQL solo en el nivel de resumen. No se muestran estadísticas en el nivel de instrucción.

Temas

- Estadísticas de resumen para Aurora MySQL
- Estadísticas por segundo de Aurora MySQL
- Estadísticas por llamada de Aurora MySQL
- Estadísticas principales para Aurora MySQL

Estadísticas de resumen para Aurora MySQL

Información sobre rendimiento recopila estadísticas de resumen SQL de la tabla `events_statements_summary_by_digest`. La base de datos administra la tabla `events_statements_summary_by_digest`.

Esta tabla no tiene una política de expulsión. Cuando la tabla está llena, se muestra el siguiente mensaje en la:Consola de administración de AWS

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
```

```
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

En esta situación, Aurora MySQL no lleva a cabo un seguimiento de las consultas SQL. Para solucionar este problema, la Información sobre rendimiento trunca automáticamente la tabla de resumen cuando se cumplen estas dos condiciones:

- La tabla está llena.
- La Información sobre rendimiento administra automáticamente el Esquema de rendimiento.

  Para la administración automática, el parámetro `performance_schema` se debe establecer en `0` y la Fuente no se debe establecer en `user`. Si Información sobre rendimiento no administra el esquema de rendimiento automáticamente, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

En la AWS CLI, compruebe el fuente de un valor de parámetro ejecutando el comando describe-db-parameters.

Estadísticas por segundo de Aurora MySQL

Las siguientes estadísticas de SQL están disponibles para los clústeres de base de datos de Aurora MySQL.

Métrica	Unidad
db.sql_tokenized.stats.count_star_per_sec	Llamadas por segundo
db.sql_tokenized.stats.sum_timer_wait_per_sec	Latencia media por segundo (en milisegundos)
db.sql_tokenized.stats.sum_select_full_join_per_sec	Unión completa de seleccionar por segundo
db.sql_tokenized.stats.sum_select_range_check_per_sec	Control de rango de seleccionar por segundo
db.sql_tokenized.stats.sum_select_scan_per_sec	Escaneo de seleccionar por segundo

Métrica	Unidad
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	Pases de fusión de clasificación por segundo
db.sql_tokenized.stats.sum_sort_scan_per_sec	Escaneos de clasificación por segundo
db.sql_tokenized.stats.sum_sort_range_per_sec	Rangos de clasificación por segundo
db.sql_tokenized.stats.sum_sort_rows_per_sec	Filas de clasificación por segundo
db.sql_tokenized.stats.sum_rows_affected_per_sec	Filas afectadas por segundo
db.sql_tokenized.stats.sum_rows_examined_per_sec	Filas examinadas por segundo
db.sql_tokenized.stats.sum_rows_sent_per_sec	Filas enviadas por segundo
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	Tablas de disco temporales creadas por segundo
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	Tablas temporales creadas por segundo
db.sql_tokenized.stats.sum_lock_time_per_sec	Tiempo de bloqueo por segundo (en milisegundos)

Estadísticas por llamada de Aurora MySQL

Las siguientes métricas ofrecen estadísticas por llamada para una instrucción SQL.

Métrica	Unidad
db.sql_tokenized.stats.sum_timer_wait_per_call	Latencia media por llamada (en milisegundos)
db.sql_tokenized.stats.sum_select_full_join_per_call	Uniones completas de seleccionar por llamada

Métrica	Unidad
db.sql_tokenized.stats.sum_select_range_check_per_call	Control de rango de s por llamada
db.sql_tokenized.stats.sum_select_scan_per_call	Escaneos de seleccionar por llamada
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	Pases de fusión de clasificación por llamada
db.sql_tokenized.stats.sum_sort_scan_per_call	Escaneos de clasificación por llamada
db.sql_tokenized.stats.sum_sort_range_per_call	Rangos de clasificación por llamada
db.sql_tokenized.stats.sum_sort_rows_per_call	Filas de clasificación por llamada
db.sql_tokenized.stats.sum_rows_affected_per_call	Filas afectadas por llamada
db.sql_tokenized.stats.sum_rows_examined_per_call	Filas examinadas por llamada
db.sql_tokenized.stats.sum_rows_sent_per_call	Filas enviadas por llamada
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	Tablas de disco temporales creadas por llamada
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	Tablas temporales creadas por llamada
db.sql_tokenized.stats.sum_lock_time_per_call	Tiempo de bloqueo por llamada (en milisegundos)

Estadísticas principales para Aurora MySQL

Las siguientes estadísticas de SQL están disponibles para los clústeres de base de datos de Aurora MySQL.

Métrica	Unidad
db.sql_tokenized.stats.count_star	Calls
db.sql_tokenized.stats.sum_timer_wait	Tiempo de espera (en ms)
db.sql_tokenized.stats.sum_select_full_join	Seleccionar unión completa
db.sql_tokenized.stats.sum_select_range_check	Seleccionar comprobaciones de intervalos
db.sql_tokenized.stats.sum_select_scan	Seleccionar análisis
db.sql_tokenized.stats.sum_sort_merge_passes	Ordenar pasadas de combinación
db.sql_tokenized.stats.sum_sort_scan	Ordenar análisis
db.sql_tokenized.stats.sum_sort_range	Ordenar intervalos
db.sql_tokenized.stats.sum_sort_rows	Ordenar filas
db.sql_tokenized.stats.sum_rows_affected	Filas afectadas
db.sql_tokenized.stats.sum_rows_examined	Filas examinadas
db.sql_tokenized.stats.sum_rows_sent	Filas enviadas
db.sql_tokenized.stats.sum_created_tmp_disk_tables	Tablas de disco temporales creadas
db.sql_tokenized.stats.sum_created_tmp_tables	Tablas temporales creadas
db.sql_tokenized.stats.sum_lock_time	Tiempo de bloqueo (en ms)

Estadísticas de SQL de Aurora PostgreSQL

Para cada llamada SQL y para cada segundo que se ejecuta una consulta, Performance Insights recopila estadísticas SQL. Todos los motores Aurora recopilan estadísticas únicamente en el nivel de resumen.

A continuación, encontrará información sobre las estadísticas de resumen de Aurora PostgreSQL.

Temas

- Estadísticas de resumen de Aurora PostgreSQL:
- Estadísticas de resumen por segundo de Aurora PostgreSQL
- Estadísticas de resumen por llamada de Aurora PostgreSQL
- Estadísticas principales para Aurora PostgreSQL

Estadísticas de resumen de Aurora PostgreSQL:

Para ver las estadísticas de resumen de SQL, debe cargar la biblioteca de `pg_stat_statements`. La biblioteca se carga de forma predeterminada para los clústeres de base de datos de Aurora PostgreSQL compatibles con PostgreSQL 10. Esta biblioteca se habilita manualmente para los clústeres de base de datos de Aurora PostgreSQL compatibles con PostgreSQL 9.6. Para habilitarlo de forma manual, añada `pg_stat_statements` a `shared_preload_libraries` en el grupo de parámetros de base de datos asociado a la instancia de base de datos. Después, reinicie la instancia de base de datos. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.

> **Note**
>
> Con Información sobre rendimiento solo se pueden recopilar estadísticas para consultas en `pg_stat_activity` que no estén truncadas. De forma predeterminada, las bases de datos de PostgreSQL truncan consultas de más de 1024 bytes. Para aumentar el volumen de la consulta, cambie el parámetro `track_activity_query_size` en el grupo de parámetros de base de datos asociado con la instancia de base de datos. Cuando se cambia este parámetro, se requiere un reinicio de la instancia de base de datos.

Estadísticas de resumen por segundo de Aurora PostgreSQL

Las siguientes estadísticas de resumen de SQL se encuentran disponibles para las instancias de base de datos de Aurora PostgreSLQ.

Métrica	Unidad
db.sql_tokenized.stats.calls_per_sec	Llamadas por segundo
db.sql_tokenized.stats.rows_per_sec	Filas por segundo

Métrica	Unidad
db.sql_tokenized.stats.total_time_per_sec	Media de ejecuciones activas (AAE) por segundo
db.sql_tokenized.stats.shared_blks_hit_per_sec	Aciertos en bloque por segundo
db.sql_tokenized.stats.shared_blks_read_per_sec	Lecturas en bloque por segundo
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	Bloques ensuciados por segundo
db.sql_tokenized.stats.shared_blks_written_per_sec	Escrituras en bloque por segundo
db.sql_tokenized.stats.local_blks_hit_per_sec	Aciertos en bloque locales por segundo
db.sql_tokenized.stats.local_blks_read_per_sec	Lecturas en bloque locales por segundo
db.sql_tokenized.stats.local_blks_dirtied_per_sec	Suciedades en bloque locales por segundo
db.sql_tokenized.stats.local_blks_written_per_sec	Escrituras en bloque locales por segundo
db.sql_tokenized.stats.temp_blks_written_per_sec	Escrituras en temporales por segundo
db.sql_tokenized.stats.temp_blks_read_per_sec	Lecturas temporales por segundo
db.sql_tokenized.stats.blk_read_time_per_sec	Media de lecturas actuales por segundo
db.sql_tokenized.stats.blk_write_time_per_sec	Media de escrituras actuales por segundo

Estadísticas de resumen por llamada de Aurora PostgreSQL

Las siguientes métricas ofrecen estadísticas por llamada para una instrucción SQL.

Métrica	Unidad
db.sql_tokenized.stats.rows_per_call	Filas por llamada
db.sql_tokenized.stats.avg_latency_per_call	Latencia media por llamada (en milisegundos)
db.sql_tokenized.stats.shared_blks_hit_per_call	Aciertos en bloque por llamada
db.sql_tokenized.stats.shared_blks_read_per_call	Lecturas en bloque por llamada
db.sql_tokenized.stats.shared_blks_written_per_call	Escrituras en bloque por llamada
db.sql_tokenized.stats.shared_blks_dirtied_per_call	Bloques ensuciados por llamada
db.sql_tokenized.stats.local_blks_hit_per_call	Aciertos en bloque locales por llamada
db.sql_tokenized.stats.local_blks_read_per_call	Lecturas en bloques locales por llamada
db.sql_tokenized.stats.local_blks_dirtied_per_call	Suciedades en bloque local por llamada
db.sql_tokenized.stats.local_blks_written_per_call	Escrituras en bloque local por llamada
db.sql_tokenized.stats.temp_blks_written_per_call	Escrituras en bloque temporal por llamada
db.sql_tokenized.stats.temp_blks_read_per_call	Lecturas en bloque temporal por llamada
db.sql_tokenized.stats.blk_read_time_per_call	Tiempo de lectura por llamada (en milisegundos)
db.sql_tokenized.stats.blk_write_time_per_call	Tiempo de escritura por llamada (en milisegundos)

Estadísticas principales para Aurora PostgreSQL

Las siguientes estadísticas de SQL se encuentran disponibles para las instancias de base de datos de Aurora PostgreSQL.

Métrica	Unidad
db.sql_tokenized.stats.calls	Calls
db.sql_tokenized.stats.rows	Filas
db.sql_tokenized.stats.total_time	Tiempo total (en ms)
db.sql_tokenized.stats.shared_blks_hit	Aciertos en bloque
db.sql_tokenized.stats.shared_blks_read	Lecturas en bloque
db.sql_tokenized.stats.shared_blks_dirtied	Bloques ensuciados
db.sql_tokenized.stats.shared_blks_written	Escrituras en bloque
db.sql_tokenized.stats.local_blks_hit	Aciertos en bloque locales
db.sql_tokenized.stats.local_blks_read	Lecturas en bloque locales
db.sql_tokenized.stats.local_blks_dirtied	Bloques locales ensuciados
db.sql_tokenized.stats.local_blks_written	Escrituras en bloque locales
db.sql_tokenized.stats.temp_blks_written	Escrituras temporales
db.sql_tokenized.stats.temp_blks_read	Lecturas temporales
db.sql_tokenized.stats.blk_read_time	Promedio de lecturas simultáneas (en ms)
db.sql_tokenized.stats.blk_write_time	Promedio de escrituras simultáneas (en ms)

Para obtener más información acerca de estas métricas, consulte pg_stat_statements en la documentación de PostgreSQL.

Métricas del sistema operativo en Supervisión mejorada

Amazon Aurora proporciona métricas en tiempo real para el sistema operativo (SO) en el que se ejecuta el clúster de bases de datos. Aurora entrega las métricas de la Supervisión mejorada a su cuenta de registros de Amazon Cloudwatch. Las tablas siguientes incluyen las métricas de SO disponibles al usar registros de Amazon Cloudwatch.

Temas

- Métricas del sistema operativo para Aurora

Métricas del sistema operativo para Aurora

Grupo	Métrica	Nombre de la consola	Descripción
`General`	`engine`	No aplicable	El motor de base de datos para la instancia de base de datos.
	`instanceID`	No aplicable	El identificador de instancias de base de datos.
	`instanceResourceID`	No aplicable	Un identificador inmutable para la instancia de base de datos que es exclusivo de una región de AWS, también utilizado como identificador de secuencia de registro.
	`numVCPUs`	No aplicable	El número de CPU virtuales para la instancia de base de datos.
	`timestamp`	No aplicable	La hora a la que se tomó la métrica.
	`uptime`	No aplicable	La cantidad de tiempo que ha estado activa la instancia de base de datos.
	`version`	No aplicable	La versión del formato JSON del flujo de la métrica del SO.

Grupo	Métrica	Nombre de la consola	Descripción
`cpuUtilization`	`guest`	CPU Guest	El porcentaje de CPU utilizado por programas invitados.
	`idle`	CPU Idle	El porcentaje inactivo de CPU.
	`irq`	CPU IRQ	El porcentaje de CPU utilizado por interrupciones de software.
	`nice`	CPU Nice	El porcentaje de CPU utilizado por programas que se ejecutan con la prioridad más baja.
	`steal`	CPU Steal	El porcentaje de CPU utilizado por otras máquinas virtuales.
	`system`	CPU System	El porcentaje de CPU utilizado por el kernel.
	`total`	CPU Total	El porcentaje total de CPU utilizado. Este valor incluye el valor `nice`.
	`user`	CPU User	El porcentaje de CPU utilizado por programas de usuario.
	`wait`	CPU Wait	El porcentaje de CPU sin utilizar mientras se espera el acceso de E/S.
`diskIO`	`avgQueueLen`	Avg Queue Size	El número de solicitudes que espera en la cola del dispositivo de E/S.
	`avgReqSz`	Ave Request Size	El tamaño promedio de la solicitudes, en kilobytes.
	`await`	Disk I/O Await	El número de milisegundos necesarios para responder a las solicitudes, incluido el tiempo de cola y el tiempo de servicio.

Grupo	Métrica	Nombre de la consola	Descripción
	`device`	No aplicable	El identificador del dispositivo de disco en uso.
	`readIOsPS`	Read IO/s	El número de operaciones de lectura por segundo.
	`readKb`	Read Total	El número total de kilobytes leídos.
	`readKbPS`	Read Kb/s	El número de kilobytes leídos por segundo.
	`readLatency`	Read Latency	El tiempo transcurrido entre el envío de una solicitud de E/S de lectura y su finalización, en milisegundos. Esta métrica solo está disponible para Amazon Aurora.
	`readThroughput`	Read Throughput	La cantidad de rendimiento de red utilizada por las solicitudes al clúster de bases de datos, en bytes por segundo. Esta métrica solo está disponible para Amazon Aurora.
	`rrqmPS`	Rrqms	El número de solicitudes leídas fusionadas en cola por segundo.
	`tps`	TPS	El número de transacciones de E/S por segundo.
	`util`	Disk I/O Util	El porcentaje de tiempo de CPU durante el cual se emitieron las solicitudes.
	`writeIOsPS`	Write IO/s	El número de operaciones de escritura por segundo.
	`writeKb`	Write Total	El número total de kilobytes escritos.
	`writeKbPS`	Write Kb/s	El número de kilobytes escritos por segundo.

Grupo	Métrica	Nombre de la consola	Descripción
	`writeLatency`	Write Latency	Tiempo medio transcurrido entre el envío de una solicitud de E/S de escritura y su finalización, en milisegundos. Esta métrica solo está disponible para Amazon Aurora.
	`writeThroughput`	Write Throughput	La cantidad de rendimiento de red utilizada por las respuestas del clúster de bases de datos, en bytes por segundo. Esta métrica solo está disponible para Amazon Aurora.
	`wrqmPS`	Wrqms	El número de solicitudes de escritura fusionadas en cola por segundo.
`mountPoint`	No aplicable	La ruta al sistema de archivos.	
`fileSys`	`maxFiles`	Max Inodes	El número máximo de archivos que se pueden crear para el sistema de archivos.
	`mountPoint`	No aplicable	La ruta al sistema de archivos. Cuando esto es `/rdsdbdata*` , representa la suma de todos los volúmenes de almacenamiento.
	`name`	No aplicable	El nombre del sistema de archivos.
	`total`	Total Filesystem	La cantidad total de espacio en disco disponible para el sistema de archivos, en kilobytes.
	`used`	Used Filesystem	La cantidad de espacio en disco utilizado por los archivos en el sistema de archivos, en kilobytes.

Grupo	Métrica	Nombre de la consola	Descripción
	`usedFilePercent`	Used Inodes	El porcentaje de archivos disponibles en uso.
	`usedFiles`	Used%	El número de archivos en el sistema de archivos.
	`usedPercent`	Used Filesystem	El porcentaje de espacio en disco del sistema de archivos que está en uso.
`loadAverageMinute`	`fifteen`	Load Avg 15 min	El número de procesos que solicitan tiempo de la CPU en los últimos 15 minutos.
	`five`	Load Avg 5 min	El número de procesos que solicitan tiempo de la CPU en los últimos 5 minutos.
	`one`	Load Avg 1 min	El número de procesos que solicitan tiempo de la CPU en el último minuto.
`memory`	`active`	Active Memory	La cantidad de memoria asignada, en kilobytes.
	`buffers`	Buffered Memory	La cantidad de memoria utilizada para almacenar en búfer solicitudes de E/S antes de escribir en el dispositivo de almacenamiento, en kilobytes.
	`cached`	Cached Memory	La cantidad de memoria utilizada para almacenar en la caché las E/S basadas en el sistema de archivos.
	`dirty`	Dirty Memory	La cantidad de páginas de memoria en la RAM que se han modificado, pero no escrito, en su bloque de datos relacionado en el almacenamiento, en kilobytes.
	`free`	Free Memory	La cantidad de memoria no asignada, en kilobytes.

Grupo	Métrica	Nombre de la consola	Descripción
	`hugePagesFree`	Huge Pages Free	El número de páginas de gran tamaño libres. Las páginas de gran tamaño son una característica del kernel de Linux.
	`hugePagesRsvd`	Huge Pages Rsvd	El número de páginas de gran tamaño confirmadas.
	`hugePagesSize`	Huge Pages Size	El tamaño de cada unidad de páginas de gran tamaño, en kilobytes.
	`hugePagesSurp`	Huge Pages Surp	El número de páginas de gran tamaño sobrantes disponibles con respecto al total.
	`hugePagesTotal`	Huge Pages Total	El número total de páginas enormes.
	`inactive`	Inactive Memory	La cantidad de páginas de memoria utilizadas con menor frecuencia, en kilobytes.
	`mapped`	Mapped Memory	La cantidad total de contenido del sistema de archivos mapeado a la memoria dentro de un espacio de direcciones de proceso, en kilobytes.
	`pageTables`	Page Tables	La cantidad de memoria utilizada por tablas de página, en kilobytes.
	`slab`	Slab Memory	La cantidad de estructuras de datos de kernel reutilizables, en kilobytes.
	`total`	Memoria total	La cantidad total de memoria, en kilobytes.

Grupo	Métrica	Nombre de la consola	Descripción
	`writeback`	Writeback Memory	La cantidad de páginas desfasadas en la RAM que se siguen escribiendo en el almacenamiento de respaldo, en kilobytes.
`network`	`interface`	No aplicable	El identificador para la interfaz de red que se utiliza para la instancia de base de datos.
	`rx`	RX	El número de bytes recibidos por segundo.
	`tx`	TX	El número de bytes cargados por segundo.
`processList`	`cpuUsedPc`	CPU %	El porcentaje de CPU utilizado por el proceso.
	`id`	No aplicable	El identificador del proceso.
	`memoryUsedPc`	MEM%	El porcentaje de memoria que utiliza el proceso.
	`name`	No aplicable	El nombre del proceso.
	`parentID`	No aplicable	El identificador correspondiente al proceso principal.
	`rss`	RES	La cantidad de RAM asignada al proceso, en kilobytes.
	`tgid`	No aplicable	El identificador del grupo de subprocesos, que es un número que representa el ID del proceso al que pertenece un subproceso. Este identificador se utiliza para agrupar subprocesos del mismo proceso.
	`vss`	VIRT	La cantidad de memoria virtual asignada al proceso, en kilobytes.

Grupo	Métrica	Nombre de la consola	Descripción
`swap`	`total`	Swap	La cantidad de memoria de intercambio disponible, en kilobytes.
	`in`	Swaps in	La cantidad de memoria, en kilobytes, intercambiada desde disco.
	`out`	Swaps out	La cantidad de memoria, en kilobytes, intercambiada del disco.
	`free`	Free Swap	La cantidad de memoria de intercambio no asignada, en kilobytes.
	`cached`	Committed Swap	La cantidad de memoria de intercambio, en kilobytes, utilizada como memoria caché.
`tasks`	`blocked`	Tasks Blocked	El número de tareas que están bloqueadas.
	`running`	Tasks Running	El número de tareas que están en ejecución.
	`sleeping`	Tasks Sleeping	El número de tareas que están inactivas.
	`stopped`	Tasks Stopped	El número de tareas que se han detenido.
	`total`	Tasks Total	El número total de tareas.
	`zombie`	Tasks Zombie	El número de tareas secundarias inactivas con una tarea principal activa.

Supervisión de eventos, registros y flujos en un clúster de bases de datos de Amazon Aurora

Cuando supervisa sus bases de datos de Amazon Aurora y sus otras soluciones de AWS, su objetivo es mantener lo siguiente:

- Fiabilidad
- Disponibilidad
- Rendimiento
- Seguridad

Supervisión de métricas en un clúster de Amazon Aurora explica cómo supervisar su clúster mediante las métricas. Una solución completa también debe supervisar los eventos, los archivos de registro y los flujos de actividad de la base de datos. AWS le proporciona las siguientes herramientas de supervisión:

- Amazon EventBridge es un bus de eventos sin servidor que facilita la conexión de sus aplicaciones con datos de varios orígenes. EventBridge proporciona un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones de software como servicio (SaaS) y servicios de AWS. EventBridge dirige esos datos a los objetivos, como AWS Lambda. De esta forma, puede supervisar los eventos que ocurren en los servicios y crear arquitecturas basadas en eventos. Para más información, consulte la Guía del usuario de Amazon EventBridge.
- Registros de Amazon CloudWatch le permite supervisar, almacenar y acceder a los archivos de registro desde instancias de Amazon Aurora, AWS CloudTrail y otros orígenes. registros de Amazon Cloudwatch puede supervisar información en los archivos de registro y enviar una notificación cuando se llega a determinados umbrales. También se pueden archivar los datos del registro en un almacenamiento de larga duración. Para obtener más información, consulte la Guía del usuario de Registros de Amazon CloudWatch.
- AWS CloudTrail captura las llamadas a la API y los eventos relacionados realizados por su cuenta de Cuenta de AWS o en su nombre. CloudTrail entrega los archivos de registros a un bucket de Amazon S3 que especifique. También puede identificar qué usuarios y cuentas llamaron a AWS, la dirección IP de origen de las llamadas y el momento en que se hicieron. Para más información, consulte la Guía del usuario de AWS CloudTrail.

- Los flujos de actividad de la base de datos son una característica de Amazon Aurora que proporciona un flujo casi en tiempo real de la actividad en su instancia de base de datos. Amazon Aurora envía actividades a un flujo de datos de Amazon Kinesis. El flujo de Kinesis se crea automáticamente. Desde Kinesis, puede configurar servicios de AWS como Amazon Data Firehose y AWS Lambda para utilizar el flujo y almacenar los datos.

Temas

- Visualización de los registros, los eventos y los flujos en la consola de Amazon RDS
- Supervisión de eventos de Amazon Aurora
- Supervisión de archivos de registro de Amazon Aurora
- Supervisión de llamadas a la API de Amazon Aurora en AWS CloudTrail
- Supervisión de Amazon Aurora con flujos de actividad de la base de datos
- Supervisión de amenazas con Amazon GuardDuty para protección de RDS para Amazon Aurora

Visualización de los registros, los eventos y los flujos en la consola de Amazon RDS

Amazon RDS se integra con Servicios de AWS para mostrar información sobre registros, eventos y flujos de actividad de bases de datos en la consola de RDS.

La pestaña Logs & events (Registros y eventos) para el clúster de base de datos de RDS muestra la siguiente información:

- Auto scaling policies and activities (Políticas y actividades de escalado automático): muestra las políticas y las actividades relacionadas con la característica escalado automático de Aurora. Esta información solo aparece en la pestaña Logs & Events (Registros y eventos) a nivel de clúster.
- Amazon CloudWatch alarms (Alarmas de Amazon CloudWatch): muestra las alarmas de métricas que ha configurado para la instancia de base de datos en el clúster de Aurora. Si no ha configurado las alarmas, puede crearlas en la consola de RDS.
- Recent events (Eventos recientes): muestra un resumen de los eventos (cambios de entorno) para la instancia de base de datos o clúster de RDS. Para obtener más información, consulte Consulta de eventos de Amazon RDS.

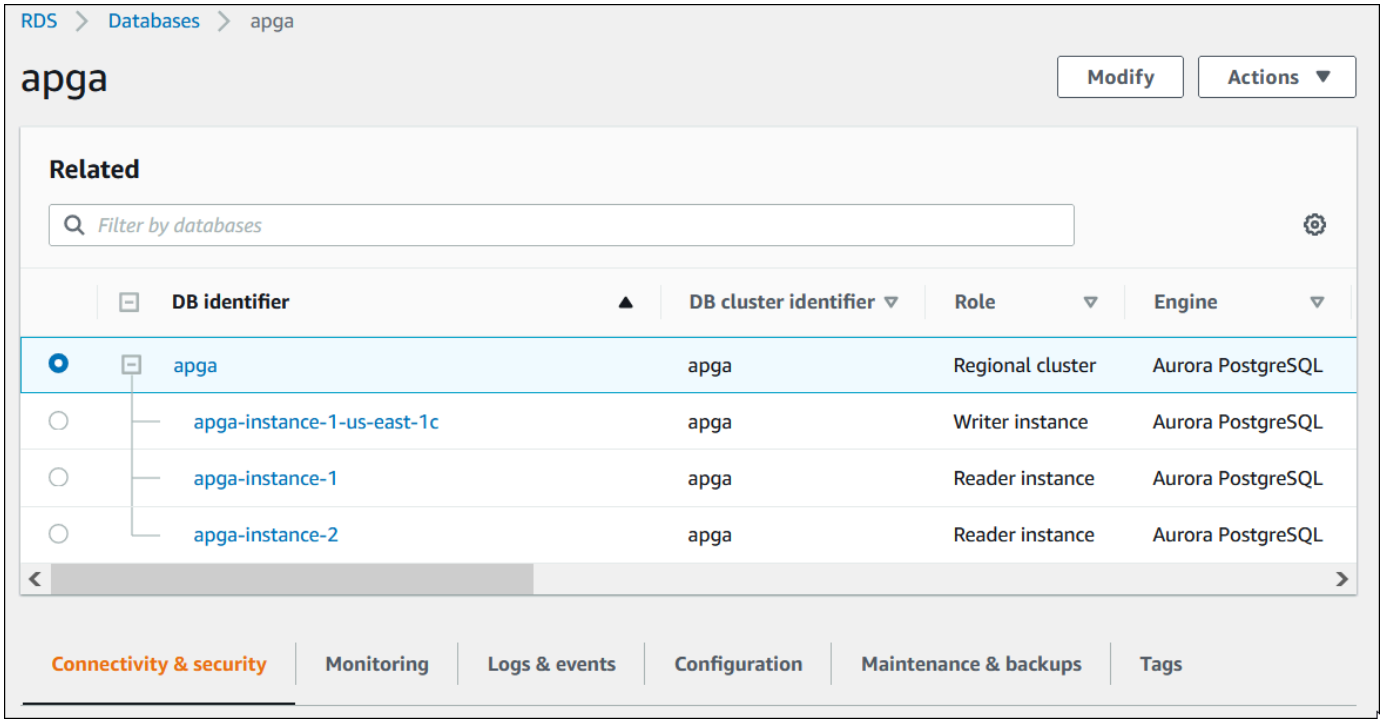
- Logs (Registros): muestra los archivos de registro de base de datos generados por una instancia de base de datos en el clúster de Aurora. Para obtener más información, consulte Supervisión de archivos de registro de Amazon Aurora.

La pestaña Configuration (Configuración) muestra información sobre flujos de actividad de la base de datos.

Para ver los registros, eventos y flujos de su instancia de Aurora en la consola de RDS

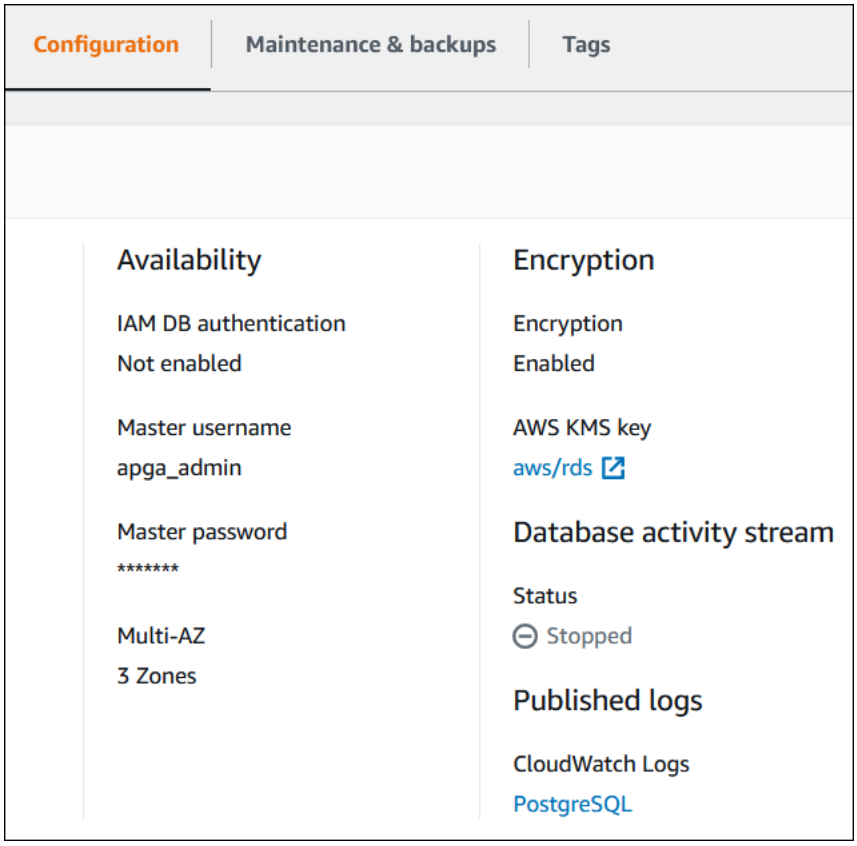
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el nombre de su instancia de Aurora que desea supervisar.

   Aparece la página de la base de datos. En el siguiente ejemplo, se muestra un clúster de base datos de Amazon Aurora PostgreSQL denominado apga.



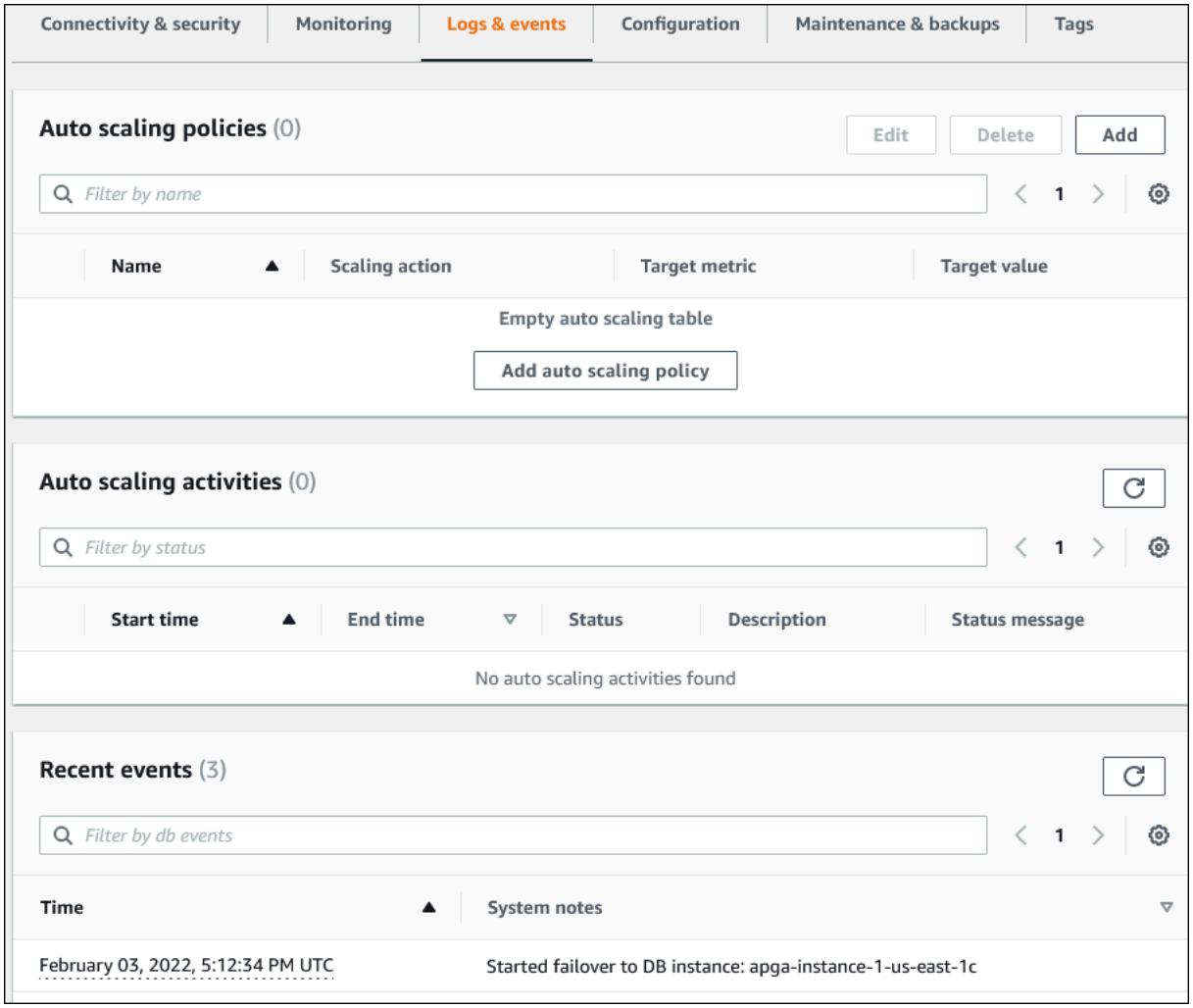
4. Desplácese hacia abajo y elija Configuration (Configuración).

   En el siguiente ejemplo, se muestra el estado de los flujos de actividad de la base de datos para la instancia de base de datos de RDS para Oracle.



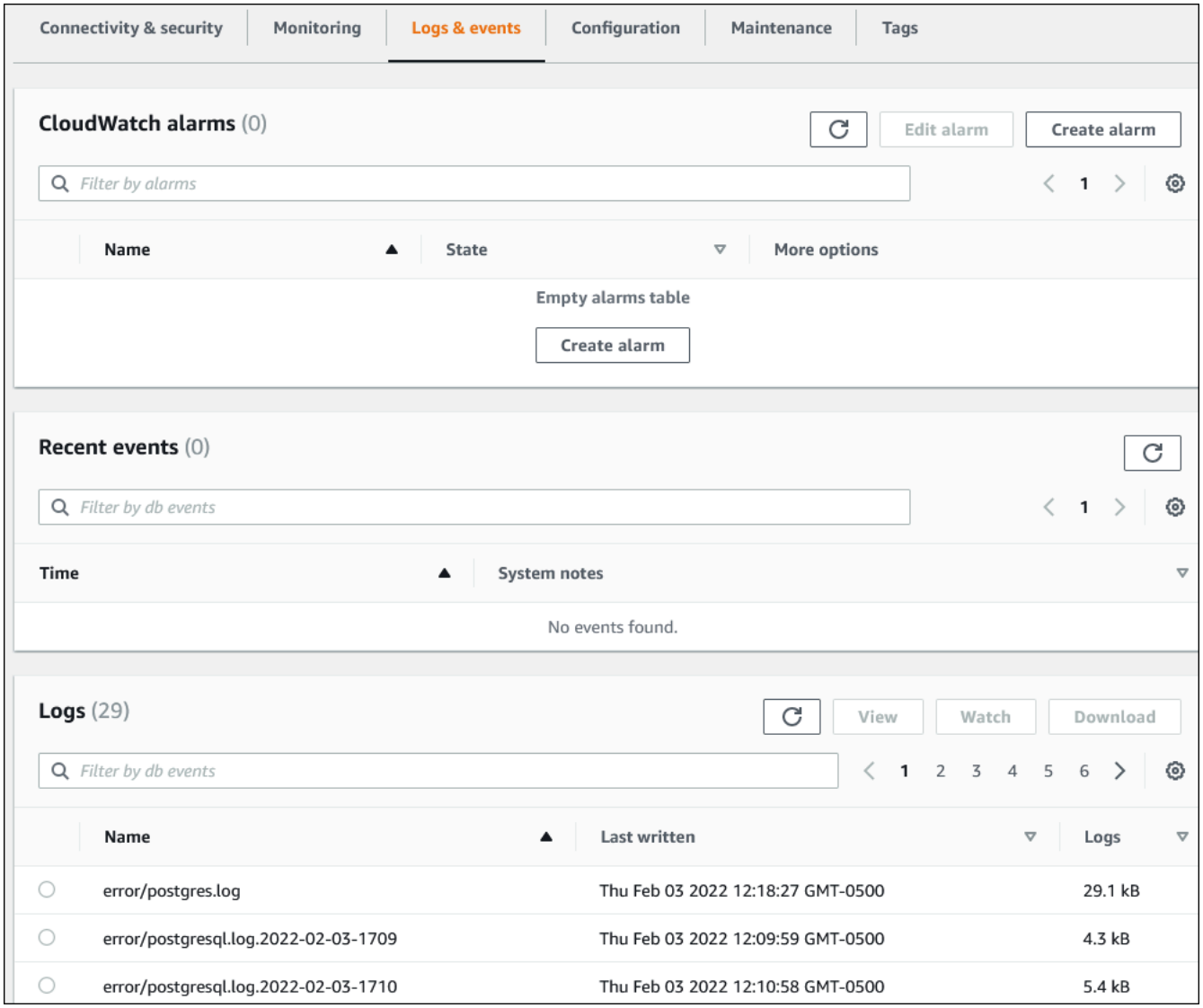
5. Seleccione Logs & events (Registros y eventos).

   Aparece la sección Logs & events (Registros y eventos).



6. Elija una instancia de base de datos en el clúster de Aurora y, a continuación, elija Logs & Events (Registros y eventos) para la instancia.

   En el siguiente ejemplo se muestra que el contenido es diferente entre la página de instancia de base de datos y la página del clúster de bases de datos. En la página de instancia de base de datos se muestran registros y alarmas.


Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance
Tags
CloudWatch alarms (0)
Edit alarm
Create alarm
Filter by alarms
1
Name
State
More options
Empty alarms table
Create alarm
Recent events (0)
Filter by db events
1
Time
System notes
No events found.
Logs (29)
View
Watch
Download
Filter by db events
1 2 3 4 5 6
Name
Last written
Logs
error/postgres.log
Thu Feb 03 2022 12:18:27 GMT-0500
29.1 kB
error/postgresql.log.2022-02-03-1709
Thu Feb 03 2022 12:09:59 GMT-0500
4.3 kB
error/postgresql.log.2022-02-03-1710
Thu Feb 03 2022 12:10:58 GMT-0500
5.4 kB

Supervisión de eventos de Amazon Aurora

Un evento indica un cambio en el entorno. Puede ser un entorno de AWS, un servicio o aplicación de socios SaaS o una aplicación o servicio personalizados. Para obtener descripciones de los eventos de Aurora, consulte Categorías y mensajes de eventos de Amazon RDS para Aurora.

Temas

- Información general de los eventos para Aurora
- Consulta de eventos de Amazon RDS
- Uso de notificaciones de eventos de Amazon RDS
- Creación de una regla que se desencadena en función de un evento Amazon Aurora
- Categorías y mensajes de eventos de Amazon RDS para Aurora

Información general de los eventos para Aurora

Un evento de RDS indica un cambio en el entorno de Aurora. Por ejemplo, Amazon Aurora genera un evento cuando el clúster de una base de datos tiene revisiones. Amazon Aurora envía eventos a EventBridge casi en tiempo real.

> **Note**
>
> Amazon RDS emite eventos de la mejor forma posible. Recomendamos que evite escribir programas que dependan del orden o de la existencia de eventos de notificación, ya que pueden faltar o no estar ordenados.

Amazon RDS registra eventos relacionados con los siguientes recursos:

- Clústeres de base de datos

  Para obtener una lista de los eventos de clúster, consulte Eventos de clúster de bases de datos.
- Instancias de base de datos

  Para obtener una lista de los eventos de instancia de base de datos, consulte Eventos de instancia de base de datos.
- Grupos de parámetros de base de datos

Para obtener una lista de eventos de grupos de parámetros de base de datos, consulte Eventos de grupo de parámetros de base de datos.

- Grupos de seguridad de base de datos

  Para obtener una lista de eventos de grupo de seguridad de base de datos, consulte Eventos de grupo de seguridad de base de datos.

- Instantáneas de clúster de base de datos

  Para obtener una lista de los eventos de instantánea de clúster de base de datos, consulte Eventos de instantánea de clúster de bases de datos.

- Eventos de RDS Proxy

  Para obtener una lista de los eventos de RDS Proxy, consulte Eventos de RDS Proxy.

- Eventos de implementación azul/verde

  Para obtener una lista de los eventos de implementación azul/verde, consulte Eventos de implementación azul/verde.

La información incluye lo siguiente:

- La fecha y la hora del evento
- El nombre de origen y el tipo de origen del evento
- Un mensaje asociado al evento
- Las notificaciones de eventos incluyen etiquetas de cuando se envió el mensaje y es posible que no reflejen las etiquetas del momento en que se produjo el evento.

Consulta de eventos de Amazon RDS

Puede recuperar la siguiente información del evento para sus recursos de Amazon Aurora:

- Nombre del recurso
- Tipo de recurso
- Hora del evento
- Resumen del mensaje del evento

Puede acceder a los eventos en las siguientes partes de la:Consola de administración de AWS

- La pestaña Eventos, que muestra los eventos de las últimas 24 horas.
- La tabla Eventos recientes de la sección Registros y eventos de la pestaña Bases de datos, que puede mostrar los eventos de las últimas 2 semanas.

También puede recuperar eventos utilizando el comando describe-events de la AWS CLI o la operación DescribeEvents de la API de RDS. Si utiliza la AWS CLI o la API de RDS para ver eventos, puede recuperar eventos de los últimos 14 días como máximo.

> **Note**
>
> Si necesita almacenar eventos durante periodos de tiempo más largos, puede enviar eventos de Amazon RDS a EventBridge. Para obtener más información, consulte Creación de una regla que se desencadena en función de un evento Amazon Aurora.

Para obtener descripciones de los eventos de Amazon Aurora, consulte Categorías y mensajes de eventos de Amazon RDS para Aurora.

Para acceder a información detallada sobre los eventos mediante AWS CloudTrail, incluidos los parámetros de la solicitud, consulte Eventos de CloudTrail.

Consola

Para ver todos los eventos de Amazon RDS de las últimas 24 horas

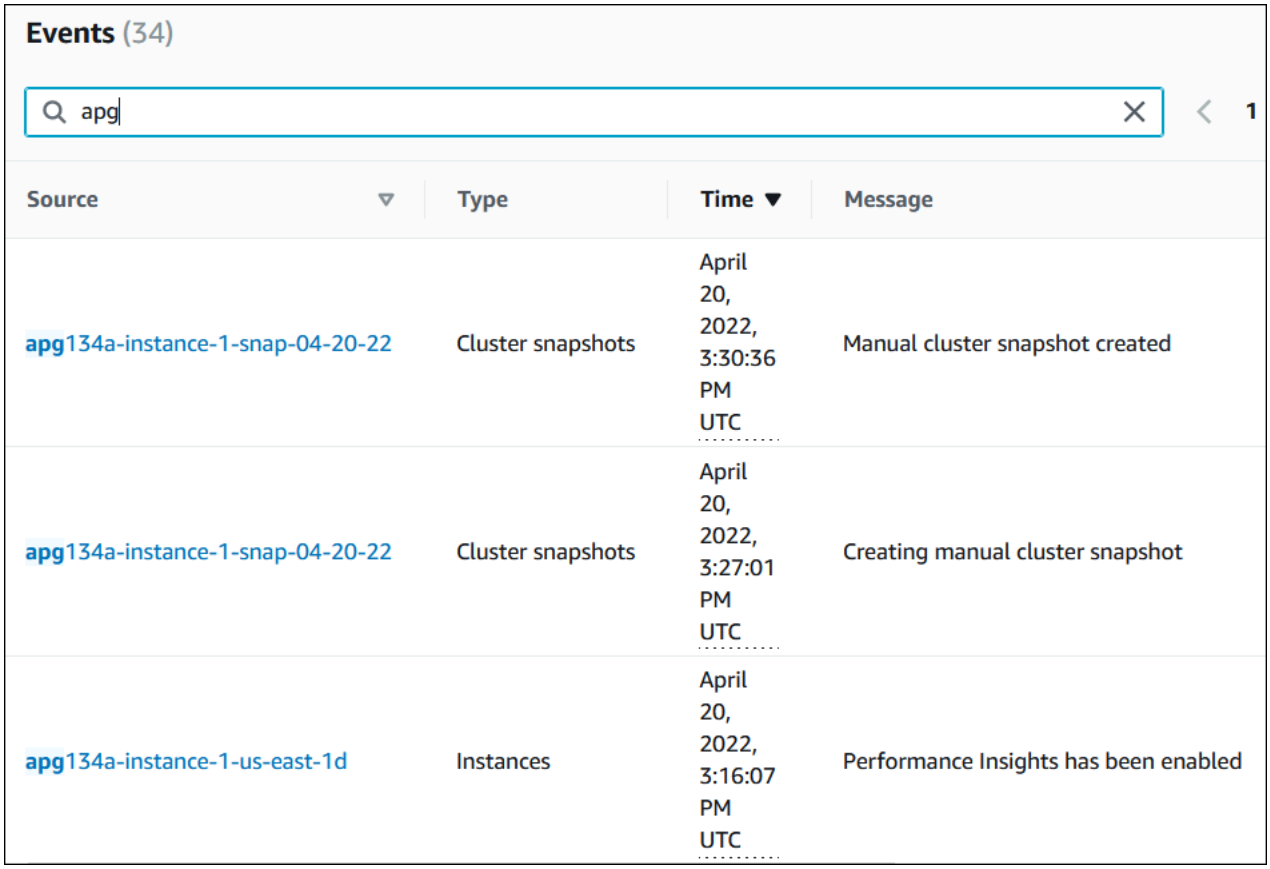
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, seleccione Events.

   Los eventos disponibles aparecen en una lista.

3. (Opcional) Ingrese una palabra de búsqueda para filtrar los resultados.

   En el siguiente ejemplo se muestra una lista de eventos filtrados por los caracteres **apg**.



AWS CLI

Para ver todos los eventos generados en la última hora, llame a describe-events sin parámetros.

```
aws rds describe-events
```

El siguiente ejemplo muestra que se ha iniciado la recuperación de una instancia de clúster de base de datos.

```
{
    "Events": [
        {
            "EventCategories": [
```

```
                "recovery"
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mycluster-instance-1",
            "Date": "2022-04-20T15:02:38.416Z",
            "Message": "Recovery of the DB instance has started. Recovery time will
 vary with the amount of data to be recovered.",
            "SourceIdentifier": "mycluster-instance-1"
        }, ...
```

Para ver todos los eventos de Amazon RDS de los últimos 10 080 minutos (7 días), llame al comando describe-events de la AWS CLI y establezca el parámetro `--duration` en `10080`.

```
aws rds describe-events --duration 10080
```

El siguiente ejemplo muestra los eventos del intervalo de tiempo especificado para la instancia de base de datos *`test-instance`*.

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

El siguiente ejemplo muestra el estado de una copia de seguridad.

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
```

```
                "backup"
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

API

Puede ver todos los eventos de las instancias de Amazon RDS de los últimos 14 días llamando a la operación DescribeEvents de la API de RDS y estableciendo el parámetro `Duration` en `20160`.

Uso de notificaciones de eventos de Amazon RDS

Amazon RDS utiliza Amazon Simple Notification Service (Amazon SNS) para proporcionar una notificación cuando se produce un evento de Amazon RDS. Estas notificaciones pueden realizarse con cualquier método que admita Amazon SNS para una región de AWS, como un email, un mensaje de texto o una llamada a un punto de enlace HTTP.

Temas

- Información general de las notificaciones de eventos de Amazon RDS
- Concesión de permisos para publicar notificaciones en un tema de Amazon SNS
- Suscripción a notificaciones de eventos de Amazon RDS
- Atributos y etiquetas de notificación de eventos de Amazon RDS
- Descripción de suscripciones a notificaciones de eventos de Amazon RDS
- Modificación de una suscripción a notificaciones de eventos de Amazon RDS
- Agregar un identificador de origen a una suscripción de notificación de eventos de Amazon RDS
- Quitar un identificador de origen de una suscripción de notificación de eventos de Amazon RDS
- Descripción de la lista de categorías de notificaciones de eventos de Amazon RDS
- Eliminar una suscripción de notificación de eventos de Amazon RDS

Información general de las notificaciones de eventos de Amazon RDS

Amazon RDS agrupa los eventos en categorías a las que puede suscribirse para recibir una notificación cada vez que se produzca un evento en esa categoría.

Temas

- Recursos de RDS aptos para la suscripción a eventos
- Proceso básico para suscribirse a las notificaciones de eventos de Amazon RDS
- Entrega de notificaciones de eventos de RDS
- Facturación de notificaciones de eventos de Amazon RDS
- Ejemplos de eventos de Aurora con Amazon EventBridge

Recursos de RDS aptos para la suscripción a eventos

Para Amazon Aurora, los eventos se producen en el clúster de bases de datos y en la instancia de base de datos. Puede suscribirse a una categoría de evento para los recursos siguientes:

- Instancia de base de datos
- clúster de bases de datos
- Instantánea de clúster de bases de datos
- Grupo de parámetros de base de datos
- Grupo de seguridad de base de datos
- RDS Proxy
- Versión del motor personalizada

Por ejemplo, si se suscribe a la categoría de copia de seguridad de una instancia de base de datos determinada, recibirá una notificación cada vez que se produzca un evento relacionado con las copias de seguridad que afecte a la instancia de base de datos. Si se suscribe a una categoría de cambio de configuración para una instancia de base de datos, recibirá una notificación cuando la instancia de base de datos se modifique. También recibirá una notificación cuando cambie una suscripción de notificación de eventos.

Es posible que desee crear varias suscripciones diferentes. Por ejemplo, puede crear una suscripción que reciba todas las notificaciones de eventos de todas las instancias de base de datos y otra que incluya solo los eventos fundamentales de un subconjunto de instancias de base de datos. Para la segunda suscripción, especifique una o más instancias de base de datos en el filtro.

Proceso básico para suscribirse a las notificaciones de eventos de Amazon RDS

El proceso para suscribirse a las notificaciones de eventos de Amazon RDS es el siguiente:

1. Cree una suscripción de notificación de eventos de Amazon RDS mediante la consola de Amazon RDS, la AWS CLI o la API.

   Amazon RDS utiliza el ARN de un tema de Amazon SNS para identificar cada suscripción. La consola de Amazon RDS crea el ARN automáticamente cuando se crea la suscripción. Cree el ARN a través de la consola de Amazon SNS, la AWS CLI o la API de Amazon SNS.
2. Amazon RDS envía un mensaje de correo electrónico o SMS de aprobación a las direcciones que envió con la suscripción.

3. Para confirmar la suscripción, elija el enlace de la notificación que ha recibido.
4. La consola de Amazon RDS actualiza la sección My Event Subscriptions (Mis suscripciones a eventos) con el estado de la suscripción.
5. Amazon RDS empieza a enviar notificaciones a las direcciones que se proporcionan al crear la suscripción.

Para obtener información acerca de Identity and access management cuando se utilice Amazon SNS, consulte Identity and access management en Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.

Puede utilizar AWS Lambda para procesar notificaciones de eventos desde una instancia de base de datos. Para obtener más información, consulte Uso de AWS Lambda con Amazon RDS en la Guía para desarrolladores de AWS Lambda.

Entrega de notificaciones de eventos de RDS

Amazon RDS envía notificaciones a las direcciones que se proporcionan al crear la suscripción. La notificación puede incluir atributos de mensaje que proporcionan metadatos estructurados sobre el mensaje. Para obtener más información acerca de los atributos de los mensajes, consulte Categorías y mensajes de eventos de Amazon RDS para Aurora.

Las notificaciones de eventos pueden tardar hasta cinco minutos en entregarse.

> **Important**
>
> Amazon RDS no garantiza el orden de los eventos enviados en una secuencia de eventos. El orden de los eventos está sujeto a cambio.

Cuando Amazon SNS envía una notificación a un punto de enlace HTTP o HTTPS suscrito, el cuerpo del mensaje POST enviado al punto de enlace contiene un documento JSON. Para obtener más información, consulte Formatos de mensaje y JSON de Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.

Puede configurar SNS para que le notifique con mensajes de texto. Para obtener más información, consulte Mensajería de texto móvil (SMS) en la Guía para desarrolladores de Simple Notification Service.

Para desactivar las notificaciones sin eliminar una suscripción, elija No en Enabled (Habilitado) en la consola de Amazon RDS. O bien, puede establecer el parámetro `Enabled` en `false` mediante la AWS CLI o la API de Amazon RDS.

Facturación de notificaciones de eventos de Amazon RDS

La facturación de notificaciones de eventos de Amazon RDS se efectúa a través de Amazon SNS. Se aplican las tarifas de Amazon SNS cuando se utiliza la notificación de eventos. Para obtener más información sobre la facturación de Amazon SNS, consulte Precios de Amazon Simple Notification Service.

Ejemplos de eventos de Aurora con Amazon EventBridge

Los siguientes ejemplos muestran los diferentes tipos de eventos de Aurora en formato JSON. Para ver un tutorial que muestra cómo capturar y ver eventos en formato JSON, consulte Tutorial: Registrar el estado de una instancia de base de datos con Amazon EventBridge.

Temas

- Ejemplo de evento de clúster de bases de datos
- Ejemplo de evento de grupo de parámetros de base de datos
- Ejemplo de evento de instantánea de clúster de bases de datos

Ejemplo de evento de clúster de bases de datos

A continuación, se muestra un ejemplo de evento de clúster de bases de datos en formato JSON. El evento muestra que se aplicaron parches al clúster denominado `my-db-cluster`. El ID del evento es `RDS-EVENT-0173`.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster"
  ],
```

```
  "detail": {
    "EventCategories": [
      "notification"
    ],
    "SourceType": "CLUSTER",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Database cluster has been patched",
    "SourceIdentifier": "my-db-cluster",
    "EventID": "RDS-EVENT-0173"
  }
}
```

Ejemplo de evento de grupo de parámetros de base de datos

A continuación, se muestra un ejemplo de un evento de grupo de parámetros de base de datos en formato JSON. El evento muestra que el parámetro `time_zone` se actualizó en el grupo de parámetros `my-db-param-group`. El ID de evento es RDS-EVENT-0037.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

Ejemplo de evento de instantánea de clúster de bases de datos

A continuación, se muestra un ejemplo de un evento de instantánea de clúster de bases de datos en formato JSON. El evento muestra la creación de la instantánea denominada `my-db-cluster-snapshot`. El ID de evento es RDS-EVENT-0074.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Snapshot Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-cluster-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "backup"
    ],
    "SourceType": "CLUSTER_SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-
cluster-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "SourceIdentifier": "my-db-cluster-snapshot",
    "Message": "Creating manual cluster snapshot",
    "EventID": "RDS-EVENT-0074"
  }
}
```

Concesión de permisos para publicar notificaciones en un tema de Amazon SNS

Para conceder a Amazon RDS permisos para publicar notificaciones en un tema de Amazon Simple Notification Service (Amazon SNS), adjunte una política de AWS Identity and Access Management (IAM) al tema de destino. Para obtener más información sobre los permisos, consulte Ejemplos de casos de control de acceso con Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.

De forma predeterminada, un tema de Amazon SNS tiene una política que permite a todos los recursos de Amazon RDS de la misma cuenta publicar notificaciones en él. Puede adjuntar una política personalizada para permitir las notificaciones entre cuentas o para restringir el acceso a determinados recursos.

A continuación se muestra un ejemplo de una política de IAM que se asocia al tema de Amazon SNS de destino. Restringe el tema a instancias de base de datos con nombres que coinciden con el prefijo especificado. Para utilizar esta política, especifique los siguientes valores:

- `Resource`: el nombre de recurso de Amazon (ARN) para el tema de Amazon SNS
- `SourceARN`: su ARN de recursos de RDS
- `SourceAccount`: su ID de Cuenta de AWS

Para ver una lista de tipos de recursos y sus ARN, consulte Tipos de recurso definidos por Amazon RDS en la Referencia de autorizaciones de servicio.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
      "Condition": {
```

```
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Suscripción a notificaciones de eventos de Amazon RDS

La forma más sencilla de crear una suscripción es con la consola de RDS. Si decide crear las suscripciones de notificaciones de eventos a través de la CLI o la API, debe crear un tema de Amazon Simple Notification Service y suscribirse a dicho tema con la consola de Amazon SNS o la API de Amazon SNS. También deberá conservar el Nombre de recurso de Amazon (ARN) del tema, ya que este se utiliza al enviar los comandos de la CLI o las operaciones de la API. Para obtener información acerca de cómo crear un tema de SNS y suscribirse a él, consulte Introducción a Amazon SNS en la Guía del desarrollador de Amazon Simple Notification Service.

Puede especificar el tipo de origen sobre el que desea recibir notificaciones y el origen de Amazon RDS que inicia el evento:

Source type (Tipo de origen)

Tipo de origen. Por ejemplo, Source Type (Tipo de origen) podría ser Instances (Instancias). Debe elegir un tipo de origen.

Resources to include (Recursos a incluir)

Los recursos de Amazon RDS que están generando los eventos. Por ejemplo, es posible que tenga que elegir Select specific instances (Seleccionar instancias específicas) y luego myDBInstance1.

En la siguiente tabla se explica lo que ocurre si se especifica o no ***Resources*** to include (Recursos a incluir).

Recursos a incluir	Descripción	Ejemplo
Especificado	RDS le notifica todos los eventos relacionados únicamente con el recurso especificado.	Si Source type (Tipo de origen) es Instances (Instancias) y tu recurso es myDBInstance1, RDS le notifica sobre todos los eventos de MyDBInstance1 únicamente.

Recursos a incluir	Descripción	Ejemplo
No especificada	RDS le notifica los eventos del tipo de origen especificado para todos los recursos de Amazon RDS.	Si Source type (Tipo de origen) es Instances (Instancias), RDS le notifica sobre todos los eventos relacionados con las instancias en su cuenta.

Un suscriptor de temas de Amazon SNS recibe de forma predeterminada todos los mensajes publicados en el tema. Para recibir solo un subconjunto de los mensajes, el suscriptor debe asignar una política de filtrado a la suscripción del tema. Para obtener más información sobre el filtrado de mensajes SNS, consulte Filtrado de mensajes en Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.

Consola

Para suscribirse a las notificaciones de eventos de RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos).
3. En la página Event Subscriptions (Suscripciones de eventos) seleccione Create Event Subscription (Crear suscripción de eventos).
4. Introduzca los detalles de su suscripción de la siguiente manera:

    a. En Name (Nombre) escriba un nombre para la suscripción de notificación de evento.

    b. En Send notification to (Enviar notificaciones a), realice una de las siguientes acciones:

    - Elija New email topic (Nuevo tema de correo electrónico). Introduzca un nombre para su tema de correo electrónico y una lista de destinatarios. Le recomendamos que configure las suscripciones a los eventos con la misma dirección de correo electrónico que tiene el contacto principal de su cuenta. Las recomendaciones, los eventos de servicio y los mensajes sanitarios personales se envían a través de diferentes canales. Las suscripciones a la misma dirección de correo electrónico garantizan que todos los mensajes se consoliden en una sola ubicación.

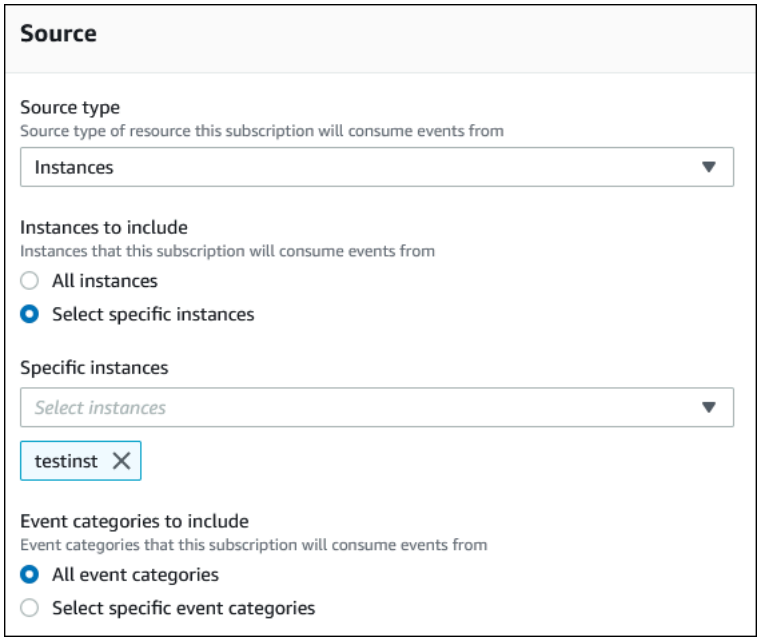
- Elija Amazon Resource Name (ARN) [Nombre de recurso de Amazon (ARN)]. A continuación, seleccione el ARN de Amazon SNS existente para un tema de Amazon SNS.

  Si desea utilizar un tema que se haya habilitado para el cifrado del servidor (SSE), conceda a Amazon RDS los permisos necesarios para acceder a la AWS KMS key. Para obtener más información, consulte Habilitar la compatibilidad entre los orígenes de eventos de los servicios de AWS y los temas cifrados en la Guía para desarrolladores de Amazon Simple Notification Service.

c. En Source type (Tipo de origen) elija un tipo de origen. Por ejemplo, elija clústers (Clústeres) clúster snapshots (Instantáneas de clústeres).

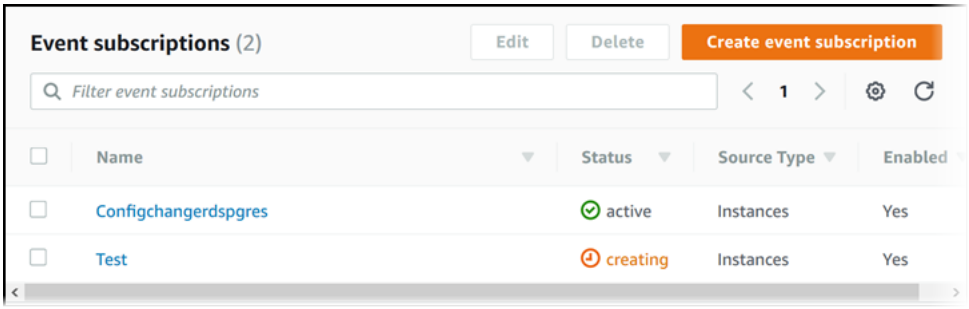
d. Elija las categorías y recursos del evento de las que desea recibir notificaciones.

   En el ejemplo siguiente se configuran las notificaciones de eventos de la instancia de base de datos denominada `testinst`.



e. Seleccione Crear.

La consola de Amazon RDS indica que se está creando la suscripción.



AWS CLI

Para suscribirse a notificaciones de eventos de RDS, utilice el comando AWS CLI de la `create-event-subscription`. Incluya los siguientes parámetros obligatorios:

- `--subscription-name`
- `--sns-topic-arn`

Example

Para Linux, macOS o:Unix

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

En:Windows

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

API

Para suscribirse a notificaciones de eventos de Amazon RDS, llame a la función `CreateEventSubscription` de la API de Amazon RDS. Incluya los siguientes parámetros obligatorios:

- `SubscriptionName`

- `SnsTopicArn`

Atributos y etiquetas de notificación de eventos de Amazon RDS

Cuando Amazon RDS envía una notificación de evento a Amazon Simple Notification Service (SNS) o Amazon EventBridge, esa notificación contiene los atributos del mensaje y las etiquetas del evento. RDS envía los atributos del mensaje por separado junto con el mensaje, mientras que las etiquetas de eventos se encuentran en el cuerpo del mensaje. Utilice los atributos de los mensajes y las etiquetas de Amazon RDS para añadir metadatos a sus recursos. Puede modificar estas etiquetas con sus propias anotaciones sobre las instancias de base de datos, los clústeres de Aurora, etc. Para obtener más información acerca del etiquetado de recursos de Amazon RDS, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.

De forma predeterminada, Amazon SNS y Amazon EventBridge reciben todos los mensajes que se les envían. SNS y EventBridge pueden filtrar el mensaje y enviar las notificaciones a través del método de comunicación que se prefiera, como un correo electrónico, un mensaje de texto o una llamada a un punto de conexión HTTP.

> **Note**
>
> La notificación que se envía por correo electrónico o en mensaje de texto no tiene etiquetas de evento.

En la tabla siguiente, se muestran los atributos de mensaje para los eventos de RDS que se han enviado al suscriptor de temas.

Atributo de evento de Amazon RDS	Descripción
EventID	Identificador de mensajes de eventos de RDS, por ejemplo, RDS-EVENT-0006.
Recurso	Identificador ARN del recurso que emite el evento como, por ejemplo, `arn:aws:rds:ap-southeast-2:123456789012:db:database-1` .

Las etiquetas RDS proporcionan datos sobre el recurso afectado por el evento de servicio. RDS añade el estado actual de las etiquetas en el cuerpo del mensaje cuando la notificación se envía a SNS o EventBridge.

Para obtener más información sobre el filtrado de atributos de mensajes SNS, consulte Filtrado de mensajes en Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.

Para obtener más información sobre etiquetas de eventos de EventBridge, consulte Operadores de comparación para su uso en patrones de eventos en Amazon EventBridge en la Guía del usuario de Amazon EventBridge.

Para obtener más información sobre el filtrado etiquetas basadas en cargas útiles para SNS, consulte Introducing payload-based message filtering for Amazon SNS.

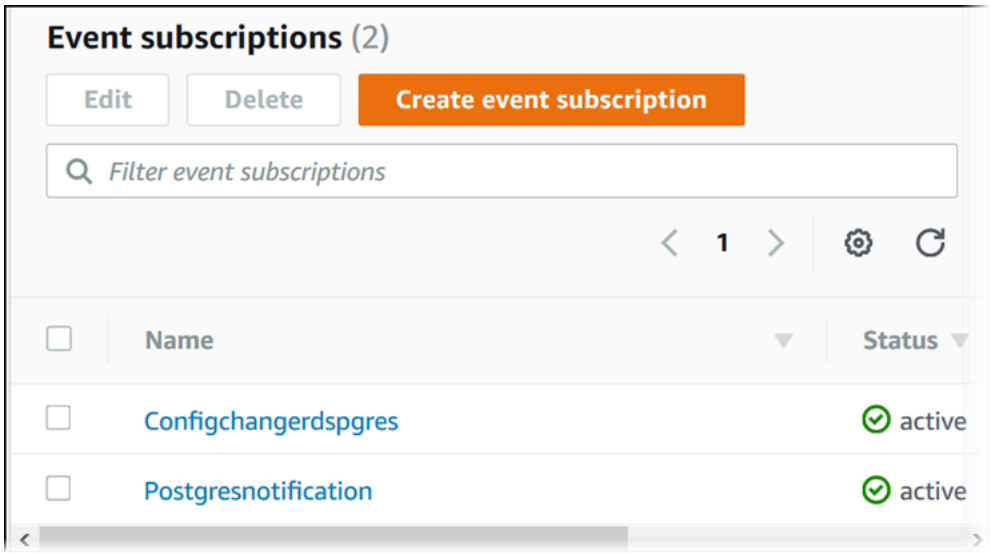
Descripción de suscripciones a notificaciones de eventos de Amazon RDS

Puede mostrar las suscripciones actuales de notificación de eventos de Amazon RDS.

Consola

Pasos mostrar las sus suscripciones actuales de notificación de eventos de Amazon RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos). El panel Event subscriptions (Suscripciones de eventos) muestra todas sus suscripciones a notificaciones de eventos.



AWS CLI

Para obtener una lista de sus suscripciones a notificaciones de eventos de Amazon RDS, utilice el comando `describe-event-subscriptions` de la AWS CLI.

Example

En el siguiente ejemplo se obtienen todas las suscripciones a eventos.

```
aws rds describe-event-subscriptions
```

En el siguiente ejemplo se obtiene la descripción de `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

Para obtener una lista de sus suscripciones actuales a notificaciones de eventos de Amazon RDS, llame a la acción `DescribeEventSubscriptions` de la API de Amazon RDS.

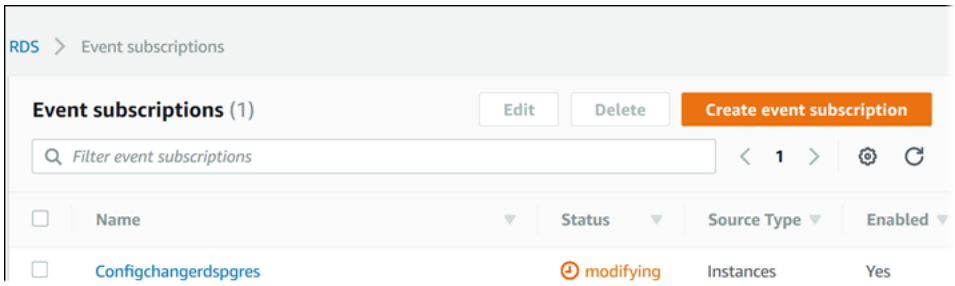
Modificación de una suscripción a notificaciones de eventos de Amazon RDS

Después de crear una suscripción, puede cambiar el nombre de la suscripción, el identificador del origen, las categorías o el ARN del tema.

Consola

Para modificar una suscripción de notificación de eventos de Amazon RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación seleccione Event Subscriptions (Suscripciones de eventos).
3. En el panel Event subscriptions (Suscripciones de eventos), elija la suscripción que desea modificar y elija Edit (Editar).
4. Realice los cambios que desee en la suscripción en las secciones Target (Objetivo) o Source (Fuente).
5. Elija Edit (Editar). La consola de Amazon RDS indica que se está modificando la suscripción.



AWS CLI

Para modificar una suscripción a notificaciones de eventos de Amazon RDS, utilice el comando `modify-event-subscription` de la AWS CLI. Incluya el siguiente parámetro obligatorio:

- `--subscription-name`

Example

El siguiente código activa `myeventsubscription`.

Para Linux, macOS o:Unix

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

En:Windows

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

Para modificar un evento de Amazon RDS, llame a la operación de la API de Amazon RDS `ModifyEventSubscription`. Incluya el siguiente parámetro obligatorio:

- `SubscriptionName`

Agregar un identificador de origen a una suscripción de notificación de eventos de Amazon RDS

Puede añadir un identificador de origen (el origen de Amazon RDS que genera el evento) a una suscripción existente.

Consola

Puede añadir o eliminar fácilmente identificadores de origen mediante la consola de Amazon RDS activándolos o desactivándolos al modificar una suscripción. Para obtener más información, consulte Modificación de una suscripción a notificaciones de eventos de Amazon RDS.

AWS CLI

Para agregar un identificador de origen a una suscripción a notificaciones de eventos de Amazon RDS, utilice el comando `add-source-identifier-to-subscription` de la AWS CLI. Incluya los siguientes parámetros obligatorios:

- `--subscription-name`
- `--source-identifier`

Example

En el siguiente ejemplo se añade el identificador de origen `mysqldb` a la suscripción `myrdseventsubscription`

Para Linux, macOS o:Unix

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

En:Windows

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Para añadir un identificador de origen a una suscripción a notificaciones de eventos de Amazon RDS, llame a la acción de la API de Amazon RDS `AddSourceIdentifierToSubscription`. Incluya los siguientes parámetros obligatorios:

- `SubscriptionName`
- `SourceIdentifier`

Quitar un identificador de origen de una suscripción de notificación de eventos de Amazon RDS

Puede eliminar un identificador de origen (el origen de Amazon RDS que genera el evento) de una suscripción si ya no desea recibir notificaciones de los eventos de ese origen.

Consola

Puede añadir o eliminar fácilmente identificadores de origen mediante la consola de Amazon RDS activándolos o desactivándolos al modificar una suscripción. Para obtener más información, consulte Modificación de una suscripción a notificaciones de eventos de Amazon RDS.

AWS CLI

Para eliminar un identificador de origen de una suscripción a notificaciones de eventos de Amazon RDS, utilice el comando `remove-source-identifier-from-subscription` de la AWS CLI. Incluya los siguientes parámetros obligatorios:

- `--subscription-name`
- `--source-identifier`

Example

En el siguiente ejemplo, se elimina el identificador de origen `mysqldb` de la suscripción `myrdseventsubscription`.

Para Linux, macOS o:Unix

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

En:Windows

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Para eliminar un identificador de origen de una suscripción a notificaciones de eventos de Amazon RDS, utilice el comando `RemoveSourceIdentifierFromSubscription` de la API de Amazon RDS. Incluya los siguientes parámetros obligatorios:

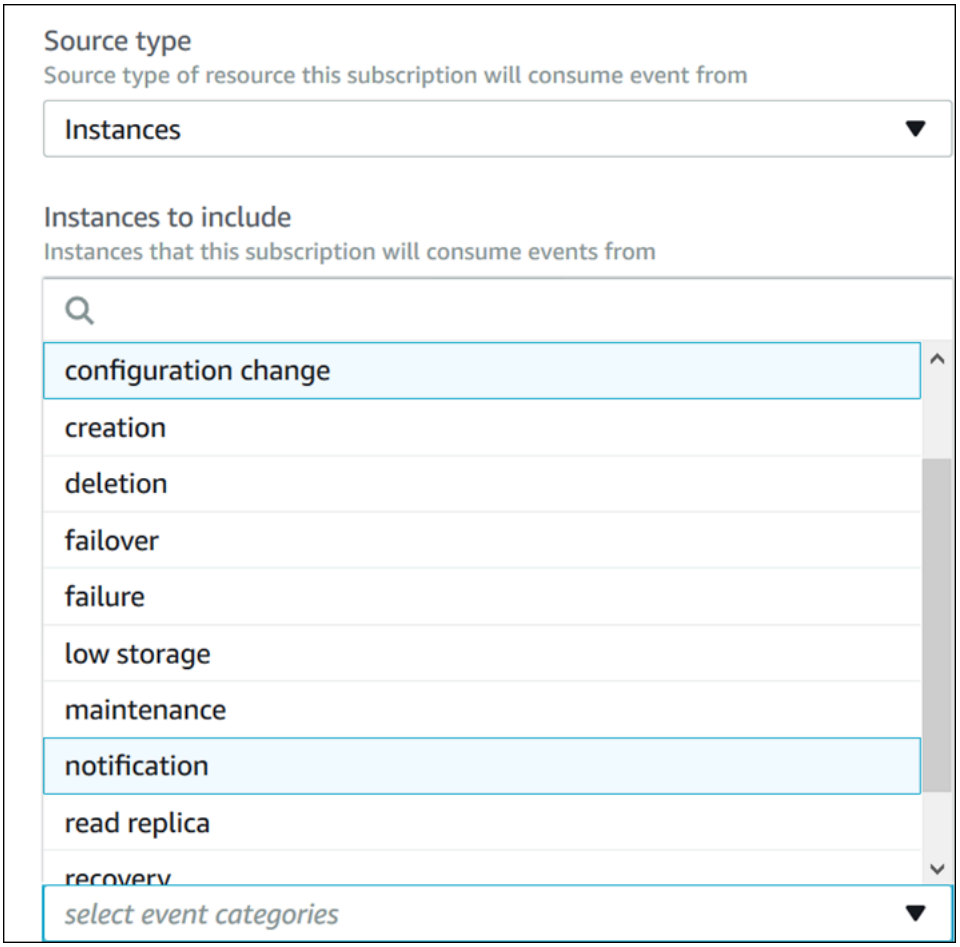
- `SubscriptionName`
- `SourceIdentifier`

Descripción de la lista de categorías de notificaciones de eventos de Amazon RDS

Todos los eventos de un tipo de recurso se agrupan en categorías. Para ver la lista de categorías disponibles, utilice los siguientes procedimientos.

Consola

Cuando se crea o modifica una suscripción a notificaciones de eventos, las categorías de eventos se muestran en la consola de Amazon RDS. Para obtener más información, consulte Modificación de una suscripción a notificaciones de eventos de Amazon RDS.



AWS CLI

Para obtener la lista de categorías de notificaciones de eventos de Amazon RDS, utilice el comando `describe-event-categories` de la AWS CLI. Este comando no tiene parámetros obligatorios.

Example

```
aws rds describe-event-categories
```

API

Para obtener la lista de categorías de notificaciones de eventos de Amazon RDS, utilice el comando `DescribeEventCategories` de la API de Amazon RDS. Este comando no tiene parámetros obligatorios.

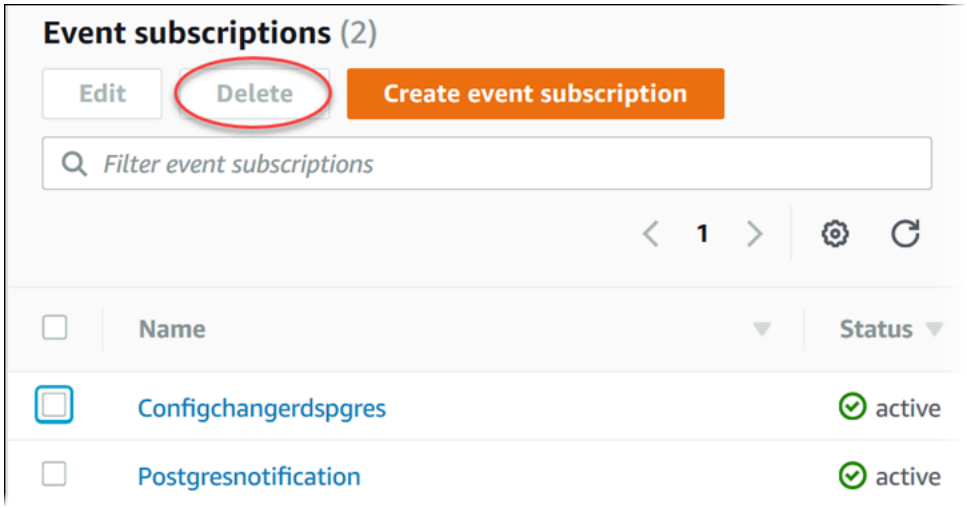
Eliminar una suscripción de notificación de eventos de Amazon RDS

Puede eliminar una suscripción cuando ya no la necesite. Los suscriptores del tema dejarán de recibir notificaciones de los eventos especificados en la suscripción.

Consola

Para eliminar una suscripción de notificación de eventos de Amazon RDS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación seleccione DB Event Subscriptions (Suscripciones a eventos de base de datos).
3. En el panel My DB Event Subscriptions (Mis suscripciones a eventos de base de datos), elija la suscripción que desea eliminar.
4. Elija Eliminar.
5. La consola de Amazon RDS indica que se está eliminando la suscripción.



AWS CLI

Para eliminar una suscripción a notificaciones de eventos de Amazon RDS, utilice el comando `delete-event-subscription` de la AWS CLI. Incluya el siguiente parámetro obligatorio:

- `--subscription-name`

Example

En el siguiente ejemplo se elimina la suscripción `myrdssubscription`.

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

API

Para eliminar una suscripción a notificaciones de eventos de Amazon RDS, utilice el comando `DeleteEventSubscription` de la API de RDS. Incluya el siguiente parámetro obligatorio:

- `SubscriptionName`

Creación de una regla que se desencadena en función de un evento Amazon Aurora

Al utilizar Amazon EventBridge, puede automatizar los servicios de AWS y responder a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios de recursos.

Temas

- Tutorial: Registrar el estado de una instancia de base de datos con Amazon EventBridge

Tutorial: Registrar el estado de una instancia de base de datos con Amazon EventBridge

En este tutorial puede crear una función de AWS Lambda que registre los cambios de estado de una instancia de . A continuación, puede crea una regla que ejecute la función cuando se produzca un cambio de estado de una instancia de base de datos de RDS existente. En el tutorial se asume que tiene una pequeña instancia de prueba en ejecución que puede apagar temporalmente.

> **Important**
>
> No realice este tutorial en una instancia de base de datos de producción en ejecución.

Temas

- Paso 1: Crear una función de AWS Lambda
- Paso 2: Crear una regla
- Paso 3: Probar la regla

Paso 1: Crear una función de AWS Lambda

Cree una función Lambda para registrar los eventos de cambio de estado. Especifique esta función cuando cree la regla.

Para crear una función Lambda

1. Abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Si es la primera vez que utiliza Lambda, aparecerá una página de bienvenida. Seleccione Get Started Now. De lo contrario, seleccione Create function (Crear función).

3. Elija Author from scratch.
4. En la página Create function (Crear función), proceda del modo siguiente:
   a. Introduzca un nombre y la descripción de la función Lambda. Por ejemplo, asigne un nombre a la función **`RDSInstanceStateChange`**.
   b. En Runtime (Tiempo de ejecución), seleccione Node.js 14x.
   c. En Architecture (Arquitectura), elija x86_64.
   d. En Execution role (Rol de ejecución), haga una de estas dos operaciones:
      - Elija Create a new role with basic Lambda permissions (Crear un nuevo rol con permisos básicos de Lambda).
      - En Existing role (Rol existente), elija Use an existing role (Usar un rol existente). Elija el rol que desee usar.
   e. Elija Create function (Crear función).
5. En la página RDSInstanceStateChange, haga lo siguiente:
   a. En Code source (Fuente del código), seleccione index.js.
   b. En el panel deindex.js, elimine el código existente.
   c. Escriba el código siguiente:

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

   d. Elija Deploy (Implementar).

Paso 2: Crear una regla

Cree una regla para ejecutar su función Lambda siempre que lance una instancia Amazon RDS.

Para crear la regla de EventBridge

1. Abra la consola de Amazon EventBridge en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas.
3. Elija Crear regla.

4. Escriba un nombre y una descripción de la regla. Por ejemplo, escriba **`RDSInstanceStateChangeRule`**.
5. Elija Rule with an event pattern (Regla con un patrón de evento) y, a continuación, elija Next (Siguiente).
6. En Origen del evento, elija Eventos de AWS o eventos de socios de EventBridge.
7. Desplácese hacia abajo en la sección Event pattern (Patrón de eventos).
8. En Origen del evento, elija Servicios de AWS.
9. En AWS service (Servicio de), elija Relational Database Service (RDS).
10. Para Event type (Tipo de evento), elija RDS DB Instance Event (Evento de instancia de base de datos RDS).
11. Deje el patrón de eventos predeterminado. A continuación, elija Next.
12. En Tipos de destino, seleccione Servicio de AWS.
13. En Select a target (Seleccione destino), elija Lambda function (Función de Lambda).
14. En Function (Función), seleccione la función Lambda que ha creado. A continuación, elija Next.
15. En Configure tags (Configurar etiquetas), elija Next (Siguiente).
16. Revise los pasos de la regla. A continuación, elija Create rule (Crear regla).

Paso 3: Probar la regla

Para probar su regla, cierre una instancia de base de datos de RDS. Después de esperar unos minutos a que la instancia se detenga, compruebe que se haya invocado la función Lambda.

Para probar la regla mediante la detención de una instancia de base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Detenga una instancia de base de datos de RDS.
3. Abra la consola de Amazon EventBridge en https://console.aws.amazon.com/events/.
4. En el panel de navegación, elija Rules (Reglas) y elija el nombre de la regla que ha creado.
5. En Detalles de la regla, seleccione Monitoreo.

   Se lo redirigirá a la consola de Amazon CloudWatch. Si no se le redirige, haga clic en Ver métricas en CloudWatch.
6. En All metrics (Todas las métricas), elija el nombre de la regla que creó.

El gráfico debe indicar que se ha invocado la regla.

7. En el panel de navegación, seleccione Log groups (Grupos de registro).
8. Seleccione el nombre del grupo de registro de su función de Lambda (/aws/lambda/***function-name***).
9. Elija el nombre del flujo de registro para ver los datos proporcionados por la función para la instancia que ha lanzado. Debería recibir un resultado similar al siguiente:

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

Para ver más ejemplos de eventos de RDS en formato JSON, consulte Información general de los eventos para Aurora.

10. (Opcional) Cuando haya terminado, puede abrir la consola de Amazon RDS y comenzar la instancia que ha lanzado.

Categorías y mensajes de eventos de Amazon RDS para Aurora

Amazon RDS genera un número significativo de eventos en categorías a las que puede suscribirse a través de la consola de Amazon RDS, la AWS CLI o la API.

Temas

- Eventos de clúster de bases de datos
- Eventos de instantánea de clúster de bases de datos
- Eventos de instancia de base de datos
- Eventos de grupo de parámetros de base de datos
- Eventos de grupo de seguridad de base de datos
- Eventos de grupo de particiones de bases de datos
- Eventos de RDS Proxy
- Eventos de implementación azul/verde

Eventos de clúster de bases de datos

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es un clúster de base de datos.

> **Note**
>
> No existen categorías de evento para Aurora Serverless en cuanto al tipo de evento de clúster de bases de datos. Los eventos de Aurora sin servidor van desde RDS-EVENT-0141 hasta RDS-EVENT-0149.

Categoría	ID de evento de RDS	Mensaje	Notas
cambio de configuración	RDS-EVENT-0016	Restablecer las credencia les maestras.	Ninguno
cambio de configuración	RDS-EVENT-0179	Los flujos de actividad de base de datos se inician	Para obtener más informaci ón, consulte Supervisión de Amazon Aurora con flujos

Categoría	ID de evento de RDS	Mensaje	Notas
		en el clúster de bases de datos	de actividad de la base de datos.
cambio de configuración	RDS-EVENT-0180	Flujos de actividad de la base de datos se detiene en el clúster de bases de datos.	Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.
creación	RDS-EVENT-0170	Se ha creado un clúster de bases de datos.	Ninguno
eliminación	RDS-EVENT-0171	Se ha eliminado un clúster de bases de datos.	Ninguno
conmutación por error	RDS-EVENT-0069	Ha fallado la conmutación por error del clúster. Compruebe el estado de las instancias del clúster e inténtelo de nuevo.	Ninguno
conmutación por error	RDS-EVENT-0070	Volver a promocionar el principal anterior: *nombre*.	Ninguno
conmutación por error	RDS-EVENT-0071	Se ha completado la conmutación por error en la instancia de base de datos: *nombre*.	Ninguno
conmutación por error	RDS-EVENT-0072	Se ha iniciado la misma conmutación por error de AZ en la instancia de base de datos: *nombre*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
conmutación por error	RDS-EVENT-0073	Se ha iniciado la conmutación por error entre AZ en la instancia de base de datos: *nombre*.	Ninguno
error	RDS-EVENT-0083	Amazon RDS no ha podido crear credenciales para acceder al bucket de Amazon S3 para el clúster de base de datos *nombre*. Esto se debe a que el rol de IAM de ingesta de instantáneas de S3 no se ha configurado correctamente en la cuenta o a que no se encuentra el bucket de Amazon S3 especificado. Consulte la sección de solución de problemas de la documentación de Amazon RDS para obtener más información.	Para obtener más información, consulte Migración física de MySQL con Percona XtraBackup y Amazon S3 .
failure	RDS-EVENT-0143	El clúster de base de datos no ha podido escalar de *unidades* a *unidades* por este motivo: *motivo*.	Escalado fallido para el clúster de bases de datos de Aurora Serverless.
error	RDS-EVENT-0354	No puede crear el clúster de base de datos porque los recursos son incompatibles. *mensaje*.	El *message* incluye detalles sobre el error.

Categoría	ID de evento de RDS	Mensaje	Notas
error	RDS-EVENT-0240	El clúster de base de datos no se puede crear porque los límites de recursos son insuficientes. *mensaje*.	El *message* incluye detalles sobre el error.
error	RDS-EVENT-0387	No se han podido particion ar las instancias de base de datos en el *nombre* del clúster de base de datos para la aplicación de parches.	Se ha producido un error en las actualizaciones del sistema operativo de las instancias de base de datos del clúster de bases de datos.
Conmutaci ón por error global	RDS-EVENT-0181	Se ha iniciado la transició n al clúster de base de datos *nombre* de la región *nombre*.	Este evento es para una operación de transició n (antes denominada “conmutación por error planificada administrada”). El proceso se puede retrasar porque están en ejecución otras operacion es en el clúster de bases de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
Conmutación por error global	RDS-EVENT-0182	El clúster de base de datos principal *nombre* anterior en la región *nombre* se ha cerrado correctamente.	Este evento es para una operación de transición (antes denominada "conmutación por error planificada administrada"). La antigua instancia principal de la base de datos global no acepta escrituras. Todos los volúmenes están sincronizados.
Conmutación por error global	RDS-EVENT-0183	Esperando a que se sincronicen los datos entre los miembros del clúster global. Retrasos actuales debidos al clúster de base de datos principal: *reason*.	Este evento es para una operación de transición (antes denominada "conmutación por error planificada administrada"). Se produce un retraso de replicación durante la fase de sincronización de la conmutación por error de base de datos global.
Conmutación por error global	RDS-EVENT-0184	El clúster de base de datos principal *nombre* anterior en la región *nombre* se ha promovido correctamente.	Este evento es para una operación de transición (antes denominada "conmutación por error planificada administrada"). La topología de volúmenes de la base de datos global se restablece con el nuevo volumen principal.

Categoría	ID de evento de RDS	Mensaje	Notas
Conmutación por error global	RDS-EVENT-0185	Ha finalizado la transición al clúster de base de datos *nombre* de la región *nombre*.	Este evento es para una operación de transición (antes denominada “conmutación por error planificada administrada”). Se ha finalizado la transición de la base de datos global en el clúster de la base de datos principal. Las réplicas pueden tardar mucho en ponerse en línea una vez completada la conmutación por error.
Conmutación por error global	RDS-EVENT-0186	Se ha cancelado la transición al clúster de base de datos *nombre* de la región *nombre*.	Este evento es para una operación de transición (antes denominada “conmutación por error planificada administrada”).
Conmutación por error global	RDS-EVENT-0187	Se ha producido un error en la transición al clúster de base de datos *nombre* de la región *nombre*.	Este evento es para una operación de transición (antes denominada “conmutación por error planificada administrada”).
Conmutación por error global	RDS-EVENT-0238	Se ha completado la conmutación por error global en el clúster de base de datos *nombre* de la región *nombre*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
Conmutación por error global	RDS-EVENT-0239	Ha fallado la conmutación por error global en el clúster de base de datos *nombre* de la región *nombre*.	Ninguno
Conmutación por error global	RDS-EVENT-0240	Se ha iniciado la resincronización de los miembros del clúster de base de datos *nombre* de la región *nombre* después de una conmutación por error global.	Ninguno
Conmutación por error global	RDS-EVENT-0241	Ha finalizado la resincronización de los miembros del clúster de base de datos *nombre* de la región *nombre* después de una conmutación por error global.	Ninguno
Conmutación por error global	RDS-EVENT-0397	Aurora terminó de cambiar el nombre de DNS al que se resuelve el punto de conexión del escritor global.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
Conmutación por error global	RDS-EVENT-0423	En espera de la sincronización de datos con el clúster de base de datos de destino. Retraso actual del clúster de base de datos de destino con respecto al clúster de base de datos principal: `%s`.	Este evento es para una operación de transición (antes denominada “conmutación por error planificada administrada”). Se produce un retraso de replicación durante la fase de sincronización de la conmutación por error de base de datos global.
mantenimiento	RDS-EVENT-0156	El clúster de bases de datos tiene disponible una actualización de versiones secundarias de motor de base de datos.	Ninguno
mantenimiento	RDS-EVENT-0173	Se ha actualizado la versión del motor de clústeres de bases de datos.	Se ha llevado a cabo la aplicación de parches al clúster de bases de datos.
mantenimiento	RDS-EVENT-0174	El clúster de base de datos tiene un estado que no se puede actualizar.	Ninguno
mantenimiento	RDS-EVENT-0176	Se ha actualizado la versión principal del motor de clústeres de bases de datos.	Ninguno
mantenimiento	RDS-EVENT-0177	La actualización del clúster de base de datos está en curso.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-0286	Se ha iniciado la actualización de la versión del motor de clústeres de bases de datos *número_versión* . El clúster permanece en línea.	Ninguno
mantenimiento	RDS-EVENT-0287	Se ha detectado el requisito de actualización del sistema operativo.	Ninguno
mantenimiento	RDS-EVENT-0288	Se ha iniciado la actualización del sistema operativo del clúster.	Ninguno
mantenimiento	RDS-EVENT-0289	Se ha completado la actualización del sistema operativo del clúster.	Ninguno
mantenimiento	RDS-EVENT-0290	Se ha aplicado un parche al clúster de base de datos: versión de origen *número_versión* => *nuevo_número_versión* .	Ninguno
mantenimiento	RDS-EVENT-0363	Preparación de la actualización en curso: *clúster_name*	Se han iniciado las comprobaciones previas de actualización para el clúster de base de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-0388	Inicio de parches sin conexión para las instancias de base de datos en la partición *nombre*/*nombre* del clúster de bases de datos *nombre*: *partición_n* .	Inicio de las actualizaciones del sistema operativo de las instancias de base de datos del clúster de bases de datos.
mantenimiento	RDS-EVENT-0389	No hemos podido actualizar el sistema operativo de su clúster de bases de datos. Puede esperar al siguiente periodo de mantenimiento o actualizar el sistema operativo del clúster de bases de datos manualmente.	Ninguno
mantenimiento	RDS-EVENT-0424	El clúster de base de datos *nombre* está ejecutando la versión *versión*, que es superior a la versión de actualización objetivo *versión* para el clúster global. No recomendamos tener un clúster secundario en una versión posterior a la del clúster global, ya que puede provocar problemas durante la conmutación por error o la transición. Considere actualizar el clúster global para que coincida.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
notificación	RDS-EVENT-0076	No se ha podido migrar de *nombre* a *nombre*. Motivo: *motivo*.	Error en la migración a un clúster de bases de datos de Aurora.
notificación	RDS-EVENT-0077	No se ha podido convertir *nombre*.*nombre* a InnoDB. Motivo: *motivo*.	Ha fallado un intento de convertir una tabla de la base de datos de origen a InnoDB durante la migración a un clúster de bases de datos de Aurora.
notificación	RDS-EVENT-0085	No se ha podido actualiza r el clúster de base de datos *nombre* porque la instancia *nombre* tiene el estado *nombre*. Resuelva el problema o elimine la instancia e inténtelo de nuevo.	Error al intentar aplicar parches al clúster de bases de datos de Aurora. Compruebe el estado de la instancia, resuelva el problema e inténtelo de nuevo. Para obtener más información, consulte Mantenimiento de un clúster de base de datos de Amazon Aurora.
notification	RDS-EVENT-0141	Se ha escalado el clúster de base de datos de *unidades* a *unidades* por este motivo: *motivo*.	Escalado iniciado para el clúster de bases de datos de Aurora Serverless.
notification	RDS-EVENT-0142	El clúster de base de datos se ha escalado de *unidades* a *unidades*.	Escalado completado para el clúster de bases de datos de Aurora Serverles s.

Categoría	ID de evento de RDS	Mensaje	Notas
notification	RDS EVENT-0144	El clúster de base de datos se ha pausado.	Se ha iniciado una pausa automática para el clúster de base de datos Aurora sin servidor.
notificación	RDS-EVENT-0145	El clúster de base de datos está pausado.	El clúster de base de datos de Aurora Serverless se ha pausado.
notificación	RDS-EVENT-0146	Pausa cancelada para el clúster de base de datos.	Se ha cancelado la pausa para el clúster de base de datos de Aurora Serverless.
notificación	RDS-EVENT-0147	Se ha reiniciado el clúster de base de datos.	Se ha iniciado una operación de reanudación para el clúster de base de datos de Aurora Serverless.
notificación	RDS-EVENT-0148	Se ha reanudado el clúster de base de datos.	Se ha completado la operación de reanudación para el clúster de base de datos de Aurora Serverless.
notificación	RDS-EVENT-0149	El clúster de base de datos se ha escalado de *`unidades`* a *`unidades`*, pero el escalado no ha sido perfecto por este motivo: *`motivo`*.	Escalación correcta completada con la opción de fuerza para el clúster de bases de datos de Aurora Serverless. Las conexiones se deben haber interrumpido según lo solicitado.

Categoría	ID de evento de RDS	Mensaje	Notas
notification	RDS-EVENT-0150	Se ha detenido el clúster de base de datos.	Ninguno
notificación	RDS-EVENT-0151	Se ha iniciado el clúster de base de datos.	Ninguno
notificación	RDS-EVENT-0152	Ha fallado la detención del clúster de base de datos.	Ninguno
notificación	RDS-EVENT-0153	Se inicia el clúster de base de datos porque se ha superado el tiempo máximo de retención permitido.	Ninguno
notificación	RDS-EVENT-0172	Se ha cambiado el nombre del clúster de *`nombre`* a *`nombre`*.	Ninguno
notificación	RDS-EVENT-0234	Error en la tarea de exportación.	Error en la tarea de exportación del clúster de bases de datos.
notificación	RDS-EVENT-0235	Se ha cancelado la tarea de exportación.	Se ha cancelado la tarea de exportación del clúster de bases de datos.
notificación	RDS-EVENT-0236	Se ha completado la tarea de exportación.	Se ha completado la tarea de exportación del clúster de bases de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
notificación	RDS-EVENT-0386	Las instancias de base de datos del clúster de bases de datos *nombre* se han particionado: *lista_de_particiones* . El clúster de bases de datos está en línea.	Se han completado las actualizaciones del sistema operativo de las instancias de base de datos del clúster de bases de datos.
notificación	RDS-EVENT-0426	RDS no puede configurar el clúster de base de datos *nombre* como origen de replicación debido a transacciones inactivas o de larga duración. Espere a que se completen las transacciones o cancélelas y vuelva a intentarlo.	Ninguno

Eventos de instantánea de clúster de bases de datos

En la siguiente tabla se muestra la categoría de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es una instantánea de clúster de base de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
copia de seguridad	RDS-EVENT-0074	Crear una instantánea manual del clúster.	Ninguno
copia de seguridad	RDS-EVENT-0075	Se ha creado una instantánea manual del clúster.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
notificación	RDS-EVENT-0162	Error en la tarea de exportación de la instantán ea del clúster.	Ninguno
notificación	RDS-EVENT-0163	Se ha cancelado la tarea de exportación de instantán ea del clúster.	Ninguno
notificación	RDS-EVENT-0164	Se ha completado la tarea de exportación de instantán ea del clúster.	Ninguno
copia de seguridad	RDS-EVENT-0168	Creación de instantáneas de clúster automatizadas.	Ninguno
copia de seguridad	RDS-EVENT-0169	Se ha creado una instantán ea de clúster automatizada.	Ninguno

Eventos de instancia de base de datos

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es una instancia de base de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
disponibilidad	RDS-EVENT-0004	Instancia de base de datos cerrada.	Ninguno
disponibilidad	RDS-EVENT-0006	Se ha reiniciado la instancia de base de datos.	Ninguno
disponibilidad	RDS-EVENT-0022	Error al reiniciar mysql: *mensaje*.	Se ha producido un error al reiniciar Aurora MySQL o RDS para MariaDB.

Categoría	ID de evento de RDS	Mensaje	Notas
disponibilidad	RDS-EVENT-0419	Amazon RDS no ha podido acceder a la clave de cifrado de KMS para el *nombre* de la instancia de la base de datos. Esta base de datos pasará a un estado inaccesible. Consulte la sección de solución de problemas de la documentación de Amazon RDS para obtener más información.	Ninguno
búsqueda de datos anteriores	RDS-EVENT-0131	El periodo de búsqueda de datos anteriores real es inferior al periodo de búsqueda de datos anteriores de destino que especificó. Considere reducir el número de horas del periodo de búsqueda de datos anteriores de destino.	Para obtener más información sobre la búsqueda de datos anteriores, consulte Búsqueda de datos anteriores de un clúster de base de datos de Aurora.
backtrack	RDS EVENT-0132	El periodo de búsqueda de datos anteriores real es el mismo que el periodo de búsqueda de datos de destino.	Ninguno
cambio de configuración	RDS-EVENT-0011	Actualizado para usar el DBParameterGroup *name*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
cambio de configuración	RDS-EVENT-0012	Se está aplicando la modificación a la clase de instancia de base de datos.	Ninguno
cambio de configuración	RDS-EVENT-0014	Ha finalizado la aplicación de la modificación a la clase de instancia de base de datos.	Ninguno
cambio de configuración	RDS-EVENT-0017	Ha finalizado la aplicación de la modificación al almacenamiento asignado.	Ninguno
cambio de configuración	RDS-EVENT-0025	Ha finalizado la aplicación de la modificación para convertirla en una instancia de base de datos multi-AZ.	Ninguno
cambio de configuración	RDS-EVENT-0029	Ha finalizado la aplicación de la modificación para convertirla en una instancia de base de datos (single-AZ) estándar.	Ninguno
cambio de configuración	RDS-EVENT-0033	Hay *número* usuarios que coinciden con el nombre de usuario maestro; solo se restablece el que no esté vinculado a un host específico.	Ninguno
cambio de configuración	RDS-EVENT-0067	No se ha podido restablecer la contraseña. Información de error: *mensaje*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
cambio de configuración	RDS-EVENT-0078	El intervalo de monitorización ha cambiado a *número*.	Se ha cambiado la configuración de monitorización mejorada.
cambio de configuración	RDS-EVENT-0092	Se ha terminado de actualizar el grupo de parámetros de base de datos.	Ninguno
creation	RDS-EVENT-0005	Instancia de base de datos creada.	Ninguno
eliminación	RDS-EVENT-0003	Se ha eliminado la instancia de base de datos.	Ninguno
error	RDS-EVENT-0035	Instancia de base de datos que pasa a *estado*. *mensaje*.	La instancia de base de datos tiene parámetros no válidos. Por ejemplo, no se ha podido iniciar la instancia de base de datos porque un parámetro relacionado con la memoria tiene un valor demasiado alto para esta clase de instancia, por lo que debería modificar el parámetro de memoria y reiniciar la instancia de base de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
error	RDS-EVENT-0036	Instancia de base de datos *estado*. *mensaje*.	La instancia de base de datos está en una red incompatible. Algunos de los ID de subred especific ados no son válidos o no existen.
error	RDS-EVENT-0079	Amazon RDS no ha podido crear credenciales para mejorar la monitorización y se ha deshabilitado la característica. Es probable que esto se deba a que rds-monitoring-role no está presente pero está configurado correctamente en su cuenta. Consulte la sección de solución de problemas de la documenta ción de Amazon RDS para obtener más información.	La monitorización mejorada no se puede activar sin el rol de IAM de monitoriz ación mejorada. Para obtener información acerca de la creación del rol de IAM, consulte Para crear un rol de IAM para el monitoreo mejorado de Amazon RDS.

Categoría	ID de evento de RDS	Mensaje	Notas
error	RDS-EVENT-0080	Amazon RDS no ha podido configurar la monitorización mejorada en su instancia: *nombre* y se ha deshabilitado esta característica. Es probable que esto se deba a que rds-monitoring-role no está presente pero está configurado correctamente en su cuenta. Consulte la sección de solución de problemas de la documentación de Amazon RDS para obtener más información.	La monitorización mejorada se ha desactivado porque ha surgido un error al realizar el cambio de configuración. Es probable que el rol de IAM de monitorización mejorada se haya configurado incorrectamente. Para obtener información acerca de la creación del rol de IAM de monitorización mejorada, consulte Para crear un rol de IAM para el monitoreo mejorado de Amazon RDS.
failure	RDS-EVENT-0082	Amazon RDS no ha podido crear credenciales para acceder al bucket de Amazon S3 para la instancia de base de datos *nombre*. Esto se debe a que el rol de IAM de ingesta de instantáneas de S3 no se ha configurado correctamente en la cuenta o a que no se encuentra el bucket de Amazon S3 especificado. Consulte la sección de solución de problemas de la documentación de Amazon RDS para obtener más información.	Aurora no ha podido copiar los datos de copia de seguridad de un bucket de Amazon S3. Es probable que los permisos que tiene Aurora para obtener acceso al bucket de Amazon S3 se hayan configurado incorrectamente. Para obtener más información, consulte Migración física de MySQL con Percona XtraBackup y Amazon S3 .

Categoría	ID de evento de RDS	Mensaje	Notas
error	RDS-EVENT-0240	La instancia de base de datos no se puede crear porque los límites de recursos son insuficientes. *mensaje*.	El *message* incluye detalles sobre el error.
error	RDS-EVENT-0418	Amazon RDS no puede acceder a la clave de cifrado de KMS del *nombre* de la instancia de base de datos. Es probable que esto se deba a que la clave está desactivada o a que Amazon RDS no puede acceder a ella. Si esto continúa, la base de datos pasará a un estado inaccesible. Consulte la sección de solución de problemas de la documenta ción de Amazon RDS para obtener más información.	Ninguno
error	RDS-EVENT-0420	Amazon RDS ahora puede acceder correctamente a la clave de cifrado de KMS del *nombre* de la instancia de base de datos.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
bajo almacenamiento	RDS-EVENT-0007	Se ha agotado el almacenamiento asignado. Asigne almacenamiento adicional para resolver el problema.	Se ha agotado el almacenamiento asignado a la instancia de base de datos. Para solucionar este problema, asigne almacenamiento adicional a la instancia de base de datos. Para obtener más información, consulte las Preguntas frecuentes de RDS. El espacio de almacenamiento de una instancia de base de datos se puede monitorizar con la métrica Free Storage Space.
low storage	RDS-EVENT-0089	La capacidad de almacenamiento libre de la instancia de base de datos: *nombre* es baja un *porcentaje* del almacenamiento aprovisionado [Almacenamiento aprovisionado: *tamaño*, Almacenamiento gratuito: *tamaño*]. Puede que desee aumentar el almacenamiento aprovisionado para solucionar este problema.	La instancia de base de datos ha consumido más del 90 % del almacenamiento asignado. El espacio de almacenamiento de una instancia de base de datos se puede monitorizar con la métrica Free Storage Space.

Categoría	ID de evento de RDS	Mensaje	Notas
bajo almacenamiento	RDS-EVENT-0227	El almacenamiento de su clúster de Aurora es peligrosamente bajo y solo quedan *cantidad* terabytes. Tome medidas para reducir la carga de almacenamiento en el clúster.	El subsistema de almacenamiento de Aurora se está quedando sin espacio.
mantenimiento	RDS-EVENT-0026	Se aplican parches fuera de línea a la instancia de base de datos.	Se está realizando el mantenimiento sin conexión de la instancia de base de datos. La instancia de base de datos no está disponible en este momento.
maintenance	RDS-EVENT-0027	Ha finalizado la aplicación de parches fuera de línea a la instancia de base de datos.	Ha finalizado el mantenimiento sin conexión de la instancia de base de datos. La instancia de base de datos ya está disponible.
maintenance	RDS-EVENT-0047	Se han aplicado parches a la instancia de base de datos.	Ninguno
mantenimiento	RDS-EVENT-0155	La instancia de base de datos tiene disponible una actualización de versiones secundarias de motor de base de datos.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-0178	La actualización de la instancia de base de datos está en curso.	Ninguno
mantenimiento	RDS-EVENT-0422	RDS reemplazará el host del *nombre* de la instancia de base de datos debido a una acción de mantenimiento pendiente.	Ninguno
notification	RDS-EVENT-0044	*message*	Se trata de una notificación emitida por el operador. Para obtener más información, consulte el mensaje del evento.
notification	RDS-EVENT-0048	Se debe retrasar la actualización del motor de base de datos, ya que esta instancia tiene réplicas de lectura que deben actualizarse primero.	Se ha retrasado la aplicación de parches a la instancia de base de datos.
notification	RDS-EVENT-0087	Se ha detenido la instancia de base de datos.	Ninguno
notificación	RDS-EVENT-0088	Se ha iniciado la instancia de base de datos.	Ninguno
notificación	RDS-EVENT-0365	Se han actualizado los archivos de zonas horarias. Reinicie la instancia de RDS para que los cambios surtan efecto.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
notificación, sin servidor	RDS-EVENT-0370	Se inició una pausa para la instancia de base de datos.	Se inició un nuevo intento de pausar una instancia de base de datos Aurora Serverless v2 inactiva.
notificación, sin servidor	RDS-EVENT-0371	Pausa cancelada para la instancia de base de datos.	No se pudo detener una instancia de base de datos Aurora Serverless v2 inactiva debido a la carga de trabajo.
notificación, sin servidor	RDS-EVENT-0372	Se ha pausado correctam ente la instancia de base de datos.	Se ha pausado la instancia de base de datos Aurora Serverless v2.
notificación, sin servidor	RDS-EVENT-0373	Se ha iniciado la reanudaci ón de la instancia de base de datos.	La instancia de base de datos Aurora Serverless v2 ha comenzado a reanudars e debido a una nueva carga de trabajo o a una actividad administrativa o de mantenimiento.
notificación, sin servidor	RDS-EVENT-0374	Se ha reanudado correctamente la instancia de base de datos.	Se ha reanudado la instancia de base de datos Aurora Serverless v2.

Categoría	ID de evento de RDS	Mensaje	Notas
notificación	RDS-EVENT-0385	Se ha actualizado la topología del clúster.	Hay cambios de DNS en el clúster de base de datos para la instancia de base de datos. Esto incluye cuando se añaden o eliminan nuevas instancias de base de datos, o cuando se produce una conmutación por error.
notificación, base de datos global	RDS-EVENT-0390	El intento de bloquear las escrituras del clúster de base de datos *cluster_id* en la región *region_id* se ha realizado correctamente.	Aurora ha comenzado a bloquear las escrituras en la capa de almacenamiento como preparación para la transición o conmutación por error de una base de datos global de Aurora.
notificación, base de datos global	RDS-EVENT-0391	Se ha agotado el tiempo de espera del intento de bloquear las escrituras del clúster de base de datos *cluster_id* en la región *region_id* .	Aurora no ha podido bloquear las escrituras en la capa de almacenamiento como preparación para la transición o conmutación por error de una base de datos global de Aurora. La transición o la conmutación por error continuarán, pero es posible que necesite recuperar los datos escritos recientemente de la instantánea del clúster principal original.

Categoría	ID de evento de RDS	Mensaje	Notas
read replica	RDS-EVENT-0045	Se ha detenido la replicación.	Este mensaje aparece cuando hay un error durante la replicación. Para determinar el tipo de error, consulte Solución de problemas de una réplica de lectura de MySQL.
réplica de lectura	RDS-EVENT-0046	Se ha reanudado la replicación de la réplica de lectura.	Este mensaje aparece cuando se crea por primera vez una réplica de lectura o como mensaje de monitoreo que confirma que la replicación funciona correctamente. Si este mensaje es posterior a una notificación `RDS-EVENT-0045` , significa que la replicación se ha reanudado después de detenerla o tras producirse un error.
réplica de lectura	RDS-EVENT-0057	Se ha interrumpido la transmisión de la replicación.	Ninguno
recuperación	RDS-EVENT-0020	Se ha iniciado la recuperación de la instancia de base de datos. El tiempo de recuperación dependerá de la cantidad de datos que deban recuperarse.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
recuperación	RDS-EVENT-0021	Ha finalizado la recuperación de la instancia de base de datos.	Ninguno
recuperación	RDS-EVENT-0023	Solicitud de instantánea emergente: *mensaje*.	Se ha solicitado una copia de seguridad manual, pero Amazon RDS está creando una instantánea de base de datos. Envíe de nuevo la solicitud cuando Amazon RDS haya terminado la instantánea de base de datos.
recovery	RDS-EVENT-0052	Se ha iniciado la recuperación de instancias multi-AZ.	El tiempo de recuperación dependerá de la cantidad de datos que deban recuperarse.
recovery	RDS-EVENT-0053	Se ha completado la recuperación de instancias de multi-AZ. Conmutación por error o activación pendientes.	Este mensaje indica que Amazon RDS ha preparado la instancia de base de datos para iniciar una conmutación por error a la instancia secundaria si es necesario.

Categoría	ID de evento de RDS	Mensaje	Notas
recuperación	RDS-EVENT-0361	Se ha iniciado la recuperación de la instancia de base de datos en espera.	La instancia de base de datos en espera se reconstruye durante el proceso de recuperación. El rendimiento de la base de datos se ve afectado durante el proceso de recuperación.
recuperación	RDS-EVENT-0362	Se ha completado la recuperación de la instancia de base de datos en espera.	La instancia de base de datos en espera se reconstruye durante el proceso de recuperación. El rendimiento de la base de datos se ve afectado durante el proceso de recuperación.
restauración	RDS-EVENT-0019	Se ha restaurado de la instancia de base de datos *`nombre`* a *`nombre`*.	La instancia de base de datos se ha restaurado a partir de una copia de seguridad de un momento dado.

Categoría	ID de evento de RDS	Mensaje	Notas
Creación de parches de seguridad	RDS-EVENT-0230	Hay disponible una actualización del sistema operativo para su instancia de base de datos. Para obtener información acerca de la aplicació n de actualizaciones, consulte «Mantenimiento de una instancia de base de datos» en la Guía del usuario de RDS.	Hay disponible una nueva versión secundari a del parche del sistema operativo para su instancia de base de datos. Para obtener más informaci ón acerca de cómo se aplican las actualizaciones, consulte Actualizaciones del sistema operativo para clústeres de base de datos de Aurora.
mantenimi ento	RDS-EVENT-0425	Amazon RDS no puede realizar la actualización del sistema operativo porque no hay direccion es IP disponibles en las subredes especific adas. Elija subredes con direcciones IP disponibles e inténtelo de nuevo.	Ninguno
mantenimi ento	RDS-EVENT-0429	Amazon RDS no puede realizar la actualización del sistema operativo debido a la capacidad insuficiente disponible para el tipo de instancia *tipo* en la zona de disponibilidad *zona*	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-0501	El certificado de servidor de la instancia de base de datos de Amazon RDS requiere la rotación a través de una acción de mantenimiento pendiente.	El certificado de servidor de la instancia de base de datos requiere la rotación a través de una acción de mantenimiento pendiente. Amazon RDS reinicia la base de datos durante este mantenimiento para completar la rotación de certificados. Para programar este mantenimiento, vaya a la pestaña Mantenimiento y copias de seguridad y elija Solicitar ahora o Programar para el siguiente periodo de mantenimiento. Si el cambio no está programado, Amazon RDS lo aplica automáticamente en el periodo de mantenimiento en la fecha de aplicación automática que se muestra en la acción de mantenimiento.

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-052	Amazon RDS ha programado una rotación de certificados de servidor para la instancia de base de datos durante el siguiente periodo de mantenimiento. Este mantenimiento requerirá el reinicio de una base de datos.	Ninguno

Eventos de grupo de parámetros de base de datos

En la siguiente tabla se muestra la categoría de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es un grupo de parámetros de base de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
cambio de configuración	RDS-EVENT-0037	Se actualizó el parámetro *`nombre`* por *`valor`* con el método de aplicación *`método`*.	Ninguno

Eventos de grupo de seguridad de base de datos

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es un grupo de seguridad de base de datos.

> **Note**
>
> Los grupos de seguridad de base de datos son recursos de EC2-Classic. EC2-Classic se retirará el 15 de agosto de 2022. Si todavía no ha migrado de EC2-Classic a una VPC, le

recomendamos que migre lo antes posible. Para obtener más información, consulte el tema Migrar de EC2-Classic a una VPC en la guía del usuario de Amazon EC2 y la publicación del blog EC2-Classic Networking is Retiring – Here's How to Prepare.

Categoría	ID de evento de RDS	Mensaje	Notas
cambio de configuración	RDS-EVENT-0038	Se ha aplicado el cambio al grupo de seguridad.	Ninguno
error	RDS-EVENT-0039	Revocar la autorización como *usuario*.	El grupo de seguridad propiedad de *usuario* no existe. Se ha revocado la autorización para el grupo de seguridad porque no es válida.

Eventos de grupo de particiones de bases de datos

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es un grupo de particiones de bases de datos.

Categoría	ID de evento de RDS	Mensaje	Notas
error	RDS-EVENT-0324	Error de trabajo de carga de datos.	El trabajo de carga de datos ha fallado por el motivo que aparece en el mensaje de error.
mantenimiento	RDS-EVENT-0302	Su acción es necesaria para finalizar un trabajo de división de partición pendiente, *job_ID*, para el subclúster con ID *número*	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
		en el grupo de particiones de bases de datos *nombre*. Para completar la operación, invoque el SQL: `SELECT rds_aurora.limitless_finalize_split_shard( job_ID);`	
mantenimiento	RDS-EVENT-0303	La finalización del trabajo de división de partición *job_ID* se ha iniciado para el subclúster con ID *número* en el grupo de particiones de bases de datos *nombre*.	Ninguno
mantenimiento	RDS-EVENT-0304	El trabajo de división de partición *job_ID* se ha completado correctamente para el subclúster con ID *número* en el grupo de particiones de bases de datos *nombre*. Se ha agregado una nueva partición con ID de subclúster *número* al grupo de particiones de base de datos *nombre*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-0305	Se ha producido un error en el trabajo de división de partición *job_ID* para el subclúster con ID *número* en el grupo de particiones de bases de datos *nombre*.	Ninguno
mantenimiento	RDS-EVENT-0366	Se ha iniciado el trabajo de división de partición *job_ID* para el subclúster con ID *número* en el grupo de particiones de bases de datos *nombre*.	Ninguno
mantenimiento	RDS-EVENT-0367	El trabajo de agregar un enrutador *job_ID* se ha iniciado en el grupo de particiones de bases de datos *nombre*.	Ninguno
mantenimiento	RDS-EVENT-0368	El trabajo de agregar un enrutador *job_ID* se ha completado correctamente. Se ha agregado un nuevo enrutador con ID de subclúster *número* al grupo de particiones de bases de datos *nombre*.	Ninguno
mantenimiento	RDS-EVENT-0369	Se ha producido un error en el trabajo de adición del enrutador *job_ID* en el grupo de particiones de bases de datos *nombre*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
mantenimiento	RDS-EVENT-0394	Se ha cancelado el trabajo de adición del enrutador *job_ID* en el grupo de particiones de bases de datos *nombre*.	Ninguno
mantenimiento	RDS-EVENT-0395	Se ha cancelado el trabajo de división de partición *job_ID* en el grupo de particiones de bases de datos *nombre*.	Ninguno
notificación	RDS-EVENT-0321	Inicialización de la infraestructura para el trabajo de carga de datos.	Ninguno
notificación	RDS-EVENT-0322	El trabajo de carga de datos está en curso.	Ninguno
notificación	RDS-EVENT-0323	El trabajo de carga de datos se ha completado correctamente.	Ninguno
notificación	RDS-EVENT-0325	Cancelando el trabajo de carga de datos según la solicitud del cliente.	Ninguno
notificación	RDS-EVENT-0326	Se ha cancelado el trabajo de carga de datos.	Ninguno

Eventos de RDS Proxy

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es una instancia de RDS Proxy.

Categoría	ID de evento de RDS	Mensaje	Notas
cambio de configuración	RDS-EVENT-0204	RDS ha modificado el proxy de la base de datos *nombre*.	Ninguno
cambio de configuración	RDS-EVENT-0207	RDS ha modificado el punto de conexión del proxy de la base de datos *nombre*.	Ninguno
cambio de configuración	RDS-EVENT-0213	RDS ha detectado la adición de la instancia de base de datos y la ha agregado automáticamente al grupo de destino del proxy de la base de datos *nombre*.	Ninguno
cambio de configuración	RDS-EVENT-0214	RDS ha detectado la eliminación de la instancia de base de datos *nombre* y la ha borrado automátic amente del grupo de destino *nombre* del proxy de la base de datos *nombre*.	Ninguno
cambio de configuración	RDS-EVENT-0215	RDS ha detectado la eliminación del clúster de base de datos *nombre* y la ha borrado automátic amente del grupo de destino *nombre* del proxy de la base de datos *nombre*.	Ninguno

Categoría	ID de evento de RDS	Mensaje	Notas
creación	RDS-EVENT-0203	RDS ha creado el proxy de la base de datos *nombre*.	Ninguno
creación	RDS-EVENT-0206	RDS ha creado el punto de conexión *nombre* del para el proxy de la base de datos *nombre*.	Ninguno
eliminación	RDS-EVENT-0205	RDS ha eliminado el proxy de la base de datos *nombre*.	Ninguno
eliminación	RDS-EVENT-0208	RDS ha eliminado el punto de conexión *nombre* del proxy de la base de datos *nombre*.	Ninguno
error	RDS-EVENT-0243	RDS no ha podido aprovisionar capacidad para el proxy *nombre* porque no hay suficientes direcciones IP disponibles en las subredes: *nombre*. Para resolver el problema, asegúrese de que sus subredes tengan el número mínimo de direcciones IP sin usar, tal como se recomienda en la documentación de RDS Proxy.	Para determinar el número recomendado para la clase de instancia, consulte Planificación de la capacidad de direcciones IP.

Categoría	ID de evento de RDS	Mensaje	Notas
error	RDS-EVENT-0275	RDS ha limitado algunas conexiones al proxy de base de datos *nombre*. El número de solicitudes de conexión simultáne as del cliente al proxy ha superado el límite.	Ninguno

Eventos de implementación azul/verde

En la siguiente tabla, se muestra la categoría de eventos y una lista de eventos cuando el tipo de origen es una implementación azul/verde.

Para obtener más información acerca de las implementaciones azul/verde, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Categoría	ID de evento de Amazon RDS	Mensaje	Notas
creación	RDS-EVENT-0244	Se han completado las tareas de implementación azul/verde. Puede realizar más modificaciones en las bases de datos del entorno verde o cambiar la implementación.	Ninguno
error	RDS-EVENT-0245	Se ha producido un error al crear la implementación azul/verde por *motivo*.	Ninguno
eliminación	RDS-EVENT-0246	Se ha eliminado la implementación azul/verde.	Ninguno

Categoría	ID de evento de Amazon RDS	Mensaje	Notas
notificación	RDS-EVENT-0247	Se ha iniciado la transición de *azul* a *verde*.	Ninguno
notificación	RDS-EVENT-0248	Se ha completado el cambio en la implement ación azul/verde.	Ninguno
error	RDS-EVENT-0249	Se ha cancelado el cambio en la implementación azul/ verde.	Ninguno
notificación	RDS-EVENT-0259	La transición del clúster de bases de datos *azul* a *verde* ha comenzado.	Ninguno
notificación	RDS-EVENT-0260	La transición del clúster de bases de datos *azul* a *verde* ha finalizado. Se ha cambiado el nombre *azul* por *azul-antiguo* y *verde* por *azul*.	Ninguno
error	RDS-EVENT-0261	La transición del clúster de bases de datos *azul* a *verde* se ha cancelado por *motivo*.	Ninguno
notificación	RDS-EVENT-0311	La sincronización de la secuencia para la transició n del clúster de bases de datos *azul* a *verde* ha comenzado. La transición al usar secuencias puede provocar un tiempo de inactividad prolongado.	Ninguno

Categoría	ID de evento de Amazon RDS	Mensaje	Notas
notificación	RDS-EVENT-0312	La sincronización de la secuencia para la transición del clúster de bases de datos *azul* a *verde* ha finalizado.	Ninguno
error	RDS-EVENT-0314	La sincronización de la secuencia para la transición del clúster de bases de datos *azul* a *verde* se ha cancelado porque las secuencias no se han sincronizado.	Ninguno
notificación	RDS-EVENT-0409	*message*	Ninguno

Supervisión de archivos de registro de Amazon Aurora

Cada motor de base de datos de RDS genera registros a los que puede acceder para realizar auditorías y solucionar problemas. El tipo de registros depende del motor de base de datos.

Puede acceder a los registros de base de datos de instancias de base de datos mediante la Consola de administración de AWS, la AWS Command Line Interface (AWS CLI) o la API de Amazon RDS. No puede visualizar, ver ni descargar registros de transacciones.

> **Note**
>
> En algunos casos, los registros contienen datos ocultos. Por tanto, la Consola de administración de AWS podría mostrar contenido en un archivo de registro, pero el archivo de registro podría estar vacío cuando lo descarga.

Temas

- Visualización y descripción de archivos de registro de base de datos
- Descarga de un archivo de registro de base de datos
- Ver un archivo de registro de base de datos
- Publicación de registros de base de datos en registros de Amazon Cloudwatch
- Lectura del contenido del archivo de registro mediante REST
- Archivos de registro de base de datos de Aurora MySQL
- Archivos de registro de bases de datos de Aurora PostgreSQL

Visualización y descripción de archivos de registro de base de datos

Puede ver los archivos de registro de base de datos de su motor de base de datos de Amazon Aurora con la Consola de administración de AWS. Puede ver los archivos de registro que están disponibles para descargar o monitorear mediante la AWS CLI o la API de Amazon RDS.

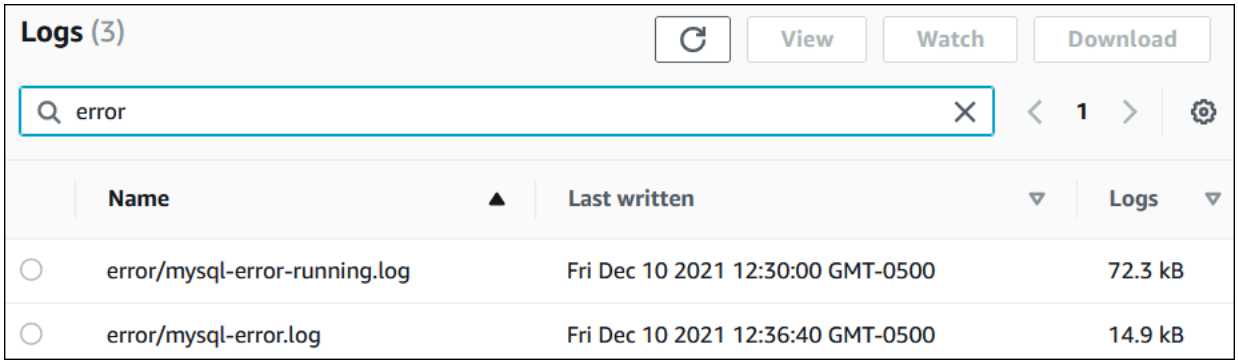
> **Note**
>
> No puede ver los archivos de registro de los clústeres de base de datos de Aurora Serverless v1 en la consola de RDS. Sin embargo, puede verlos en la consola de Amazon CloudWatch en https://console.aws.amazon.com/cloudwatch/.

Consola

Para ver un archivo de registro de base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el nombre de la instancia de base de datos que tiene el archivo de registro que desea visualizar.
4. Seleccione la pestaña Logs & events (Registros y eventos).
5. Desplácese hacia abajo hasta la sección Logs (Registros).
6. (Opcional) Ingrese un término de búsqueda para filtrar los resultados.

   En el siguiente ejemplo, se enumeran los registros filtrados por el texto **`error`**.



7. Elija el registro que quiera visualizar y, a continuación, elija View (Ver).

AWS CLI

Para ver los archivos de registro de base de datos disponibles para una instancia de base de datos, use el comando AWS CLI de la `describe-db-log-files`.

El siguiente ejemplo devuelve una lista de los archivos de registro de una instancia de base de datos denominada `my-db-instance`.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

API de RDS

Para ver los archivos de registro de base de datos de una instancia de base de datos, use la acción `DescribeDBLogFiles` de la API de Amazon RDS.

Descarga de un archivo de registro de base de datos

Puede usar la Consola de administración de AWS, la AWS CLI o la API para descargar un archivo de registro de base de datos.

Consola

Para descargar un archivo de registro de base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el nombre de la instancia de base de datos que tiene el archivo de registro que desea visualizar.
4. Seleccione la pestaña Logs & events (Registros y eventos).
5. Desplácese hacia abajo hasta la sección Logs (Registros).
6. En la sección Logs (Registros), elija el botón junto al registro que desee descargar y, a continuación, elija Download (Descargar).
7. Abra el menú contextual (haga clic con el botón derecho) del enlace que se proporciona y elija Save Link As (Guardar enlace como). Escriba la ubicación en la que desee guardar el archivo de registro y elija Save (Guardar).

AWS CLI

Para descargar un archivo de registro de base de datos, use el comando AWS CLI de la `download-db-log-file-portion`. De forma predeterminada, este comando solo descarga la última porción de un archivo de registro. Sin embargo, puede descargar un archivo entero especificando el parámetro `--starting-token 0`.

En el siguiente ejemplo se muestra cómo descargar todo el contenido de un archivo de registro llamado log/ERROR.4 y almacenarlo en un archivo local denominado errorlog.txt.

Example

Para Linux, macOS o:Unix

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

En:Windows

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

API de RDS

Para descargar un archivo de registro de base de datos, use la acción `DownloadDBLogFilePortion` de la API de Amazon RDS.

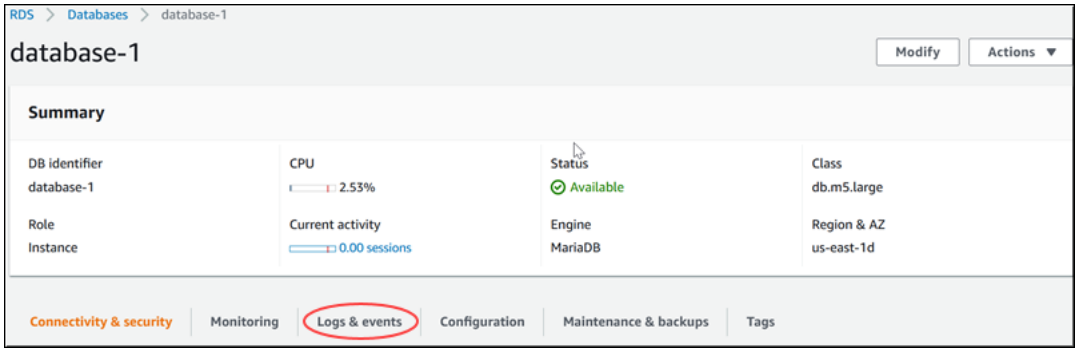
Ver un archivo de registro de base de datos

Ver un archivo de registro de base de datos equivale a detallar el archivo en un sistema UNIX o Linux. Puede ver un archivo de registro usando la Consola de administración de AWS. RDS actualiza el detalle del registro cada 5 segundos.

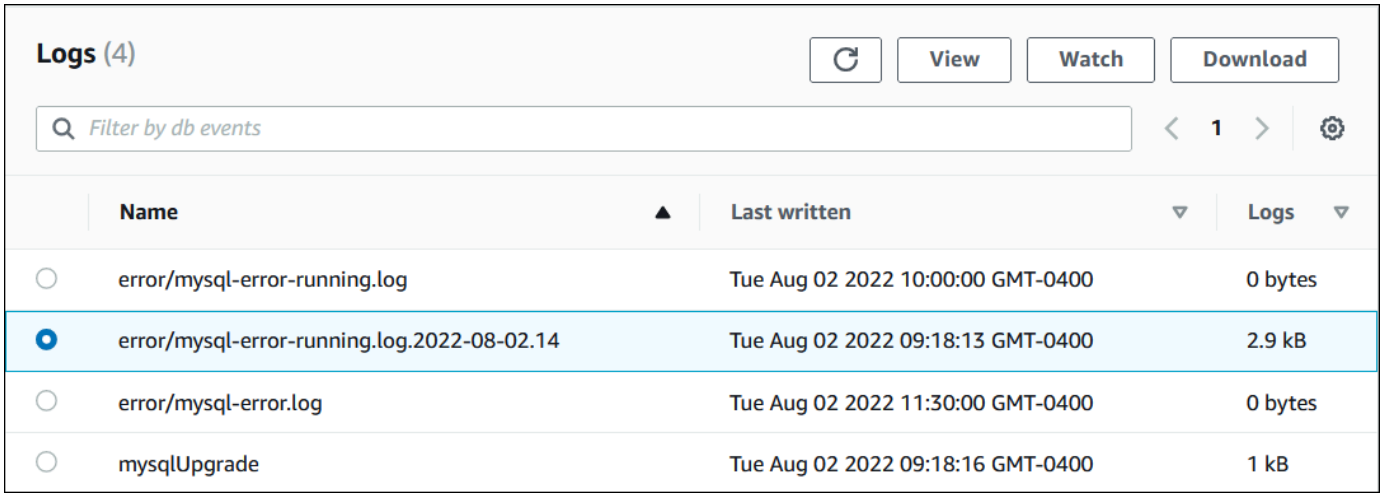
Para monitorizar un archivo de registro de base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

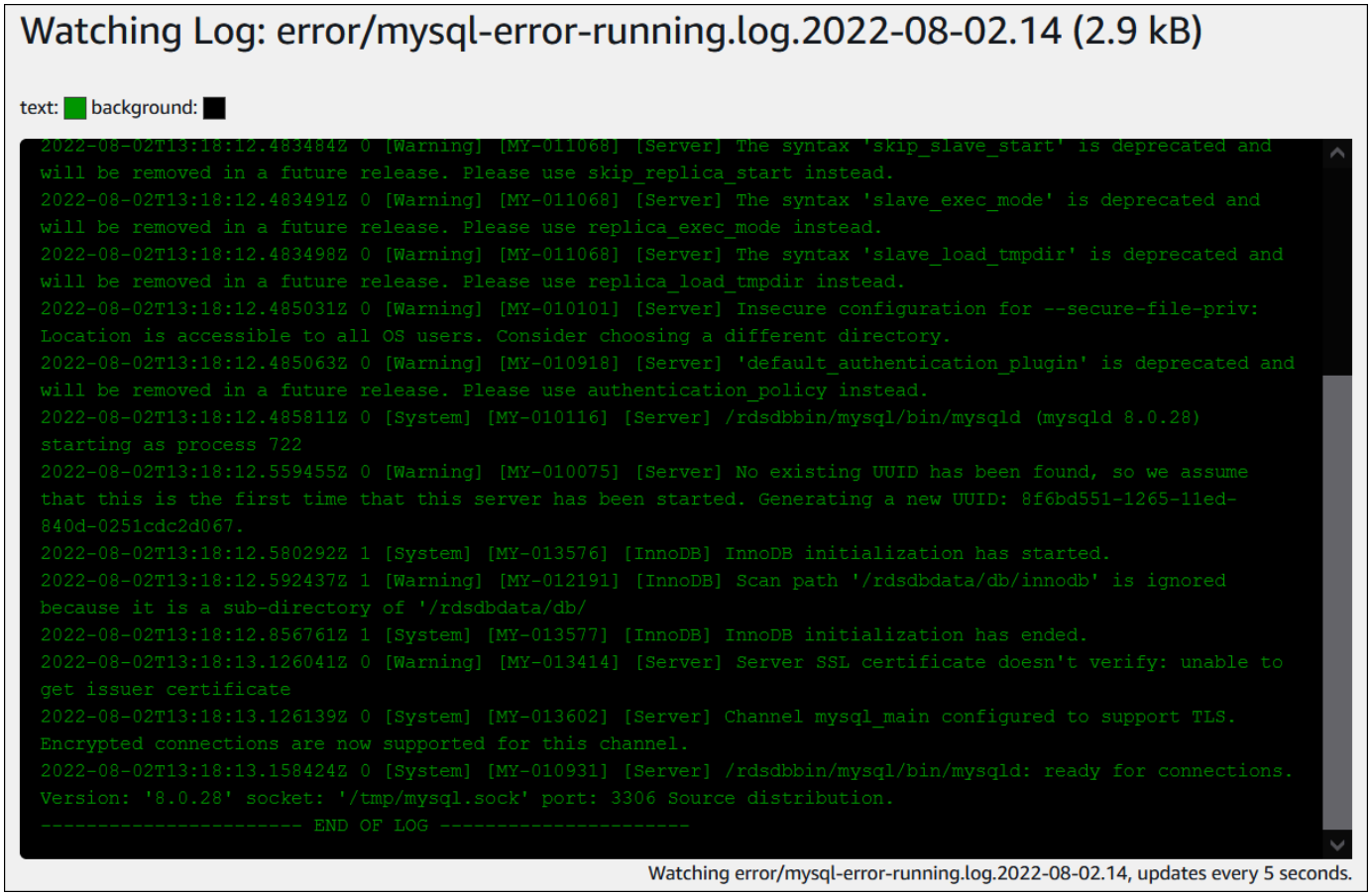
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione el nombre de la instancia de base de datos que tiene el archivo de registro que desea visualizar.
4. Seleccione la pestaña Logs & events (Registros y eventos).



5. En la sección Logs (Registros), elija un archivo de registro y, a continuación, elija Watch (Ver).



RDS muestra el detalle del registro, como en el siguiente ejemplo de MySQL.



Publicación de registros de base de datos en registros de Amazon Cloudwatch

En una base de datos en las instalaciones, los registros de la base de datos residen en el sistema de archivos. Amazon RDS no proporciona acceso de host a los registros de base de datos del sistema de archivos de el clúster de base de datos. Por este motivo, Amazon RDS le permite exportar registros de base de datos a registros de Amazon CloudWatch. Con CloudWatch Logs, puede realizar análisis en tiempo real de los datos de registro. También puede guardarlos en un almacenamiento de larga duración y gestionarlos con el agente de CloudWatch Logs.

Temas

- Descripción general de la integración de RDS con CloudWatch Logs
- Decisión sobre los registros que desea publicar en CloudWatch Logs
- Especificación de registros que desea publicar en CloudWatch Logs
- Búsqueda y filtrado de los registros en CloudWatch Logs

Descripción general de la integración de RDS con CloudWatch Logs

En CloudWatch Logs, un flujo de registro es una secuencia de eventos de registro que comparten el mismo origen. Cada fuente independiente de registros en Registros de CloudWatch constituye un flujo de registros independiente. Un grupo de registro es un grupo de flujos de registro que comparten la misma configuración de retención, monitorización y control de acceso.

Amazon Aurora transmite continuamente sus registros de clúster de base de datos en un grupo de registro. Por ejemplo, hay un grupo de registros `/aws/rds/cluster/`*`cluster_name`*`/`*`log_type`* para cada tipo de registro que se publica. Este grupo de registros se encuentra en la misma región de AWS que la instancia de base de datos que genera el registro.

AWS conserva los datos de registro publicados en CloudWatch Logs durante un periodo de tiempo indefinido a menos que se especifique un periodo de retención. Para obtener más información, consulte Cambiar la retención de datos de registro en CloudWatch Logs.

Decisión sobre los registros que desea publicar en CloudWatch Logs

Cada motor de base de datos RDS admite su propio conjunto de registros. Para obtener más información sobre las opciones del motor de base de datos, revise los siguientes temas:

- the section called “Publicación de registros de Aurora MySQL en CloudWatch Logs”
- the section called “Publicación de registros de Aurora PostgreSQL en CloudWatch Logs”

Especificación de registros que desea publicar en CloudWatch Logs

Puede especificar qué registros se van a publicar en la consola. Asegúrese de que tiene un rol vinculado a un servicio en AWS Identity and Access Management (IAM). Para obtener más información acerca de los roles vinculados a servicios, consulte Uso de roles vinculados a servicios de Amazon Aurora.

Para especificar los registros que se van a publicar

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Haga una de estas dos operaciones:
   - Elija Creación de base de datos.
   - Elija una base de datos de la lista y luego seleccione Modify (Modificar).

4. En Logs exports (Exportaciones de registros), elija los registros que desea publicar.

   En el siguiente ejemplo se especifican el registro de auditoría, los registros de errores, el registro general, el registro de instancias, el registro de errores de autenticación de bases de datos de IAM y el registro de consultas lentas para un clúster de bases de datos de Aurora MySQL.

Búsqueda y filtrado de los registros en CloudWatch Logs

Puede buscar las entradas de registro que cumplan los criterios especificados mediante la consola de CloudWatch Logs. Puede acceder a los registros a través de la consola de RDS, que lo lleva a la consola de CloudWatch Logs, o directamente desde la consola de CloudWatch Logs.

Para buscar los registros de RDS mediante la consola de RDS

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija un clúster de base de datos o una instancia de base de datos.
4. Elija Configuración.
5. En Published logs (Registros publicados), elija el registro de base de datos que desea ver.

Para buscar registros de RDS mediante la consola de CloudWatch Logs

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, seleccione Grupos de registro.
3. En el cuadro de filtro, escriba **`/aws/rds`**.
4. En Log Groups (Grupos de registro), elija el nombre del grupo de registro que contiene el flujo de registros que desea buscar.
5. En Log Streams (Flujos de registros), elija el nombre del flujo de registros que desea buscar.
6. En Log events (Eventos de registros), escriba la sintaxis del filtro que se va a utilizar.

Para obtener más información, consulte el temas sobre cómo buscar y filtrar datos de registros en la guía del usuario de Amazon CloudWatch. Para ver un tutorial en el blog que explique cómo monitorear los registros de RDS, consulte la publicación del blog sobre cómo crear un monitoreo proactivo de bases de datos para Amazon RDS con registros de Amazon Cloudwatch, AWS Lambda y Amazon SNS.

Lectura del contenido del archivo de registro mediante REST

Amazon RDS proporciona un punto de enlace REST que permite el acceso a los archivos de registro de instancia de base de datos. Esto resulta útil si necesita escribir una aplicación para transmitir el contenido del archivo de registro de Amazon RDS.

La sintaxis es la siguiente:

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

Se requieren los siguientes parámetros:

- *DBInstanceIdentifier*: el nombre asignado de la instancia de base de datos que contiene el archivo de registro que se desea descargar.
- *LogFileName*: el nombre del archivo de registro que se va a descargar.

La respuesta incluye el contenido del archivo de registro solicitado como una secuencia.

En el siguiente ejemplo se descarga el archivo de registro denominado log/ERROR.6 para la instancia de base de datos denominada sample-sql en la región us-west-2.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

Si especifica una instancia de base de datos no existente, la respuesta consta del error siguiente:

- `DBInstanceNotFound`: *DBInstanceIdentifier* no hace referencia a una instancia de base de datos existente. (Código de estado HTTP: 404)

Archivos de registro de base de datos de Aurora MySQL

Puede supervisar los registros de Aurora MySQL directamente desde la consola de Amazon RDS, la API de Amazon RDS, AWS CLI o los SDK de AWS. También es posible el acceso a los registros de MySQL dirigiéndolos a una tabla de la base de datos principal y consultando esa tabla. Puede usar la utilidad mysqlbinlog para descargar un registro binario.

Para obtener más información acerca de la visualización, descarga y vigilancia de los registros de bases de datos basados en archivos, consulte Supervisión de archivos de registro de Amazon Aurora.

Temas

- Información general de los registros de bases de datos de Aurora MySQL
- Envío de la salida del registro de Aurora MySQL a las tablas
- Configuración del registro binario de Aurora MySQL para bases de datos de Single-AZ
- Acceso a los registros binarios de MySQL

Información general de los registros de bases de datos de Aurora MySQL

Puede supervisar los siguientes tipos de archivos de registro de Aurora MySQL:

- Registro de errores
- Registro de consultas lentas
- Registro general
- Registro de auditoría
- Registro de instancias
- Registro de errores de autenticación de base de datos de IAM

El registro de errores de Aurora MySQL se genera de forma predeterminada. Puede generar la consulta lenta y los registros generales estableciendo parámetros en su grupo de parámetros de base de datos.

Temas

- Registros de errores de Aurora MySQL
- Registros generales y de consultas lentas de Aurora MySQL

- Registro de auditoría de Aurora MySQL
- Registro de instancias de Aurora MySQL
- Rotación y retención de registros en Aurora MySQL
- Publicación de registros de Aurora MySQL en Amazon CloudWatch Logs

Registros de errores de Aurora MySQL

Aurora MySQL escribe los errores en el archivo `mysql-error.log`. Cada archivo de registro tiene la hora a la que se generó (en UTC) agregada a su nombre. Los archivos de registro también tienen una marca temporal que ayuda a determinar cuándo se escribieron las entradas del registro.

Aurora MySQL solo escribe en el registro de errores durante el inicio, el cierre y cuando encuentra errores. Una instancia de base de datos puede pasar horas o días sin que se escriban nuevas entradas en el registro de errores. Si no hay entradas recientes, se debe a que el servidor no ha encontrado ningún error que genere una entrada en el registro.

Por diseño, los registros de errores se filtran para que solo se muestren los eventos inesperados, por ejemplo, los errores. No obstante, los registros de errores también contienen información adicional sobre la base de datos, por ejemplo, el progreso de la consulta, que no se muestra. Por lo tanto, incluso sin ningún error real, el tamaño de los registros de errores podría aumentar debido a las actividades continuas de la base de datos. Y, aunque puede que vea un tamaño determinado en bytes o kilobytes en los registros de error de la Consola de administración de AWS, es posible que tengan 0 bytes al descargarlos.

Aurora MySQL escribe `mysql-error.log` en disco cada 5 minutos. Adjunta el contenido del registro a `mysql-error-running.log`.

Aurora MySQL rota el archivo `mysql-error-running.log` cada hora.

> **Note**
>
> Tenga en cuenta que el periodo de retención es diferente entre Amazon RDS y Aurora.

Registros generales y de consultas lentas de Aurora MySQL

Puede escribir el registro de consultas lentas de Aurora MySQLRDS para MySQL y el registro general en un archivo o en una tabla de la base de datos. Para hacerlo, establezca los parámetros en el grupo de parámetros de la base de datos. Para obtener información acerca de cómo crear y

modificar un grupo de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora. Debe definir estos parámetros para poder ver el registro de consultas lentas o el registro general en la consola de Amazon RDS o a través de la API de Amazon RDS, la CLI de Amazon RDS o los SDK de AWS.

Puede controlar lo que registra Aurora MySQL con los parámetros de esta lista:

- `slow_query_log`: para crear el registro de consultas lentas, use el valor 1. El valor predeterminado es 0.
- `general_log`: para crear el registro general, use el valor 1. El valor predeterminado es 0.
- `long_query_time`: para evitar que se registren consultas rápidas en el registro de consultas lentas, especifique el valor del tiempo de ejecución mínimo de una consulta, en segundos, para que se registre. El valor predeterminado es 10 segundos y el mínimo es 0. Si log_output = FILE, puede especificar un valor de punto flotante que llega a una resolución de microsegundos. Si log_output = TABLE, debe especificar un valor entero con resolución de segundos. Solo se registran las consultas cuyo tiempo de ejecución exceda el valor de `long_query_time`. Por ejemplo, si configura `long_query_time` como 0,1, evitará que se registren las consultas que tarden menos de 100 milisegundos en ejecutarse.
- `log_queries_not_using_indexes`: para incluir en el registro de consultas lentas todas las consultas que no usen un índice, use el valor 1. Las consultas que no usen un índice se registran incluso si su tiempo de ejecución es inferior al valor del parámetro `long_query_time`. El valor predeterminado es 0.
- `log_output` *`option`*: puede especificar una de las opciones siguientes para el parámetro `log_output`.
  - TABLE : las consultas generales se escriben en la tabla `mysql.general_log` y las consultas lentas en la tabla `mysql.slow_log`.
  - FILE: tanto los registros de las consultas generales como los de las consultas lentas se escriben en el sistema de archivos.
  - NONE: deshabilitar registro.

  Para las versiones 2 y 3 de Aurora MySQL, el valor predeterminado de `log_output` es `FILE`.

Para que los datos de consultas lentas aparezcan en Registros de Amazon CloudWatch, se deben cumplir las siguientes condiciones:

- Registros de CloudWatch debe configurarse para incluir registros de consultas lentas.

- `slow_query_log` debe estar habilitado.
- `log_output` se debe establecer en `FILE`.
- La consulta debe tardar más tiempo que el tiempo configurado para `long_query_time`.

Para obtener más información sobre la el registro de consultas lentas y el registro general, consulte los siguientes temas de la documentación de MySQL:

- El registro de consultas lentas
- El registro de consultas generales

Registro de auditoría de Aurora MySQL

El registro de auditoría para Aurora MySQL se llama auditoría avanzada. Para activar la auditoría avanzada, hay que establecer ciertos parámetros del clúster de base de datos. Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.

Registro de instancias de Aurora MySQL

Aurora crea un archivo de registro independiente para las instancias de base de datos que tienen la pausa automática habilitada. Este archivo instance.log registra cualquier motivo por el que estas instancias de base de datos no han podido pausarse cuando se esperaba. Para obtener más información sobre el comportamiento del archivo de registro de instancias y la capacidad de pausa automática de Aurora, consulte Supervisión de la actividad de pausa y reanudación de Aurora sin servidor v2.

Rotación y retención de registros en Aurora MySQL

Cuando el registro está habilitado, Amazon Aurora rota o elimina los archivos de registro a intervalos regulares. Esta medida es una precaución para reducir el riesgo de que un archivo de registro grande bloquee el uso de la base de datos o afecte al desempeño. Aurora MySQL gestiona la rotación y la eliminación de la siguiente manera:

- El tamaño de los archivos de registro de errores de Aurora MySQL está limitado al 15 % del almacenamiento local de una instancia de base de datos. Para mantener este umbral, los registros se rotan automáticamente cada hora. Aurora MySQL elimina los registros después de 30 días o cuando se alcanza el 15 % del espacio en disco. Si el tamaño combinado de los archivos de registro sobrepasa el umbral después de eliminar los archivos de registro antiguos, los archivos de

registro más grandes se eliminan hasta que el tamaño del archivo de registro deje de sobrepasar el umbral.

- Aurora MySQL elimina los registros de auditoría, generales y de consultas lentas después de 24 horas o cuando se ha consumido el 15 % del almacenamiento.
- Cuando el registro `FILE` está activado, los archivos de registro general y de consulta lenta se examinan cada hora y se eliminan los archivos de registro con más de 24 horas de antigüedad. En algunos casos, el tamaño restante del archivo de registro combinado después de la eliminación puede superar el umbral del 15 % del espacio local de una instancia de base de datos. En estos casos, los archivos de registro más antiguos se eliminan hasta que el tamaño del archivo de registro no sobrepase el umbral.
- Cuando el registro `TABLE` está activado, las tablas de registro no se rotan ni se eliminan. Las tablas de registro se truncan cuando el tamaño de todos los registros combinados es demasiado grande. Puede suscribirse a la categoría de evento `low storage` para recibir una notificación cuando las tablas de registro se deban rotar o eliminar manualmente para liberar espacio. Para obtener más información, consulte Uso de notificaciones de eventos de Amazon RDS.

  Puede rotar la tabla `mysql.general_log` manualmente si llama al procedimiento `mysql.rds_rotate_general_log`. Para rotar la tabla `mysql.slow_log` puede ejecutar el procedimiento `mysql.rds_rotate_slow_log`.

  Cuando rota las tablas de registro manualmente, la tabla de registro actual se copia en una tabla de registro de copia de seguridad y se eliminan las entradas de la tabla de registro actual. Si la tabla de registro de copia de seguridad ya existe, se elimina antes de copiar la tabla del registro actual en la copia de seguridad. Puede consultar la tabla de registro de copia de seguridad si es necesaria. La tabla de registro de copia de seguridad para la tabla `mysql.general_log` se llama `mysql.general_log_backup`. La tabla de registro de copia de seguridad de la tabla `mysql.slow_log` se llama `mysql.slow_log_backup`.
- Los registros de auditoría de Aurora MySQL se rotan cuando el tamaño de archivo alcanza los 100 MB y se eliminan después de 24 horas.
- Amazon RDS rota los archivos de registro de errores de autenticación de la base de datos de IAM de más de 10 MB. Amazon RDS elimina los archivos de registro de errores de autenticación de la base de datos de IAM que tengan más de cinco días o más de 100 MB.

Para trabajar con los registros desde la consola de Amazon RDS, la API de Amazon RDS, la CLI de Amazon RDS o los SDK de AWS, configure el parámetro `log_output` en FILE. Al igual que el registro de errores de Aurora MySQL, estos archivos de registro rotan cada hora. Se conservan

los archivos de registro que se generaron durante las 24 horas anteriores. Tenga en cuenta que el período de retención es diferente entre Amazon RDS y Aurora.

Publicación de registros de Aurora MySQL en Amazon CloudWatch Logs

Puede configurar un clúster de bases de datos de Aurora MySQL para publicar datos de registro en un grupo de registros en Amazon CloudWatch Logs. Con CloudWatch Logs, puede realizar análisis en tiempo real de los datos de registro y utilizar CloudWatch para crear alarmas y ver métricas. Puede utilizar CloudWatch Logs para almacenar los registros de registros en almacenamiento de larga duración. Para obtener más información, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.

Envío de la salida del registro de Aurora MySQL a las tablas

Para dirigir los registros general y de consultas lentas a tablas de la instancia de base de datos, cree un grupo de parámetros de base de datos y asigne al parámetro `log_output` del servidor el valor `TABLE`. Las consultas generales se registrarán entonces en la tabla `mysql.general_log` y las consultas lentas en la tabla `mysql.slow_log`. Puede consultar las tablas para obtener acceso a la información del registro. Al habilitar este registro, se incrementa la cantidad de datos que se escribe en la base de datos, lo que puede degradar el desempeño.

Tanto el registro general como los registros de consultas lentas están deshabilitados de manera predeterminada. Para habilitar el registro en tablas, también debe asignar a los parámetros `general_log` y `slow_query_log` del servidor el valor `1`.

Las tablas de registro seguirán creciendo hasta que las actividades de registro respectivas se desactiven cambiando el valor del parámetro a `0`. A menudo, se acumula una gran cantidad de datos, lo que puede consumir un porcentaje elevado del espacio de almacenamiento asignado. Amazon Aurora no permite truncar las tablas de registro, pero sí mover su contenido. Al rotar una tabla, su contenido se guarda en una tabla de copia de seguridad y se crea una nueva tabla de registro vacía. Puede aplicar manualmente la rotación de las tablas de registro con los procedimientos de línea de comandos siguientes, en los que el símbolo del sistema se representa por:`PROMPT>`

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

Para eliminar por completo los datos antiguos y recuperar el espacio del disco, llame al procedimiento correspondiente dos veces consecutivas.

Configuración del registro binario de Aurora MySQL para bases de datos de Single-AZ

El registro binario es un conjunto de archivos de registro que contienen información acerca de las modificaciones de datos hechas en una instancia de servidor de Aurora MySQL. El registro binario contiene información como la siguiente:

- Eventos que describen cambios en la base de datos, como la creación de tablas o las modificaciones de filas.
- Información sobre la duración de cada instrucción que actualizó los datos.
- Eventos para instrucciones que podrían haber actualizado datos, pero que no lo hicieron.

El registro binario registra las instrucciones que se envían durante la replicación. También es necesario para algunas operaciones de recuperación. Para obtener más información, consulte The Binary Log en la documentación de MySQL.

Los registros binarios solo son accesibles desde la instancia principal de base de datos, no desde las réplicas.

MySQL en Amazon Aurora admite los formatos de registro binario basado en filas, basado en instrucciones y mixto. Recomendamos mezclarlos, a menos que necesite un formato binlog concreto. Para obtener información detallada acerca de los formatos de registro binarios de MySQL de Aurora, consulte Binary logging formats en la documentación de MySQL.

Si tiene pensado utilizar la replicación, el formato de registro binario es importante porque determina el registro de los cambios de datos que se registra en la fuente y se envía a los objetivos de replicación. Para obtener más información acerca de las ventajas y desventajas de distintos tipos de formatos de registro binarios para la replicación, consulte Advantages and Disadvantages of Statement-Based and Row-Based Replication en la documentación de MySQL.

> **Important**
>
> Con MySQL 8.0.34, MySQL ha dejado de utilizar el parámetro `binlog_format`. En versiones posteriores de MySQL, MySQL planea eliminar el parámetro y admitir únicamente la replicación basada en filas. Por ello, recomendamos utilizar el registro basado en filas para las nuevas configuraciones de replicación de MySQL. Para obtener más información, consulte binlog_format en la documentación de MySQL.

Las versiones 8.0 y 8.4 de MySQL aceptan el parámetro `binlog_format`. Al usar este parámetro, MySQL emite una advertencia de obsolescencia. En una futura versión principal, MySQL eliminará el parámetro `binlog_format`.
La replicación basada en instrucciones puede causar incoherencias entre el clúster de la de base de datos de origen y la réplica de lectura. Para obtener más información, consulte Determination of Safe and Unsafe Statements in Binary Logging en la documentación de MySQL.
Habilitar el registro binario aumenta el número de operaciones de E/S de escritura en el disco en el clúster de base de datos. Puede supervisar el uso de IOPS con la métrica de CloudWatch `VolumeWriteIOPs`.

Para configurar el formato de registro binario de MySQL

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Grupos de parámetros.
3. Seleccione el grupo de parámetros del clúster de base de datos asociado con el clúster de base de datos que quiera modificar.

   No puede modificar un grupo de parámetros predeterminado. Si el clúster de la de base de datos emplea un grupo de parámetros predeterminado, cree un nuevo grupo de parámetros y asócielo con el clúster de la de base de datos.

   Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.
4. En Acciones, elija Editar.
5. Establezca el parámetro `binlog_format` en el formato de registro binario de su elección (`ROW`, `STATEMENT` o `MIXED`). También puede utilizar el valor `OFF` para desactivar el registro binario.

   > **Note**
   >
   > Si `binlog_format` se establece en `OFF` en el grupo de parámetros del clúster de base de datos, se deshabilita la variable de sesión `log_bin`. Esto deshabilita el registro binario en el clúster de base de datos de Aurora MySQL, lo que a su vez restablece la variable de sesión `binlog_format` al valor predeterminado de `ROW` en la base de datos.

6. Elija Guardar cambios para guardar los cambios realizados en el grupo de parámetros del clúster de la base de datos.

Tras realizar estos pasos, debe reiniciar la instancia de escritor en el clúster de base de datos para que se apliquen los cambios. En la versión 2.09 de Aurora MySQL y las versiones anteriores, al reiniciar la instancia del escritor, también se reinician todas las instancias del lector en el clúster de base de datos. En la versión 2.10 de Aurora MySQL y versiones posteriores, debe reiniciar todas las instancias del lector manualmente. Para obtener más información, consulte Reinicio de un clúster de base de datos de Amazon Aurora o de una instancia de base de datos de Amazon Aurora.

> **Important**
>
> El cambio de un grupo de parámetros de clúster de bases de datos afecta a todos los clústeres de base de datos que utilizan ese grupo de parámetros. Si desea especificar diferentes formatos de registro binario para diferentes clústeres de base de datos de Aurora MySQL en una región de AWS, los clústeres de base de datos deben utilizar diferentes grupos de parámetros de clúster de bases de datos. Estos grupos de parámetros identifican diferentes formatos de registro. Asigne el grupo de parámetros de clúster de bases de datos apropiado a cada clúster de bases de datos. Para obtener más información acerca de los parámetros Aurora MySQL, consulte Parámetros de configuración de Aurora MySQL.

Acceso a los registros binarios de MySQL

Puede usar la herramienta mysqlbinlog para descargar o transmitir los registros binarios desde las instancias de Amazon RDS para MySQL. El registro binario se descarga en el equipo local, donde puede ejecutar acciones tales como reproducirlo con la utilidad mysql. Para obtener más información acerca del uso de la herramienta mysqlbinlog, consulte Using mysqlbinlog to Back Up Binary Log Files (Uso de mysqlbinlog para realizar copias de seguridad de archivos de registro binarios) en la documentación de MySQL.

Para ejecutar la utilidad mysqlbinlog en una instancia de Amazon RDS, use las siguientes opciones:

- `--read-from-remote-server`: obligatorio.
- `--host`: el nombre de DNS del punto de conexión de la instancia.
- `--port`: el puerto que utiliza la instancia.
- `--user`: un usuario de MySQL al que se le concede el permiso `REPLICATION SLAVE`.

- `--password`: la contraseña del usuario de MySQL, o bien no indique ninguna para que la herramienta le pida una.
- `--raw`: descargue el archivo en formato binario.
- `--result-file`: el archivo local en que se guardará la salida sin procesar.
- `--stop-never`: transmita los archivos de registro binarios.
- `--verbose`: cuando utilice el formato binlog de `ROW`, incluya esta opción para ver los eventos de fila como instrucciones pseudo-SQL. Para obtener más información acerca de la opción `--verbose`, consulte mysqlbinlog row event display (Visualización de eventos de fila mysqlbinlog) en la documentación de MySQL.
- Especifique los nombres de uno o varios de los archivos de registro binarios. Para obtener una lista de los registros disponibles, use el comando de SQL `SHOW BINARY LOGS`.

Para obtener más información acerca de las opciones de mysqlbinlog, consulte mysqlbinlog - Utility for Processing Binary Log Files (mysqlbinlog - Utilidad para procesar archivos de registro binarios) en la documentación de MySQL.

En los siguientes ejemplos, se muestra cómo utilizar la herramienta mysqlbinlog.

Para Linux, macOS o Unix:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

Para Windows:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
```

```
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

Los registros binarios deben permanecer disponibles en la instancia de base de datos para que la utilidad mysqlbinlog pueda acceder a ellos. Para garantizar su disponibilidad, utilice el procedimiento mysql.rds_set_configuration almacenado y especifique un periodo con tiempo suficiente para descargar los registros. Si esta configuración no está establecida, Amazon RDS purga los registros binarios lo antes posible, lo que genera huecos en los registros binarios que recupera la utilidad mysqlbinlog.

En el siguiente ejemplo se define el periodo de retención en 1 día.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

Para mostrar el valor actual, utilice el procedimiento almacenado mysql.rds_show_configuration.

```
call mysql.rds_show_configuration;
```

Archivos de registro de bases de datos de Aurora PostgreSQL

Puede supervisar los siguientes tipos de archivos de registro de Aurora PostgreSQL:

- Registro de PostgreSQL
- Registro de instancias
- Registro de errores de autenticación de base de datos de IAM

> **Note**
>
> Para habilitar los registros de errores de autenticación de base de datos de IAM, primero debe habilitar la autenticación de base de datos de IAM para el clúster de base de datos de Aurora PostgreSQL. Para obtener más información sobre la habilitación de la autenticación de bases de datos de IAM, consulte Activación y desactivación de la autenticación de bases de datos de IAM.

Aurora PostgreSQL registra las actividades de la base de datos en el archivo de registro de PostgreSQL predeterminado. En el caso de una instancia de base de datos de PostgreSQL en las instalaciones, estos mensajes se almacenan localmente en `log/postgresql.log`. Para un clúster de base de datos de Aurora PostgreSQL, el archivo de registro está disponible en el clúster de Aurora, . También puede acceder a estos registros a través de la Consola de administración de AWS, donde podrá verlos o descargarlos. El nivel de registro predeterminado captura los errores de inicio de sesión, los errores graves del servidor, los bloqueos y los errores de consulta.

Para obtener más información sobre cómo puede ver, descargar y observar los registros de la base de datos basados en archivos, consulte Supervisión de archivos de registro de Amazon Aurora. Para saber más sobre los registros de PostgreSQL, consulte Trabajo con registros de Amazon RDS y Aurora PostgreSQL: parte 1 y Trabajo con registros de Amazon RDS y Aurora PostgreSQL: parte 2.

Además de los registros estándar de PostgreSQL que se describen en este tema, Aurora PostgreSQL también admite la extensión de auditoría de PostgreSQL (`pgAudit`). La mayoría de los sectores regulados y las agencias gubernamentales necesitan mantener un registro de auditoría o registro de auditoría de los cambios realizados en los datos para cumplir con los requisitos legales. Para obtener información acerca del modo de instalar y usar pgAudit, consulte Uso de pgAudit para registrar la actividad de la base de datos.

Aurora crea un archivo de registro independiente para las instancias de base de datos que tienen la pausa automática habilitada. Este archivo instance.log registra cualquier motivo por el que estas instancias de base de datos no han podido pausarse cuando se esperaba. Para obtener más información sobre el comportamiento del archivo de registro de instancias y la capacidad de pausa automática de Aurora, consulte Supervisión de la actividad de pausa y reanudación de Aurora sin servidor v2.

Temas

- Parámetros de registro en Aurora PostgreSQL
- Activación de registro de consultas para su clúster de base de datos de Aurora PostgreSQL

Parámetros de registro en Aurora PostgreSQL

Puede personalizar el comportamiento de registro de su clúster de base de datos de Aurora PostgreSQL modificando varios parámetros. En la siguiente tabla, puede encontrar los parámetros que afectan al tiempo que se almacenan los registros, cuándo se debe rotar el registro y si se debe generar el registro en formato CSV (valor separado por comas). También puede encontrar la salida de texto enviada a STDERR, entre otras configuraciones. Para cambiar la configuración de los parámetros que se pueden modificar, use un grupo de parámetros de clúster de base de datos personalizado para su clúster de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.

Parámetro	Predeterminado/a	Descripción
log_destination	stderr	Establece el formato de salida para el registro. El valor predeterminado es `stderr`, pero también puede especificar un valor separado por comas (CSV) agregándolo `csvlog` al ajuste. Para obtener más información, consulte Configuración del destino del registro (`stderr`, `csvlog`).
log_filename	postgresql.log.%Y-%m-%d-%H%M	Especifica el patrón del nombre del archivo de registro. Además del valor predeterminado, este parámetro admite `postgresql.log.%Y-%m-%d` y `postgresql.log.%Y-%m-%d-%H` para el patrón del nombre de archivo.

Parámetro	Predeterminado/a	Descripción
		Para Aurora PostgreSQL versión 17.4 y posteriores, no puede modificar este parámetro .
log_line_prefix	%t:%r:%u@%d:[%p]:	Define el prefijo de cada línea de registro que se escribe en `stderr`, para anotar la hora (%t), el host remoto (%r), el usuario (%u), la base de datos (%d) y el ID del proceso (%p).
log_rotation_age	60	Minutos después de los cuales el archivo de registro rota automáticamente. Puede cambiar este valor dentro del intervalo de 1 a 1440 minutos. Para obtener más informaci ón, consulte Configuración de la rotación del archivo de registro.
log_rotation_size	–	El tamaño (kB) con el que se rota automátic amente el registro. Puede cambiar este valor dentro del intervalo de 50 000 a 1 000 000 kilobytes. Para obtener más información, consulte Configuración de la rotación del archivo de registro.
rds.log_retention_ period	4320	Los registros de PostgreSQL que superan el número de minutos especificado se eliminará n. El valor predeterminado de 4320 minutos elimina los archivos de registro transcurridos 3 días. Para obtener más información, consulte Configuración de periodo de retención de registros.

Para identificar problemas de aplicaciones, puede buscar errores de consulta, errores de inicio de sesión, interbloqueos y errores de servidor graves en el registro. Por ejemplo, suponga que ha convertido una aplicación heredada de Oracle a Aurora PostgreSQL, pero no todas las consultas se han convertido correctamente. Estas consultas con formato incorrecto generan mensajes de

error que se pueden encontrar en los registros para ayudar a identificar los problemas. Para más información sobre el registro de consultas, consulte Activación de registro de consultas para su clúster de base de datos de Aurora PostgreSQL.

En los temas siguientes, encontrará información sobre cómo configurar varios parámetros que controlan los detalles básicos de sus registros de PostgreSQL.

Temas

- Configuración de periodo de retención de registros
- Configuración de la rotación del archivo de registro
- Configuración del destino del registro (stderr, csvlog)
- Descripción del parámetro log_line_prefix

Configuración de periodo de retención de registros

El parámetro `rds.log_retention_period` especifica cuánto tiempo su clúster de base de datos de Aurora PostgreSQL mantiene sus archivos de registro. La configuración predeterminada es de 3 días (4320 minutos), pero puede configurarla entre 1 día (1440 minutos) y 7 días (10 080 minutos). Asegúrese de que su clúster de base de datos de Aurora PostgreSQL tenga suficiente almacenamiento para almacenar los archivos de registro durante ese período de tiempo.

Le recomendamos que publique sus registros de forma rutinaria en Registros de Amazon CloudWatch, de modo que pueda ver y analizar los datos del sistema mucho tiempo después de que los registros se hayan eliminado de su clúster de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Publicación de registros de Aurora PostgreSQL en Amazon CloudWatch Logs. Después de configurar la publicación en CloudWatch, Aurora no elimina un registro hasta que se publique en CloudWatch Logs.

Amazon Aurora comprime los registros de PostgreSQL más antiguos cuando el almacenamiento de la instancia de base de datos alcanza un umbral. Aurora comprime los archivos mediante la utilidad de compresión gzip. Para obtener más información, consulte el sitio web de gzip.

Cuando el almacenamiento para la instancia de base de datos es bajo y se comprimen todos los registros disponibles, se obtiene una advertencia como la siguiente:

```
Warning: local storage for PostgreSQL log files is critically low for
this Aurora PostgreSQL instance, and could lead to a database outage.
```

Si no hay suficiente almacenamiento, Aurora podría eliminar los registros comprimidos de PostgreSQL antes de que finalice el período de retención especificado. Si eso sucede, verá un mensaje parecido al siguiente:

```
The oldest PostgreSQL log files were deleted due to local storage constraints.
```

Configuración de la rotación del archivo de registro

Aurora crea los nuevos archivos de registro cada hora de forma predeterminada. El tiempo lo controla el parámetro `log_rotation_age`. Este parámetro tiene un valor predeterminado de 60 (minutos), pero puede configurarlo entre 1 minuto y 24 horas (1440 minutos). Cuando llega el momento de la rotación, se crea un nuevo archivo de registro distinto. Al archivo se le asigna un nombre de conformidad con el patrón especificado por el parámetro `log_filename`.

Los archivos de registro también se pueden rotar según su tamaño, tal y como se especifica en el parámetro `log_rotation_size`. Este parámetro especifica que el registro debe rotarse cuando alcance el tamaño especificado (en kilobytes). El `log_rotation_size` predeterminado es de 100 000 kB (kilobytes) para un clúster de base de datos de Aurora PostgreSQL, pero puede establecer en cualquier valor desde 50 000 hasta 1 000 000 de kilobytes.

Los nombres de archivo de registro se basan en el patrón de nombre de archivo especificado en el parámetro `log_filename`. La configuración disponible para este parámetro es la siguiente:

- `postgresql.log.%Y-%m-%d`: formato predeterminado para el nombre del archivo de registro. Incluye el año, el mes y la fecha en el nombre del archivo de registro.
- `postgresql.log.%Y-%m-%d-%H`: incluye la hora en el formato del nombre del archivo de registro.
- `postgresql.log.%Y-%m-%d-%H%M`: incluye la hora:minuto en el formato del nombre del archivo de registro.

Si establece el parámetro `log_rotation_age` en menos de 60 minutos, establezca el parámetro `log_filename` en el formato de minutos.

Para obtener más información, consulte `log_rotation_age` y `log_rotation_size` en la documentación de PostgreSQL.

Configuración del destino del registro (**`stderr`**, **`csvlog`**)

De forma predeterminada, Aurora PostgreSQL genera registros en formato de error estándar (stderr). Esta es la configuración predeterminada del parámetro `log_destination`. Cada mensaje lleva el prefijo según el patrón especificado en el parámetro `log_line_prefix`. Para obtener más información, consulte Descripción del parámetro log_line_prefix.

Aurora PostgreSQL también puede generar los registros con el formato `csvlog`. `csvlog` resulta útil para analizar los datos de registro como valores separados con coma (CSV). Por ejemplo, supongamos que utiliza la extensión `log_fdw` para trabajar con los registros como tablas extranjeras. La tabla extranjera creada en los archivos de registro `stderr` contienen una sola columna con datos de eventos de registro. Al añadir `csvlog` al parámetro `log_destination`, se obtiene el archivo de registro en formato CSV con demarcaciones para las múltiples columnas de la tabla externa. Esto le permite ordenar y analizar los registros con mayor facilidad.

Si especifica `csvlog` para este parámetro, tenga en cuenta que se generan los archivos `stderr` y `csvlog`. Asegúrese de supervisar el almacenamiento consumido por los registros, teniendo en cuenta `rds.log_retention_period` y otras configuraciones que afectan al almacenamiento de registros y a los análisis. Usar `stderr` y `csvlog` duplica de sobra el almacenamiento consumido por los registros.

Si añade `csvlog` a `log_destination` y quiere volver solo a `stderr` solo, debe restablecer el parámetro. Para ello, utilice la consola de Amazon RDS y abra el grupo de parámetros del clúster de base de datos personalizado para su instancia. Elija el parámetro `log_destination`, elija Edit parameter (Editar parámetro) y, a continuación, seleccione Reset (Restablecer).

Para obtener más información acerca de la configuración de los registros, consulte la entrada del blog sobre trabajar con registros de Amazon RDS y Aurora PostgreSQL: parte 1.

Descripción del parámetro log_line_prefix

El formato de registro de `stderr` prefija cada mensaje de registro con los detalles especificados por el parámetro `log_line_prefix`. El valor predeterminado es:

```
%t:%r:%u@%d:[%p]:t
```

A partir de la versión 16 de Aurora PostgreSQL, también puede elegir:

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

Cada entrada de registro enviada a stderr incluye la siguiente información en función del valor seleccionado:

- `%t`: hora de entrada del registro sin milisegundos
- `%m`: hora de entrada del registro con milisegundos
- `%r`: dirección del host remoto.
- `%u@%d`: nombre de usuario @ nombre de base de datos.
- `[%p]`: ID del proceso si está disponible.
- `%l`: Número de línea de registro por sesión
- `%e`: Código de error SQL
- `%s`: Marca temporal de inicio de proceso
- `%v`: ID de transacción virtual
- `%x`: ID de transacción
- `%c`: ID de sesión
- `%q`: Terminador que no sea de sesión
- `%a`: Nombre de la aplicación

Activación de registro de consultas para su clúster de base de datos de Aurora PostgreSQL

Puede recopilar información más detallada sobre las actividades de la base de datos, incluidas las consultas, las consultas en espera de bloqueos, los puntos de control y muchos otros detalles configurando algunos de los parámetros que se enumeran en la tabla siguiente. Este tema se centra en el registro de consultas.

Parámetro	Predeterminado/a	Descripción
log_connections	–	Registra cada conexión realizada correctamente. Para obtener información sobre cómo utilizar este parámetro con `log_disconnections` para detectar la pérdida de conexiones, consulte Administración de la pérdida de conexión de Aurora PostgreSQL con agrupación .

Parámetro	Predeterminado/a	Descripción
log_disconnections	–	Registra el final de cada sesión y su duración. Para obtener información sobre cómo utilizar este parámetro con `log_connections` para detectar la pérdida de conexiones, consulte Administración de la pérdida de conexión de Aurora PostgreSQL con agrupación .
log_checkpoints	:	No se aplica a Aurora PostgreSQL
log_lock_waits	–	Registra las esperas de bloqueo largas. Por defecto, este parámetro no está configurado.
log_min_duration_sample	–	(ms) Establece el tiempo mínimo de ejecución a partir del cual se registra una muestra de instrucciones. El tamaño de la muestra se establece mediante el parámetro `log_statement_sample_rate` .
log_min_duration_statement	–	Se registra cualquier instrucción SQL que se ejecute al menos durante el período de tiempo especificado o durante más tiempo. Por defecto, este parámetro no está configurado. Si se activa este parámetro, puede resultar más sencillo encontrar consultas no optimizadas.

Parámetro	Predeterminado/a	Descripción
log_statement	–	Define el tipo de declaraciones que se deben registrar. De forma predeterminada, este parámetro no está configurado, pero puede cambiarlo a `all`, `ddl` o `mod` para especific ar los tipos de instrucciones SQL que desea que se registren. Si especifica algo que no sea `none` para este parámetro, también debe adoptar medidas adicionales para evitar que las contraseñas aparezcan en los archivos de registro. Para obtener más información, consulte Mitigar el riesgo de exposición de contraseñas al utilizar el registro de consultas.
log_statement_samp le_rate	–	El porcentaje de sentencias que superan el tiempo especificado en `log_min_d uration_sample` se registrarán, expresado como un valor de coma flotante entre 0.0 y 1.0.
log_statement_stats	–	Escribe las estadísticas de rendimiento acumulativas en el registro del servidor.

Uso del registro para encontrar consultas con un rendimiento lento

Puede registrar instrucciones y consultas SQL para ayudar a encontrar consultas que se den con resultados lentos. Para activar esta función, modifique la configuración de los parámetros `log_statement` y `log_min_duration`, tal como se describe en esta sección. Antes de activar el registro de consultas para su clúster de base de datos de Aurora PostgreSQL, debe conocer la posible exposición de contraseñas en los registros y cómo mitigar los riesgos. Para obtener más información, consulte Mitigar el riesgo de exposición de contraseñas al utilizar el registro de consultas.

A continuación, encontrará información de referencia sobre los parámetros `log_statement` y `log_min_duration`.

log_statement

Este parámetro especifica el tipo de instrucciones SQL que deben enviarse al registro. El valor predeterminado es `none`. Si cambia este parámetro a `all`, `ddl` o `mod`, asegúrese de aplicar algunas de las medidas recomendadas para reducir el riesgo de exponer las contraseñas en los registros. Para obtener más información, consulte Mitigar el riesgo de exposición de contraseñas al utilizar el registro de consultas.

all

Registra todas las instrucciones. Para depuración, se recomienda esta configuración.

ddl

Registra todas las instrucciones del lenguaje de definición de datos (DDL), como CREATE, ALTER, DROP, etc.

MOD

Registra todas las instrucciones DDL y las instrucciones de lenguaje de manipulación de datos (DML), como INSERT, UPDATE y DELETE, que modifican los datos.

none

No se registra ninguna instrucción SQL. Recomendamos esta configuración para evitar el riesgo de exponer las contraseñas en los registros.

log_min_duration_statement

Se registra cualquier instrucción SQL que se ejecute al menos durante el período de tiempo especificado o durante más tiempo. Por defecto, este parámetro no está configurado. Si se activa este parámetro, puede resultar más sencillo encontrar consultas no optimizadas.

–1–2147483647

El número de milisegundos (ms) de tiempo de ejecución durante los cuales se registra una instrucción.

Para configurar el registro de consultas

Estos pasos suponen que su clúster de base de datos de Aurora PostgreSQL utiliza un grupo de parámetros del clúster de base de datos personalizado.

1. Establezca el parámetro `log_statement` como `all`. En el siguiente ejemplo se muestra la información que se escribe en el archivo con esta configuración de parámetros.

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
---------------------- END OF LOG ----------------------
```

2. Establezca el parámetro `log_min_duration_statement`. En el siguiente ejemplo se muestra la información que se escribe en el archivo `postgresql.log`cuando se establece el parámetro en:1

   Se registran las consultas que superan la duración especificada en el parámetro `log_min_duration_statement`. A continuación se muestra un ejemplo. Puede ver el archivo de registro de su clúster de base de datos de Aurora PostgreSQL en la consola de Amazon RDS.

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
```

```
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
----------------------- END OF LOG ----------------------
```

Mitigar el riesgo de exposición de contraseñas al utilizar el registro de consultas

Le recomendamos que mantenga `log_statement` establecido en `none` para evitar exponer las contraseñas. Si establece `log_statement` en `all`, `ddl` o `mod`, le recomendamos que siga uno o más de los siguientes pasos.

- Para el cliente, cifre la información confidencial. Para obtener más información, consulte Encryption Options en la documentación de PostgreSQL. Utilice las opciones `ENCRYPTED` (y `UNENCRYPTED`) de las instrucciones `CREATE` y `ALTER`. Para obtener más información, consulte CREATE USER en la documentación de PostgreSQL.
- Para su clúster de base de datos de Aurora PostgreSQL, configure y utilice la extensión de auditoría de PostgreSQL (pgAudit). Esta extensión redacta la información confidencial de las instrucciones CREATE y ALTER enviadas al registro. Para obtener más información, consulte Uso de pgAudit para registrar la actividad de la base de datos.
- Restrinja el acceso a los Registros de CloudWatch.
- Utilice mecanismos de autenticación más sólidos como IAM.

Supervisión de llamadas a la API de Amazon Aurora en AWS CloudTrail

AWS CloudTrail es un servicio de AWS que ayuda a auditar la cuenta de AWS. AWS CloudTrail se activa en la cuenta de AWS cuando esta se crea. Para obtener más información acerca de CloudTrail, consulte la Guía del usuario de AWS CloudTrail.

Temas

- Integración de CloudTrail con Amazon Aurora
- Entradas de archivos de registro de Amazon Aurora

Integración de CloudTrail con Amazon Aurora

Todas las acciones de Amazon Aurora se registran en CloudTrail. CloudTrail proporciona un registro de las acciones que realiza un usuario, un rol o un servicio de AWS en Amazon Aurora.

Eventos de CloudTrail

CloudTrail captura las llamadas a la API de Amazon Aurora como eventos. Un evento representa una solicitud específica realizada desde un origen y contiene información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. Los eventos incluyen las llamadas realizadas desde la consola de Amazon RDS y las llamadas desde el código a las operaciones de la API de Amazon RDS.

La actividad de Amazon Aurora se registra en un evento de CloudTrail de Event history (Historial de eventos). Puede utilizar la consola de CloudTrail para ver los últimos 90 días de actividad y eventos de API registrados en una región de AWS. Para obtener más información, consulte Visualización de eventos con el historial de eventos de CloudTrail.

Registros de seguimiento de CloudTrail

Para mantener un registro continuo de eventos en la cuenta de AWS, incluidos los eventos de Amazon Aurora, cree una traza. Un seguimiento es una configuración que permite la entrega de eventos en un bucket de Amazon S3 especificado. CloudTrail normalmente entrega archivos de registro en el plazo de 15 minutos después de producirse la actividad de la cuenta.

> **Note**
>
> Si no configura un registro de seguimiento, puede ver los eventos más recientes en la consola de CloudTrail en el Event history (Historial de eventos).

Puede crear dos tipos de registros de seguimiento en una cuenta de AWS : uno que se aplique a todas las regiones o uno que se aplique a una región específica. De manera predeterminada, cuando crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones.

También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte:

- Introducción a la creación de registros de seguimiento
- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de registro de CloudTrail desde varias regiones y Recibir archivos de registro de CloudTrail desde varias cuentas

Entradas de archivos de registro de Amazon Aurora

Los archivos log de CloudTrail pueden contener una o varias entradas de log. Los archivos de registro de CloudTrail no son un rastro de la stack ordenado de las llamadas a las API públicas, por lo que no aparecen en ningún orden específico.

En el ejemplo siguiente, se muestra una entrada de registro de CloudTrail que ilustra la acción `CreateDBInstance`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
```

```
        "userName": "johndoe"
    },
    "eventTime": "2018-07-30T22:14:06Z",
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
            "audit",
            "error",
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
```

```
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
        "readReplicaDBInstanceIdentifiers": [],
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1e"},
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1f"},
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1c"},
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1d"},
                    "subnetIdentifier": "subnet-42599426",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1a"},
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetStatus": "Active"
                }
            ],
            "vpcId": "vpc-136a4c6a",
            "subnetGroupStatus": "Complete"
        },
        "masterUsername": "myawsuser",
        "multiAZ": false,
        "autoMinorVersionUpgrade": true,
```

```
        "engine": "mysql",
        "cACertificateIdentifier": "rds-ca-2015",
        "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
        "dBSecurityGroups": [],
        "pendingModifiedValues": {
            "masterUserPassword": "****",
            "pendingCloudwatchLogsExports": {
                "logTypesToEnable": [
                    "audit",
                    "error",
                    "general",
                    "slowquery"
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Tal y como se muestra en el elemento `userIdentity` del ejemplo anterior, cada evento o entrada del registro contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario de IAM.

- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información sobre `userIdentity`, consulte Elemento userIdentity de CloudTrail. Para obtener más información sobre `CreateDBInstance` y otras acciones de Amazon Aurora, consulte la Referencia de la API de Amazon RDS.

Supervisión de Amazon Aurora con flujos de actividad de la base de datos

Mediante la característica de flujos de actividad de la base de datos, puede supervisar flujos de actividad de la base de datos prácticamente en tiempo real.

Temas

- Información general sobre flujos de actividad de la base de datos
- Requisitos previos de red para flujos de actividad de la base de datos de Aurora MySQL
- Inicio de una secuencia de actividades de la base de datos
- Obtención del estado de un flujo de actividad de la base de datos
- Detención de un flujo de actividad de la base de datos
- Monitoreo de secuencias de actividades de la base de datos
- Ejemplos de políticas de IAM para flujos de actividad de base de datos

Información general sobre flujos de actividad de la base de datos

Como administrador de base de datos de Amazon Aurora, debe proteger su base de datos y cumplir los requisitos normativos y de conformidad. Una estrategia es integrar flujos de actividad de la base de datos con sus herramientas de monitoreo. De esta manera, monitoriza y configura alarmas para la actividad de auditoría en su clúster de Amazon Aurora.

Las amenazas de seguridad son tanto externas como internas. Para protegerse contra amenazas internas, puede controlar el acceso del administrador a los flujos de datos mediante la configuración de la característica de flujos de actividad de la base de datos. Los DBA de no tienen acceso a la recopilación, transmisión, almacenamiento ni procesamiento de los flujos.

Contenido

- Cómo funcionan los flujos de actividad de la base de datos
- Modo asíncrono y síncrono para flujos de actividad de la base de datos
- Requisitos y limitaciones de los flujos de actividad de la base de datos
- Disponibilidad en regiones y versiones
- Clases de instancia de base de datos admitidas para los flujos de actividad de la base de datos

Cómo funcionan los flujos de actividad de la base de datos

En Amazon Aurora, el flujo de actividad de la base de datos se inicia en el nivel del clúster. Todas las instancias de base de datos del clúster tienen habilitadas los flujos de actividad de la base de datos.

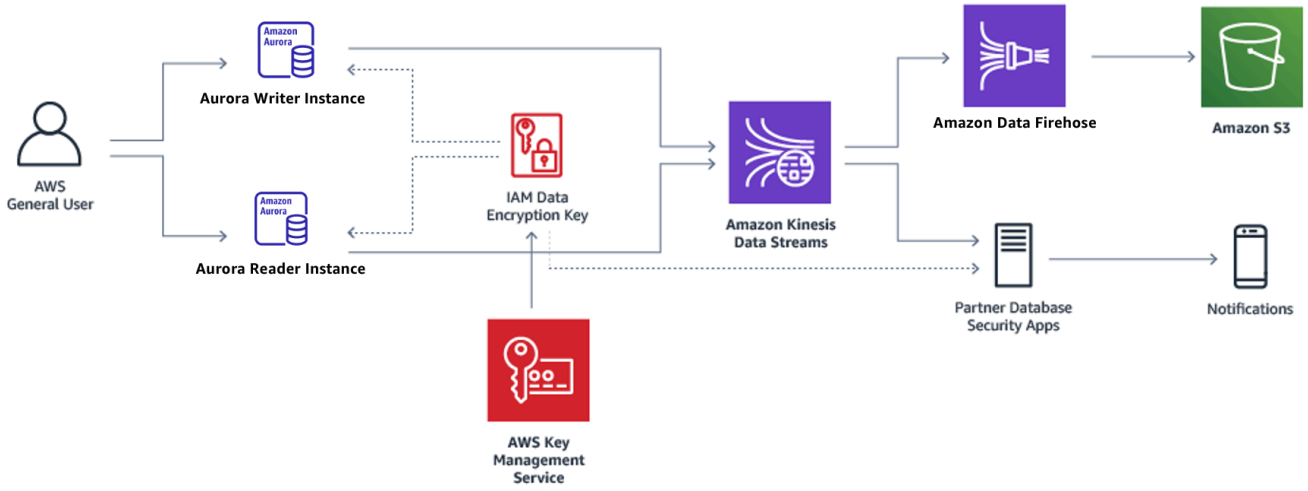
Su clúster de base de datos de Aurora envía actividades a un flujo de datos de Amazon Kinesis prácticamente en tiempo real. El flujo de Kinesis se crea automáticamente. Desde Kinesis, puede configurar servicios de AWS como Amazon Data Firehose y AWS Lambda para utilizar el flujo y almacenar los datos.

> **Important**
>
> La característica de flujo de actividad de la base de datos en Amazon Aurora se puede utilizar de forma gratuita, pero Amazon Kinesis cobra por un flujo de datos. Para obtener más información, consulte los Precios de Amazon Kinesis Data Streams.

Si utiliza una base de datos global de Aurora, inicie un flujo de actividad de base de datos en cada clúster de base de datos por separado. Cada clúster suministra datos de auditoría a su propio flujo de Kinesis dentro de su propia Región de AWS. Los flujos de actividad no funcionan de forma diferente durante una conmutación por error. Siguen auditando su base de datos global como siempre.

Puede configurar aplicaciones para administrar la conformidad para consumir los flujos de actividad de la base de datos. Para Aurora PostgreSQL, las aplicaciones de conformidad incluyen Security Guardium de IBM y SecureSphere Database Audit and Protection de Imperva. Esas aplicaciones pueden utilizar el flujo para generar alertas y auditar la actividad del clúster de base de datos de Aurora.

En el gráfico siguiente, se muestra un clúster de bases de datos de Aurora configurado con Amazon Data Firehose.



Modo asíncrono y síncrono para flujos de actividad de la base de datos

Puede elegir que la sesión de la base de datos gestione los eventos de actividad de la base de datos en cualquiera de los siguientes modos:

- Modo asíncrono: cuando una sesión de la base de datos genere un evento de flujo de actividad, la sesión volverá de inmediato a las actividades normales. En segundo plano, el evento de flujo de actividad se convertirá en un registro permanente. Si se produce un error en la tarea en segundo plano, se enviará un evento RDS. Dicho evento marca el principio y el fin de todo período de tiempo en el que podrían haberse perdido registros de eventos de la secuencia de actividades.

  El modo asíncrono favorece el rendimiento de la base de datos con respecto a la precisión de la secuencia de actividades.

  > **Note**
  >
  > El modo asíncrono está disponible tanto para Aurora PostgreSQL como para Aurora MySQL.

- Modo síncrono: cuando una sesión de la base de datos genera un evento de flujo de actividad, la sesión bloquea otras actividades hasta que el evento sea permanente. Si, por algún motivo, no se puede conseguir que el evento sea permanente, la sesión de la base de datos volverá a las actividades normales. Sin embargo, se enviará un evento RDS para indicar que podrían haberse

perdido registros de la secuencia de actividades durante algún tiempo. Cuando el estado del sistema vuelva a ser bueno, se enviará otro evento RDS.

El modo síncrono favorece la precisión de la secuencia de actividades con respecto al rendimiento de la base de datos.

> **Note**
>
> El modo sincrónico está disponible para Aurora PostgreSQL. No puede usar el modo sincrónico con Aurora MySQL.

Requisitos y limitaciones de los flujos de actividad de la base de datos

En Aurora, los flujos de actividad de la base de datos tienen los límites y los requisitos siguientes:

- Los flujos de actividad de la base de datos requieren Amazon Kinesis.
- Se requiere AWS Key Management Service (AWS KMS) porque los flujos de actividad de la base de datos siempre están cifrados.
- La aplicación de cifrado adicional al flujo de datos de Amazon Kinesis no es compatible con los flujos de actividad de la base de datos, que ya están cifrados con su clave AWS KMS.
- Inicie el flujo de actividad de la base de datos en el nivel de clúster de la base de datos. Si agrega una instancia de base de datos a su clúster, no deberá iniciar un flujo de actividad en la instancia pues se audita automáticamente.
- Si utiliza una base de datos global de Aurora, inicie un flujo de actividad en cada clúster de base de datos por separado. Cada clúster suministra datos de auditoría a su propio flujo de Kinesis dentro de su propia Región de AWS.
- En Aurora PostgreSQL, asegúrese de detener el flujo de actividad de la base de datos antes de realizar una actualización de la versión principal. Puede iniciar el flujo de actividad de la base de datos una vez finalizada la actualización.

Disponibilidad en regiones y versiones

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos de Aurora y entre Regiones de AWS. Para obtener más información sobre la disponibilidad de versiones y regiones con Aurora y las secuencias de actividades de base de datos, consulte

Regiones y motores de base de datos Aurora admitidos para los flujos de actividad de bases de datos.

Clases de instancia de base de datos admitidas para los flujos de actividad de la base de datos

Para Aurora MySQL, puede usar flujos de actividad de la base de datos con las siguientes clases de instancia de base de datos:

- db.r8g.*large
- db.r7g.*large
- db.r7i.*large
- db.r6g.*large
- db.r6i.*large
- db.r5.*large
- db.x2g.*

Para Aurora PostgreSQL, puede usar flujos de actividad de la base de datos con las siguientes clases de instancia de base de datos:

- db.r8g.*large
- db.r7i.*large
- db.r7g.*large
- db.r6g.*large
- db.r6i.*large
- db.r6id.*large
- db.r5.*large
- db.r4.*large
- db.x2g.*

Requisitos previos de red para flujos de actividad de la base de datos de Aurora MySQL

En la siguiente sección, encontrará cómo configurar la nube virtual privada (VPC) para utilizarla con flujos de actividades de la base de datos.

> **Note**
>
> Los requisitos previos de red de Aurora MySQL se aplican a las siguientes versiones del motor:
>
> - Aurora MySQL versión 2, hasta la versión 2.11.3
> - Aurora MySQL versión 2.12.0
> - Aurora MySQL versión 3, hasta la versión 3.04.2

Temas

- Requisitos previos para los puntos de enlace de AWS KMS
- Requisitos previos para la disponibilidad pública
- Requisitos previos para la disponibilidad privada

Requisitos previos para los puntos de enlace de AWS KMS

Las instancias de un clúster de Aurora MySQL que utilizan secuencias de actividades deben poder tener acceso a los puntos de enlace de AWS KMS. Asegúrese de que se cumple este requisito antes de habilitar secuencias de actividades de la base de datos para el clúster de Aurora MySQL. Si el clúster de Aurora está disponible públicamente, este requisito se cumple automáticamente.

> **Important**
>
> Si el clúster de bases de datos de Aurora MySQL no puede acceder al punto de enlace de AWS KMS, el flujo de actividad deja de funcionar. En ese caso, Aurora le notifica sobre este problema mediante eventos RDS.

Requisitos previos para la disponibilidad pública

Para que un clúster de bases de datos de Aurora sea público, debe cumplir con los requisitos siguientes:

- Publicly Accessible (Accesible públicamente) debe estar configurado en Yes (Sí) en la página de detalles del clúster de Consola de administración de AWS.
- El clúster de bases de datos debe encontrarse en una subred pública de una Amazon VPC. Para obtener más información acerca de las instancias de bases de datos con acceso público, consulte Uso de una clúster de base de datos en una VPC. Para obtener más información acerca de las subredes de Amazon VPC públicas, consulte VPC y subredes.

Requisitos previos para la disponibilidad privada

Si el clúster de base de datos de Aurora se encuentra en una subred pública de una VPC y no es de acceso público, es privado. Para mantener el clúster privado y utilizarlo con secuencias de actividades de base de datos, puede elegir una de las opciones siguientes:

- Configurar la traducción de direcciones de red (NAT) en la VPC. Para obtener más información, consulte Puerta de enlace NAT.
- Crear un punto de enlace de AWS KMS en la VPC. Se recomienda esta opción porque su configuración es más sencilla.

Crear un punto de enlace de AWS KMS en la VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, elija Puntos de conexión.
3. Elija Crear punto de conexión.

   Aparecerá la página Create Endpoint (Creación de un punto de enlace).
4. Haga lo siguiente:

   - En Service category (Categoría de servicios), elija AWS Services (Servicios).
   - En Nombre del servicio, elija com.amazonaws.***región***.kms, donde *región* es la Región de AWS en la que se encuentra el clúster.
   - Para la VPC, elija la VPC en la que se encuentra su clúster.

5. Elija Create Endpoint (Crear punto de conexión).

Para obtener más información acerca de la configuración de puntos de enlace de la VPC, consulte Puntos de enlace de la VPC.

Inicio de una secuencia de actividades de la base de datos

Para supervisar la actividad de la base de datos de todas las instancias de la base de datos del clúster de Aurora, inicie un flujo de actividad en el nivel del clúster. Todas las instancias de base de datos que se agreguen al clúster también se monitorearán automáticamente. Si utiliza una base de datos global de Aurora, inicie un flujo de actividad de base de datos en cada clúster de base de datos por separado. Cada clúster suministra datos de auditoría a su propio flujo de Kinesis dentro de su propia Región de AWS.

Cuando inicie un flujo de actividad, todos los eventos de actividad de la base de datos que haya configurado en la política de auditoría generarán un evento del flujo de actividad. Los eventos de acceso se generan a partir de comandos SQL como `CONNECT` y `SELECT`. Los eventos de cambio se generan a partir de comandos SQL como `CREATE` y `INSERT`.

Console

Inicio de un flujo de actividad de la base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos en la que desea iniciar un flujo de actividad.
4. En Actions (Acciones), elija Start activity stream (Iniciar secuencia de actividades).

   Aparecerá la ventana Start database activity stream: *name* (Iniciar el flujo de actividad de la base de datos: nombre), donde *name* (nombre) equivale a su clúster de bases de datos.
5. Ingrese la siguiente configuración:

   - En AWS KMS key, seleccione una clave de la lista de AWS KMS keys.

> **Note**
>
> Si el clúster de Aurora MySQL no puede acceder a las claves de KMS, siga las instrucciones de Requisitos previos de red para flujos de actividad de la base de datos de Aurora MySQL para habilitar primero dicho acceso.

Aurora utiliza la clave de KMS para cifrar la clave que, a su vez, cifra la actividad de la base de datos. Elija una clave de KMS distinta de la clave predeterminada. Para obtener más información sobre las claves de cifrado y AWS KMS, consulte ¿Qué es AWS Key Management Service? en la guía para desarrolladores de AWS Key Management Service.

- En Database activity stream mode (Modo de secuencia de actividades de base de datos), elija Asynchronous (Asíncrono) o Synchronous (Síncrono).

> **Note**
>
> Esta opción solo se aplica a Aurora PostgreSQL. Para Aurora MySQL, solo puede usar el modo asíncrono.

- Elija Immediately (De inmediato).

  Cuando elige Immediately (De inmediato), el clúster de bases de datos se reinicia de inmediato. Si elige During the next maintenance window (Durante el siguiente periodo de mantenimiento), el clúster de bases de datos no se reinicia de inmediato. En este caso, la secuencia de actividades de la base de datos no se inicia hasta la siguiente ventana de mantenimiento.

6. Seleccione Start database activity stream (Iniciar secuencia de actividades de base de datos).

   El estado del clúster de bases de datos muestra que el flujo de actividad se está iniciando.

> **Note**
>
> Si aparece el error `You can't start a database activity stream in this configuration`, compruebe Clases de instancia de base de datos admitidas

para los flujos de actividad de la base de datos para ver si su clúster de base de datos utiliza una clase de instancia compatible.

AWS CLI

Para empezar a transmitir la actividad de la base de datos de un clúster de base de datos , configure el clúster de base de datos mediante el comando start-activity-stream de la AWS CLI.

- `--resource-arn` *arn*: especifica el nombre de recurso de Amazon (ARN) del clúster de base de datos.
- `--mode` *sync-or-async*: especifica el modo síncrónico (`sync`) o asíncrono (`async`). Para Aurora PostgreSQL, puede elegir cualquiera de los valores. Para Aurora MySQL, especifique `async`.
- `--kms-key-id` *key*: especifica el identificador de clave KMS para cifrar mensajes en el flujo de actividad de la base de datos. El identificador de clave de KMS AWS es el ARN clave, el ID de clave, el ARN de alias o el nombre de alias de la AWS KMS key.

El siguiente ejemplo inicia un flujo de actividad de la base de datos para un clúster de base de datos en modo asíncrono.

Para Linux, macOS o Unix:

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-cluster-arn \
    --apply-immediately
```

Para Windows:

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-cluster-arn ^
    --apply-immediately
```

Amazon RDS API

Para iniciar flujos de actividad de base de datos de un clúster de base de datos, configure el clúster mediante la operación StartActivityStream.

Llame a la acción con los siguientes parámetros:

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`

> **Note**
>
> Si recibe un error que indica que no puede iniciar un flujo de actividad de base de datos con la versión actual de la base de datos de Aurora PostgreSQL, aplique el último parche para Aurora PostgreSQL antes de iniciar un flujo de actividad de base de datos. Para obtener información sobre la actualización de la base de datos de Aurora PostgreSQL, consulte Actualización de clústeres de base de datos de Amazon Aurora.
> A continuación, se presentan las versiones de parches mínimas para iniciar flujos de actividad de base de datos con Aurora PostgreSQL.
>
> - 3.4.15 (11.9.15), 11.21.10
> - 12.9.15, 12.15.9, 12.16.10, 12.17.7, 12.18.5, 12.19.4, 12.20.3, 12.22.3
> - 13.9.12, 13.11.9, 13.12.10, 13.13.7, 13.14.5, 13.15.4, 13.16.3, 13.18.3
> - 14.6.12, 14.8.9, 14.9.10, 14.10.7, 14.11.5, 14.12.4, 14.13.3, 14.15.3
> - 15.3.9, 15.4.10, 15.5.7, 15.6.5, 15.7.4, 15.8.3, 15.10.3
> - 16.1.7, 16.2.5, 16.3.4, 16.4.3, 16.6.3

Obtención del estado de un flujo de actividad de la base de datos

Puede obtener el estado de un flujo de actividad mediante la consola o AWS CLI.

Consola

Obtención del estado de un flujo de actividad de la base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, luego, el enlace del clúster de bases de datos.
3. Elija la pestaña Configuración y consulte el estado en Secuencia de actividades de base de datos.

AWS CLI

Puede usar la configuración del flujo de actividad para un clúster de base de datos como respuesta a una solicitud describe-db-clusters de la CLI.

En el siguiente ejemplo, se describe *my-cluster*.

```
aws rds --region my-region describe-db-clusters --db-cluster-identifier my-cluster
```

El ejemplo siguiente muestra una respuesta JSON. Se muestran los siguientes campos:

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`
- `ActivityStreamStatus`
- `ActivityStreamMode`
-

Estos campos son los mismos para Aurora PostgreSQL y Aurora MySQL, excepto que `ActivityStreamMode` siempre es `async` para Aurora MySQL, mientras que para Aurora PostgreSQL, podría ser `sync` o `async`.

```
{
    "DBClusters": [
        {
      "DBClusterIdentifier": "my-cluster",
            ...
            "ActivityStreamKinesisStreamName": "aws-rds-das-cluster-
A6TSYXITZCZXJHIRVFUBZ5LTWY",
```

```
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "12345678-abcd-efgh-ijkl-bd041f170262",
            "ActivityStreamMode": "async",
            "DbClusterResourceId": "cluster-ABCD123456"
            ...
        }
    ]
}
```

API de RDS

Puede usar la configuración del flujo de actividad de un clúster de bases de datos como respuesta a una operación DescribeDBClusters .

Detención de un flujo de actividad de la base de datos

Puede detener una secuencia de actividades mediante la consola o AWS CLI.

Si elimina su clúster de bases de datos, el flujo de actividad se detiene y el flujo subyacente de Amazon Kinesis se elimina automáticamente.

Consola

Para desactivar una secuencia de actividades

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija un clúster de bases de datos cuyo flujo de actividad de la base de datos quiera detener.
4. En Actions (Acciones), elija Stop activity stream (Detener secuencia de actividades). Se visualizará la ventana Database Activity Stream (Secuencia de actividades de base de datos).
    a. Elija Immediately (De inmediato).

        Cuando elige Immediately (De inmediato), el clúster de bases de datos se reinicia de inmediato. Si elige During the next maintenance window (Durante el siguiente periodo de mantenimiento), el clúster de bases de datos no se reinicia de inmediato. En este caso, el flujo de actividad de la base de datos no se detiene hasta el siguiente periodo de mantenimiento.
    b. Elija Continue.

AWS CLI

Para detener flujos de actividad de la base de datos de su clúster de bases de datos, configure el clúster de bases de datos ejecutando el comando stop-activity-stream de AWS CLI. Identifique la región de AWS del clúster de bases de datos mediante el parámetro `--region`. El parámetro `--apply-immediately` es opcional.

Para Linux, macOS o:Unix

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_CLUSTER_ARN \
    --apply-immediately
```

En:Windows

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_CLUSTER_ARN ^
    --apply-immediately
```

API de RDS

Para detener los flujos de actividad del clúster de bases de datos, configure el clúster mediante la operación StopActivityStream. Identifique la región de AWS del clúster de bases de datos mediante el parámetro `Region`. El parámetro `ApplyImmediately` es opcional.

Monitoreo de secuencias de actividades de la base de datos

Los flujos de actividad de la base de datos monitorean y notifican las actividades. La secuencia de actividades se recopila y se transmite a Amazon Kinesis. Desde Kinesis, puede monitorear la secuencia de actividad, o bien otros servicios y aplicaciones pueden consumir la secuencia de actividades para un análisis posterior. Puede encontrar el nombre del flujo de Kinesis subyacente mediante el comando `describe-db-clusters` de la AWS CLI o la operación de la API de RDS `DescribeDBClusters`.

Aurora administra el flujo de Kinesis de la siguiente manera:

- Aurora crea el flujo de Kinesis automáticamente con un periodo de retención de 24 horas.
- Aurora escala el flujo de Kinesis si es necesario.

- Si detiene el flujo de actividad de la base de datos o elimina el clúster de bases de datos, Aurora elimina el flujo de Kinesis.

Las categorías de actividad siguientes se monitorizan y se ponen en el registro de auditoría de secuencias de actividades:

- Comandos SQL: todos los comandos SQL se auditan, así como las instrucciones preparadas, las funciones integradas y las funciones en lenguaje de procedimientos para SQL (PL/SQL). Las llamadas a procedimientos almacenados se auditan. Cualquier instrucción SQL emitida dentro de procedimientos o funciones almacenados también se auditan.
- Otra información de la base de datos: la actividad monitoreada incluye la instrucción SQL completa, el recuento de las filas afectadas de los comandos DML, los objetos a los que se accede y el nombre único de la base de datos. Para Aurora PostgreSQL, los flujos de actividad de la base de datos también monitorean las variables de enlace y los parámetros del procedimiento almacenados.

  > **Important**
  >
  > El texto SQL completo de cada instrucción está visible en el registro de auditoría de secuencia de actividades, incluida la información confidencial. Sin embargo, las contraseñas de usuario de base de datos se redactan siAurora las puede determinar a partir del contexto, tal y como pasa con la siguiente instrucción SQL.
  >
  > ```
  > ALTER ROLE role-name WITH password
  > ```

- Información de conexión: la actividad monitorizada incluye la información de sesión y de red, el ID de proceso del servidor y los códigos de salida.

Si un flujo de actividad tiene un error mientras monitorea una instancia de base de datos, se lo notificará mediante eventos RDS.

En las siguientes secciones, puede acceder a los flujos de actividad de base de datos, auditarlos y procesarlos.

Temas

- Acceso a un flujo de actividad desde Amazon Kinesis

- Ejemplos y contenido sobre el registro de auditoría en flujos de actividad de bases de datos
- Matriz JSON databaseActivityEventList para flujos de actividad de base de datos
- Procesamiento de un flujo de actividad de la base de datos mediante SDK de AWS

Acceso a un flujo de actividad desde Amazon Kinesis

Cuando habilite un flujo de actividad para un clúster de bases de datos, se creará un flujo de Kinesis para usted. En Kinesis podrá monitorizar la actividad de la base de datos en tiempo real. Para profundizar en el análisis de la actividad de la base de datos, puede conectar su secuencia de Kinesis a aplicaciones de consumidor. También puede conectar el flujo de datos con aplicaciones de administración de conformidad como Security Guardium de IBM o SecureSphere Database Audit and Protection de Imperva.

Puede acceder a su transmisión de Kinesis desde la consola de RDS o la consola de Kinesis.

Para acceder a una secuencia de actividades desde Kinesis utilizando la consola de RDS

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster de base de datos en la que desea iniciar un flujo de actividad.
4. Elija Configuration (Configuración).
5. En Database activity stream (Secuencia de actividad de base de datos), seleccione el enlace en Kinesis stream (Secuencia de Kinesis).
6. En la colsola de Kinesis, elija Monitoring (Supervisión) para empezar a observar la actividad de la base de datos.

Para acceder a una secuencia de actividades desde Kinesis utilizando la consola de Kinesis

1. Abra la consola de Kinesis en https://console.aws.amazon.com/kinesis.
2. Elija la secuencia de actividades en la lista de secuencias de Kinesis.

   El nombre de un flujo de actividad consta de un prefijo `aws-rds-das-cluster-` seguido del ID de recurso del clúster de bases de datos. A continuación se muestra un ejemplo.

   ```
   aws-rds-das-cluster-NHVOV4PCLWHGF52NP
   ```

Para utilizar la consola de Amazon RDS para encontrar el ID de recurso del clúster de bases de datos, elija su clúster de bases de datos en la lista de bases de datos y, luego, elija la pestaña Configuration (Configuración).

Para utilizar la AWS CLI para encontrar el nombre completo del flujo de Kinesis de un flujo de actividad, utilice una solicitud describe-db-clústers de la CLI y anote el valor de `ActivityStreamKinesisStreamName` en la respuesta.

3. Elija Monitoring (Monitorización) para empezar a observar la actividad de la base de datos.

Para obtener más información acerca del uso de Amazon Kinesis, consulte ¿Qué es Amazon Kinesis Data Streams?.

Ejemplos y contenido sobre el registro de auditoría en flujos de actividad de bases de datos

Los eventos monitoreados se representan en el flujo de actividad de la base de datos como cadenas JSON. La estructura está formada por un objeto JSON que contiene un `DatabaseActivityMonitoringRecord`, el cual, a su vez, contiene una matriz `databaseActivityEventList` de eventos de actividad.

> **Note**
>
> Para los flujos de actividad de la base de datos, la matriz JSON de `paramList` no incluye valores nulos de las aplicaciones de Hibernate.

Temas

- Ejemplos de un registro de auditoría de flujo de actividad de la base de datos
- Objeto JSON DatabaseActivityMonitoringRecords
- Objeto JSON databaseActivityEvents

Ejemplos de un registro de auditoría de flujo de actividad de la base de datos

A continuación mostramos registros de auditoría JSON descifrados de muestra de registros de eventos de actividad.

Example Registro de eventos de actividad de una instrucción Aurora PostgreSQL instrucción CONNECT SQL

El siguiente registro de eventos de actividad muestra un inicio de sesión con el uso de una instrucción SQL `CONNECT` (`command`) por parte de un cliente psql (`clientApplication`).


```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-10-30 00:39:49.940668+00",
          "logTime": "2019-10-30 00:39:49.990579+00",
          "statementId": 1,
          "substatementId": 1,
          "objectType": null,
          "command": "CONNECT",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "49804",
          "sessionId": "5ce5f7f0.474b",
          "rowCount": null,
          "commandText": null,
          "paramList": [],
          "pid": 18251,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "MISC",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
```

```
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Registro de evento de actividad de una instrucción SQL CONNECT de Aurora MySQL

El siguiente registro de eventos de actividad muestra un inicio de sesión donde un cliente mysql (`clientApplication`) usa una instrucción SQL `CONNECT` (`command`).

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:13.267214+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"rdsadmin",
      "databaseName":"",
      "remoteHost":"localhost",
      "remotePort":"11053",
      "command":"CONNECT",
      "commandText":"",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"",
      "statementId":0,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725121",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:13.267207+00",
      "endTime":"2020-05-22 18:07:13.267213+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
```

```
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

Example Registro de evento de actividad de una instrucción CREATE TABLE de Aurora PostgreSQL

En el siguiente ejemplo, se muestra un evento `CREATE TABLE` para Aurora PostgreSQL.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:36:54.403455+00",
          "logTime": "2019-05-24 00:36:54.494235+00",
          "statementId": 2,
          "substatementId": 1,
          "objectType": null,
          "command": "CREATE TABLE",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": null,
          "commandText": "create table my_table (id serial primary key, name
 varchar(32));",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "DDL",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
```

```
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Registro de evento de actividad de una instrucción CREATE TABLE de Aurora MySQL

En el siguiente ejemplo, se muestra una instrucción `CREATE TABLE` para Aurora MySQL. La operación se representa como dos registros de eventos independientes. Un evento tiene `"class":"MAIN"`. El otro evento tiene `"class":"AUX"`. Los mensajes pueden llegar en cualquier orden. El campo `logTime` del evento `MAIN` es siempre anterior a los campos `logTime` de cualquier evento `AUX` correspondiente.

En el ejemplo siguiente se muestra el evento con un valor `class` de `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.250221+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"CREATE TABLE test1 (id INT)",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":1,
      "exitCode":"0",
```

```
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.250222+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

El siguiente ejemplo muestra el evento correspondiente con un valor `class` de AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.247182+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"CREATE",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"725118",
      "rowCount":0,
```

```
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.247182+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Example Registro de evento de actividad de una instrucción SELECT

En el siguiente ejemplo, se muestra un evento `SELECT` .

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:39:49.920564+00",
          "logTime": "2019-05-24 00:39:49.940668+00",
          "statementId": 6,
          "substatementId": 1,
          "objectType": "TABLE",
          "command": "SELECT",
          "objectName": "public.my_table",
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": 10,
          "commandText": "select * from my_table;",
          "paramList": [],
```

```
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "READ",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
```

```
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x010500000000000515000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```


Example Registro de evento de actividad de una instrucción SELECT de Aurora MySQL

En el siguiente ejemplo, se muestra un evento `SELECT`.

En el ejemplo siguiente se muestra el evento con un valor `class` de `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986467+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"SELECT * FROM test1 WHERE id < 28",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"726571",
      "rowCount":2,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986467+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

El siguiente ejemplo muestra el evento correspondiente con un valor `class` de `AUX`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986399+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"READ",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"726571",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986399+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Objeto JSON DatabaseActivityMonitoringRecords

Los registros de eventos de actividad de la base de datos se encuentran en un objeto JSON que contiene la siguiente información.

Campo JSON	Tipo de datos	Descripción
`type`	string	Tipo de registro JSON. El valor es `DatabaseActivityMonitoringRecords` .
`version`	string	La versión de los registros de monitoreo de actividad de la base de datos. La versión de los registros de actividad de la base de datos generados dependen de la versión del motor del clúster de bases de datos. • Los registros de actividad de la base de datos de la versión 1.1 se generan para clústeres de base de datos de Aurora PostgreSQL que ejecutan las versiones 10.10 y posteriores del motor y las versiones 11.5 y posteriores. • Los registros de actividad de base de datos de la versión 1.0 se generan para clústeres de base de datos de Aurora PostgreSQL que ejecutan las versiones 10.7 y 11.4 del motor. Todos los campos siguientes están en la versión 1.0 y en la versión 1.1, excepto donde se indique específicamente.
databaseActivityEvents	cadena	Un objeto JSON que contiene los eventos de actividad.
key	cadena	Una clave de cifrado que se utiliza para descifrar el Matriz de JSON databaseActivityEventList.

Objeto JSON databaseActivityEvents

El objeto JSON `databaseActivityEvents` contiene la siguiente información.

Campos de nivel superior en el registro JSON

Cada evento del registro de auditoría se envuelve dentro de un registro en formato JSON. Este registro contiene los siguientes campos.

type

Este campo siempre tiene el valor `DatabaseActivityMonitoringRecords`.

version

Este campo representa la versión del protocolo o contrato de datos del flujo de actividad de la base de datos. Define los campos que están disponibles.

La versión 1.0 representa el soporte de secuencias de actividades de datos originales para Aurora PostgreSQL versiones 10.7 y 11.4. La versión 1.1 representa el soporte de secuencias de actividades de datos para Aurora PostgreSQL versiones 10.10 y posteriores y Aurora PostgreSQL 11.5 y posteriores. La versión 1.1 incluye los campos adicionales `errorMessage` y `startTime`. La versión 1.2 representa el soporte de secuencias de actividad de datos para Aurora MySQL 2.08 y superior. La versión 1.2 incluye los campos adicionales `endTime` y `transactionId`.

databaseActivityEvents

Una cadena cifrada que representa uno o más eventos de actividad. Se representa como una matriz de bytes base64. Al descifrar la cadena, el resultado es un registro en formato JSON con campos, tal y como se muestra en los ejemplos de esta sección.

key

Clave de datos cifrada utilizada para cifrar la cadena `databaseActivityEvents`. Esta es la misma AWS KMS key que proporcionó cuando inició la secuencia de actividades de la base de datos.

En el ejemplo siguiente se muestra el formato de este registro.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
```

```
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

Siga estos pasos para descifrar el contenido del campo:`databaseActivityEvents`

1. Descifrar el valor en el campo JSON `key` mediante la clave de KMS que proporcionó al iniciar la secuencia de actividades de la base de datos. Al hacerlo, se devuelve la clave de cifrado de datos en texto sin cifrar.
2. Decodifique en base64 el valor en el campo JSON `databaseActivityEvents` para obtener el texto cifrado, en formato binario, de la carga útil de auditoría.
3. Descifrar el texto cifrado binario con la clave de cifrado de datos que decodificó en el primer paso.
4. Descomprimir la carga útil descifrada.
   - La carga cifrada está en el campo `databaseActivityEvents`.
   - El campo `databaseActivityEventList` contiene una matriz de registros de auditoría. Los campos `type` de la matriz pueden ser `record` o `heartbeat`.

Un registro de evento de actividad de registro de auditoría es un objeto JSON que contiene la información siguiente.

Campo JSON	Tipo de datos	Descripción
`type`	string	Tipo de registro JSON. El valor es `DatabaseActivityMonitoringRecord` .
`clusterId`	string	Identificador del recurso del clúster de bases de datos. Corresponde al atributo del clúster de bases de datos `DbClusterResourceId` .
`instanceId`	string	El identificador del recurso de instancia de base de datos. Corresponde al atributo de instancia de base de datos `DbiResourceId` .
Matriz de JSON databaseActivityEventList	string	Una matriz de registros de auditoría de actividad o mensajes de latido.

Matriz JSON databaseActivityEventList para flujos de actividad de base de datos

La carga de registro de auditoría es una matriz JSON `databaseActivityEventList` cifrada. En las tablas siguientes, se enumeran alfabéticamente los campos de cada evento de actividad de la matriz `DatabaseActivityEventList` descifrada de un registro de auditoría. Los campos son diferentes en función de si utiliza Aurora PostgreSQL o Aurora MySQL. Consulte la tabla que se aplica al motor de base de datos.

> **Important**
>
> La estructura de los eventos está sujeta a cambio. Aurora podría agregar nuevos campos a eventos de actividad en el futuro. En las aplicaciones que analizan los datos JSON, asegúrese de que el código puede ignorar o tomar las acciones adecuadas para nombres de campo desconocidos.

Campos de databaseActivityEventList para Aurora PostgreSQL

A continuación, encontrará campos `databaseActivityEventList` para Aurora PostgreSQL.

Campo	Tipo de datos	Descripción
`class`	string	Clase de evento de actividad. Los valores válidos para Aurora PostgreSQL son los siguientes: • `ALL` • `CONNECT`: evento de conexión o desconexión. • `DDL`: instrucción DDL que no está incluida en la lista de instrucciones de la clase `ROLE`. • `FUNCTION`: llamada de función o bloque `DO`. • `MISC`: comando variado como, por ejemplo, `DISCARD`, `FETCH`, `CHECKPOINT` o `VACUUM`. • `NONE` • `READ`: instrucción `SELECT` o `COPY` donde el origen es una relación o una consulta.

Campo	Tipo de datos	Descripción
		• `ROLE`: instrucción relacionada con roles y privilegios como `GRANT`, `REVOKE` y `CREATE/ALTER/DROP ROLE`. • `WRITE`: instrucción `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` o `COPY` cuando el destino es una relación.
`clientApplication`	string	Aplicación que el cliente ha usado para establecer conexión según notificación del cliente. No es obligatorio que el cliente notifique esta información, por lo que el valor puede ser nulo.
`command`	string	Nombre del comando SQL sin ningún detalle del comando.
`commandText`	string	Instrucción SQL real que el usuario ha pasado. Para Aurora PostgreSQL, el valor es idéntico a la instrucción SQL original. Este campo se usa para todo tipo de registros, salvo para registros de conexión o desconexión, en cuyo caso el valor es nulo. **Important** El texto SQL completo de cada instrucción está visible en el registro de auditoría de secuencia de actividades, incluida la información confidencial. Sin embargo, las contraseñas de usuario de la base de datos se redactan si se Aurora puede determinar a partir del contexto, tal y como pasa con la siguiente instrucción SQL. `ALTER ROLE role-name WITH password`
`databaseName`	string	Base de datos a la que se ha conectado el usuario.
`dbProtocol`	string	El protocolo de base de datos, por ejemplo `Postgres 3.0`.

Campo	Tipo de datos	Descripción
`dbUserName`	string	Usuario de base de datos que el cliente ha usado para autenticarse.
`errorMessage` (solo registros de actividad de la base de datos de la versión 1.1)	string	Si hubo algún error, este campo se rellena con el mensaje de error que habría generado el servidor de base de datos. El valor `errorMessage` es nulo para las declaraciones normales que no dieron lugar a un error. Un error se define como cualquier actividad que produzca un evento de registro de errores de PostgreSQL visible para el cliente en un nivel de gravedad de `ERROR` o superior. Para obtener más información, consulte Niveles de gravedad de mensajes de PostgreSQL. Por ejemplo, los errores de sintaxis y las cancelaciones de consultas generan un mensaje de error. Los errores internos del servidor PostgreSQL, como los errores de proceso del generador de puntos de comprobación en segundo plano, no generan un mensaje de error. Sin embargo, los registros de tales eventos se siguen emitiendo independientemente de la configuración del nivel de gravedad del registro. Esto evita que los atacantes desactiven el registro para intentar evitar la detección. Consulte también el campo `exitCode`.

Campo	Tipo de datos	Descripción
`exitCode`	int	Valor usado para un registro de salida de sesión. Si la salida es limpia, contiene el código de salida. No siempre se puede obtener un código de salida en algunos escenarios de error. Por ejemplo, este es el caso cuando PostgreSQL genera un `exit()` o si un operador ejecuta un comando `kill -9`. Si se produjo algún error, el campo `exitCode` muestra el código de error SQL, `SQLSTATE`, como se muestra en Códigos de error de PostgreSQL. Consulte también el campo `errorMessage` .
`logTime`	string	Marca temporal según se ha registrado en la ruta del código de auditoría. Esto representa la hora de finalización de la ejecución de la instrucción SQL. Consulte también el campo `startTime` .
`netProtocol`	string	Protocolo de comunicación de red
`objectName`	string	Nombre del objeto de base de datos si la instrucción SQL opera en uno. Este campo solo se usa cuando la instrucción SQL funciona en un objeto de base de datos. Si la instrucción SQL no opera en un objeto, este valor es nulo.

Campo	Tipo de datos	Descripción
`objectType`	string	Tipo de objeto de base de datos como mesa, índice, vista, etc. Este campo solo se usa cuando la instrucción SQL funciona en un objeto de base de datos. Si la instrucción SQL no opera en un objeto, este valor es nulo. Entre los valores válidos se incluyen los siguientes: • `COMPOSITE TYPE` • `FOREIGN TABLE` • `FUNCTION` • `INDEX` • `MATERIALIZED VIEW` • `SEQUENCE` • `TABLE` • `TOAST TABLE` • `VIEW` • `UNKNOWN`
`paramList`	string	Matriz de parámetros separados por comas que se transfier e a la instrucción SQL. Si la instrucción SQL no tiene parámetros, este valor es una matriz vacía.
`pid`	int	ID del proceso de backend que se asigna para atender la conexión cliente.
`remoteHost`	string	La dirección IP del cliente o el nombre de host. Para Aurora PostgreSQL, el que se utiliza depende de la configuración del parámetro `log_hostname` de la base de datos. El valor `remoteHost` también incluye `[local]` y `localhost`, que indican actividad del usuario `rdsadmin`.
`remotePort`	cadena	Número de puerto del cliente.

Campo	Tipo de datos	Descripción
`rowCount`	int	Número de filas de tabla que la instrucción SQL recupera o a las que afecta. Este campo se usa únicamente para instrucciones SQL que son instrucciones de lenguaje de manipulación de datos (DML). Si la instrucción SQL no es una instrucción DML, este valor es nulo.
`serverHost`	cadena	Dirección IP del host de servidor de la base de datos. El valor `serverHost` también incluye `[local]` y `localhost`, que indican actividad del usuario `rdsadmin`.
`serverType`	cadena	Tipo de servidor de base de datos; por ejemplo, `PostgreSQL`.
`serverVersion`	string	La versión del servidor de base de datos, por ejemplo `2.3.1` para Aurora PostgreSQL.
`serviceName`	string	Nombre del servicio, por ejemplo `Amazon Aurora PostgreSQL-Compatible edition`.
`sessionId`	int	Identificador de sesión pseudoúnico.
`sessionId`	int	Identificador de sesión pseudoúnico.
`startTime` (solo registros de actividad de la base de datos de la versión 1.1)	string	La hora a la que comenzó la ejecución de la instrucción SQL. Para calcular el tiempo de ejecución aproximado de la instrucción SQL, utilice `logTime - startTime`. Consulte también el campo `logTime`.
`statementId`	int	Un identificador de la instrucción SQL del cliente. El contador está en el nivel de sesión y aumenta con cada instrucción SQL que el cliente introduce.

Campo	Tipo de datos	Descripción
`substatementId`	int	Un identificador de una subinstrucción SQL. Este valor cuenta las subinstrucciones contenidas por cada instrucción SQL que el campo `statementId` identifica.
`type`	string	Tipo de evento. Los valores válidos son `record` o `heartbeat` .

Campos de databaseActivityEventList para Aurora MySQL

A continuación, encontrará campos `databaseActivityEventList` para Aurora MySQL.

Campo	Tipo de datos	Descripción
`class`	string	Clase de evento de actividad. Los valores válidos para Aurora MySQL son los siguientes: • `MAIN`: el evento principal que representa una instrucción SQL. • `AUX`: un evento complementario que contiene detalles adicionales. Por ejemplo, una instrucción que cambia el nombre de un objeto puede tener un evento con clase `AUX` que refleje el nuevo nombre. Para buscar eventos `MAIN` y `AUX` que correspondan a la misma instrucción, compruebe si hay distintos eventos que tengan los mismos valores para el campo `pid` y para el campo `statementId` .
`clientApplication`	string	Aplicación que el cliente ha usado para establecer conexión según notificación del cliente. No es obligatorio que el cliente notifique esta información, por lo que el valor puede ser nulo.

Campo	Tipo de datos	Descripción
command	string	La categoría general de la instrucción SQL. Los valores de este campo dependen del valor de class. Los valores cuando class es MAIN incluyen lo siguiente: • CONNECT: cuando se conecta una sesión de cliente. • QUERY: una instrucción SQL. Acompañado de uno o más eventos con un valor class de AUX. • DISCONNECT : cuando se desconecta una sesión de cliente. • FAILED_CONNECT : cuando un cliente intenta conectarse pero no puede hacerlo. • CHANGEUSER – Un cambio de estado que forma parte del protocolo de red MySQL, no de una instrucción que emita. Los valores cuando class es AUX incluyen lo siguiente: • READ: instrucción SELECT o COPY donde el origen es una relación o una consulta. • WRITE: instrucción INSERT, UPDATE, DELETE, TRUNCATE o COPY cuando el destino es una relación. • DROP: eliminación de un objeto. • CREATE: creación de un objeto. • RENAME: cambio de nombre de un objeto. • ALTER: cambio de las propiedades de un objeto.

Campo	Tipo de datos	Descripción
`commandText`	string	Para los eventos con un valor `class` de `MAIN`, este campo representa la instrucción SQL real que el usuario ha pasado. Este campo se usa para todo tipo de registros, salvo para registros de conexión o desconexión, en cuyo caso el valor es nulo. Para los eventos con un valor `class` de `AUX`, este campo contiene información adicional sobre los objetos involucrados en el evento. Para Aurora MySQL, los caracteres como comillas van precedidos de una barra diagonal inversa, que representa un carácter de escape. **Important** El texto SQL completo de cada instrucción está visible en el registro de auditoría, incluida la información confidencial. Sin embargo, las contraseñas de usuario de la base de datos se redactan si se Aurora puede determinar a partir del contexto, tal y como pasa con la siguiente instrucción SQL. `mysql> SET PASSWORD = 'my-password ';` **Note** Especifique una contraseña distinta de la que se muestra aquí como práctica recomendada de seguridad.
`databaseName`	cadena	Base de datos a la que se ha conectado el usuario.

Campo	Tipo de datos	Descripción
`dbProtocol`	string	Protocolo de la base de datos. Actualmente, este valor es siempre `MySQL` para Aurora MySQL.
`dbUserName`	string	Usuario de base de datos que el cliente ha usado para autenticarse.
`endTime` (solo registros de actividad de la base de datos de la versión 1.2)	string	La hora a la que finalizó la ejecución de la instrucción SQL. Se representa en formato de hora universal coordinada (UTC). Para calcular el tiempo de ejecución de la instrucción SQL, utilice `endTime - startTime` . Consulte también el campo `startTime` .
`errorMessage` (solo registros de actividad de la base de datos de la versión 1.1)	string	Si hubo algún error, este campo se rellena con el mensaje de error que habría generado el servidor de base de datos. El valor `errorMessage` es nulo para las declaraciones normales que no dieron lugar a un error. Un error se define como cualquier actividad que produciría un evento de registro de errores de MySQL visible para el cliente en un nivel de gravedad de `ERROR` o superior. Para obtener más información, consulte The Error Log en el Manual de referencia de MySQL. Por ejemplo, los errores de sintaxis y las cancelaciones de consultas generan un mensaje de error. Los errores internos del servidor MySQL, como los errores de proceso del generador de puntos de comprobación en segundo plano, no generan un mensaje de error. Sin embargo, los registros de tales eventos se siguen emitiendo independientemente de la configuración del nivel de gravedad del registro. Esto evita que los atacantes desactive n el registro para intentar evitar la detección. Consulte también el campo `exitCode`.

Campo	Tipo de datos	Descripción
`exitCode`	int	Valor usado para un registro de salida de sesión. Si la salida es limpia, contiene el código de salida. No siempre se puede obtener un código de salida en algunos escenarios de error. En tales casos, este valor puede ser cero o puede estar en blanco.
`logTime`	string	Marca temporal según se ha registrado en la ruta del código de auditoría. Se representa en formato de hora universal coordinada (UTC). Para obtener la forma más precisa de calcular la duración de la instrucción, consulte los campos `startTime` y `endTime`.
`netProtocol`	string	Protocolo de comunicación de red Actualmente, este valor es siempre `TCP` para Aurora MySQL.
`objectName`	string	Nombre del objeto de base de datos si la instrucción SQL opera en uno. Este campo solo se usa cuando la instrucción SQL funciona en un objeto de base de datos. Si la instrucción SQL no opera en un objeto, este valor es nulo. Para crear el nombre completo del objeto, combine `databaseName` y `objectName`. Si la consulta implica a varios objetos, este campo puede ser una lista de nombres separados por comas.
`objectType`	string	El tipo de objeto de base de datos como tabla, índice, vista, etc. Este campo solo se usa cuando la instrucción SQL funciona en un objeto de base de datos. Si la instrucción SQL no opera en un objeto, este valor es nulo. Los valores válidos de Aurora MySQL incluyen lo siguiente: • `INDEX` • `TABLE` • `UNKNOWN`

Campo	Tipo de datos	Descripción
`paramList`	string	Este campo no se utiliza para Aurora MySQL y siempre es nulo.
`pid`	int	ID del proceso de backend que se asigna para atender la conexión cliente. Cuando se reinicia el servidor de base de datos, los cambios `pid` y el contador para el campo `statementId` se inicia de nuevo.
`remoteHost`	string	La dirección IP o el nombre de host del cliente que emitió la instrucción SQL. Para Aurora MySQL, el que se utiliza depende de la configuración del parámetro `skip_name_resolve` de la base de datos. El valor `localhost` indica la actividad del usuario especial `rdsadmin`.
`remotePort`	string	Número de puerto del cliente.
`rowCount`	int	La cantidad de filas devueltas por la instrucción SQL. Por ejemplo, si una instrucción SELECT devuelve 10 filas, rowCount es 10. Para las instrucciones INSERT o UPDATE, rowCount es 0.
`serverHost`	cadena	El identificador de instancia del servidor de base de datos.
`serverType`	cadena	Tipo de servidor de base de datos; por ejemplo, `MySQL`.
`serverVersion`	string	Versión del servidor de base de datos. Actualmente, este valor es siempre `MySQL 5.7.12` para Aurora MySQL.
`serviceName`	string	Nombre del servicio de Actualmente, este valor es siempre `Amazon Aurora MySQL` para Aurora MySQL.
`sessionId`	int	Identificador de sesión pseudoúnico.

Campo	Tipo de datos	Descripción
`startTime` (solo registros de actividad de la base de datos de la versión 1.1)	string	La hora a la que comenzó la ejecución de la instrucción SQL. Se representa en formato de hora universal coordinada (UTC). Para calcular el tiempo de ejecución de la instrucción SQL, utilice `endTime - startTime` . Consulte también el campo `endTime`.
`statementId`	int	Un identificador de la instrucción SQL del cliente. El contador aumenta con cada instrucción SQL introducida por el cliente. El contador se restablece cuando se reinicia la instancia de base de datos.
`substatementId`	int	Un identificador de una subinstrucción SQL. Este valor es 1 para eventos con clase `MAIN` y 2 para eventos con clase `AUX`. Utilice el campo `statementId` para identificar todos los eventos generados por la misma instrucción.
`transactionId` (solo registros de actividad de la base de datos de la versión 1.2)	int	Un identificador de una transacción.
`type`	string	Tipo de evento. Los valores válidos son `record` o `heartbeat` .

Procesamiento de un flujo de actividad de la base de datos mediante SDK de AWS

Puede procesar una secuencia de actividades mediante programación con AWS SDK. A continuación, mostramos ejemplos de Java y Python totalmente funcionales sobre cómo puede procesar el flujo de datos de Kinesis.

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
```

```
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {

    private static final String STREAM_NAME = "aws-rds-das-[cluster-external-
resource-id]";
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String DBC_RESOURCE_ID = "[cluster-external-resource-id]";
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
    private static final BasicAWSCredentials CREDENTIALS = new
 BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
    private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
 AWSStaticCredentialsProvider(CREDENTIALS);

    private static final AwsCrypto CRYPTO = new AwsCrypto();
    private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
            .withRegion(REGION_NAME)
            .withCredentials(CREDENTIALS_PROVIDER).build();

    class Activity {
        String type;
        String version;
        String databaseActivityEvents;
        String key;
    }

    class ActivityEvent {
        @SerializedName("class") String _class;
        String clientApplication;
        String command;
```

```
        String commandText;
        String databaseName;
        String dbProtocol;
        String dbUserName;
        String endTime;
        String errorMessage;
        String exitCode;
        String logTime;
        String netProtocol;
        String objectName;
        String objectType;
        List<String> paramList;
        String pid;
        String remoteHost;
        String remotePort;
        String rowCount;
        String serverHost;
        String serverType;
        String serverVersion;
        String serviceName;
        String sessionId;
        String startTime;
        String statementId;
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId;
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }

    static class RecordProcessorFactory implements IRecordProcessorFactory {
        @Override
        public IRecordProcessor createProcessor() {
            return new RecordProcessor();
        }
    }

    static class RecordProcessor implements IRecordProcessor {
```

```
        private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
        private static final int PROCESSING_RETRIES_MAX = 10;
        private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
        private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

        private static final Cipher CIPHER;
        static {
            Security.insertProviderAt(new BouncyCastleProvider(), 1);
            try {
                CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
            } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
                throw new ExceptionInInitializerError(e);
            }
        }

        private long nextCheckpointTimeInMillis;

        @Override
        public void initialize(String shardId) {
        }

        @Override
        public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
            for (final Record record : records) {
                processSingleBlob(record.getData());
            }

            if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
                checkpoint(checkpointer);
                nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
            }
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }
```

```
       private void processSingleBlob(final ByteBuffer bytes) {
           try {
               // JSON $Activity
               final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

               // Base64.Decode
               final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
               final byte[] decodedDataKey = Base64.decode(activity.key);

               Map<String, String> context = new HashMap<>();
               context.put("aws:rds:dbc-id", DBC_RESOURCE_ID);

               // Decrypt
               final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
               final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
               final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));

               // GZip Decompress
               final byte[] decompressed = decompress(decrypted);
               // JSON $ActivityRecords
               final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

               // Iterate throught $ActivityEvents
               for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                   System.out.println(GSON.toJson(event));
               }
           } catch (Exception e) {
               // Handle error.
               e.printStackTrace();
           }
       }

       private static byte[] decompress(final byte[] src) throws IOException {
           ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
```

```
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
                    break;
                } catch (ShutdownException se) {
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
```

```
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
            return out.toByteArray();
        }
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
CREDENTIALS_PROVIDER, workerId);

kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
```

```
    }
}
```

Python

```
import base64
import json
import zlib
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # cluster-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-cluster-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL

class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                        wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)

    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
```

```
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)


def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],

 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:dbc-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
```

```
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

Ejemplos de políticas de IAM para flujos de actividad de base de datos

Cualquier usuario que tenga privilegios de rol de AWS Identity and Access Management (IAM) apropiados para los flujos de actividad de la base de datos puede crear, iniciar, detener y modificar la configuración del flujo de actividad de un clúster de bases de datos. Estas acciones se incluyen en el registro de auditoría de la secuencia. Como práctica recomendada de cumplimiento, le recomendamos que no proporcione estos privilegios a los DBA.

Establezca el acceso a las secuencias de actividades de base de datos mediante políticas IAM. Para obtener más información acerca de la autenticación de Aurora, consulte Administración de la identidad y el acceso en Amazon Aurora. Para obtener más información sobre la creación de políticas de IAM, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM.

Example Política para permitir configurar secuencias de actividades de la base de datos

Para dar a los usuarios un acceso detallado con el fin de modificar flujos de actividad, utilice las claves de contexto de operación específica del servicio `rds:StartActivityStream` y `rds:StopActivityStream` de una política de IAM. En el siguiente ejemplo de política de IAM el usuario o rol pueden configurar secuencias de actividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureActivityStreams",
            "Effect": "Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Example Política para permitir iniciar secuencias de actividades de la base de datos

En el siguiente ejemplo de política de IAM el usuario o rol pueden iniciar secuencias de actividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
    ]
}
```

Example Política para permitir detener secuencias de actividades de la base de datos

En el siguiente ejemplo de política de IAM el usuario o rol pueden detener secuencias de actividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Política para rechazar iniciar secuencias de actividades de la base de datos

En el siguiente ejemplo de política de IAM se evita que un usuario o rol inicie secuencias de actividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Política para rechazar la detención de secuencias de actividades de la base de datos

En el siguiente ejemplo de política de IAM se evita que un usuario o rol detenga secuencias de actividades.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Supervisión de amenazas con Amazon GuardDuty para protección de RDS para Amazon Aurora

Amazon GuardDuty es un servicio de detección de amenazas que ayuda a proteger las cuentas, los contenedores, las cargas de trabajo y los datos de su entorno de AWS. Mediante modelos de machine learning (ML) y capacidades de detección de anomalías y amenazas, GuardDuty supervisa continuamente los diferentes orígenes de registro y la actividad en tiempo de ejecución para identificar y priorizar los posibles riesgos de seguridad y actividades maliciosas en su entorno.

GuardDuty para protección de RDS analiza y perfila los eventos de inicio de sesión para detectar posibles amenazas de acceso a sus bases de datos de Amazon Aurora. Al activar Protección de RDS, GuardDuty consume los eventos de inicio de sesión de RDS de sus bases de datos de Aurora. RDS Protection supervisa estos eventos y los perfila para detectar posibles amenazas internas o agentes externos.

Para obtener más información sobre la forma de habilitar la protección de RDS de Amazon GuardDuty, consulte GuardDuty RDS Protection (Protección de RDS de Amazon GuardDuty) en la Guía del usuario de Amazon GuardDuty.

Cuando RDS Protection detecta una amenaza potencial, como un patrón inusual de intentos de inicio de sesión con éxito o fallidos, GuardDuty genera un nuevo hallazgo con detalles sobre la base de datos potencialmente comprometida. Puede ver los detalles de los resultados en la sección de resumen de los resultados de la consola de Amazon GuardDuty. Los detalles de los resultados varían según el tipo de resultado. Los detalles principales, el tipo de recurso y el rol del recurso determinan el tipo de información disponible para cualquier resultado. Para obtener más información sobre los detalles más comunes disponibles en los resultados y los tipos de resultados, consulte Finding details (Detalles de resultados) y GuardDuty RDS Protection (Protección de RDS de GuardDuty) respectivamente, en la Guía del usuario de Amazon GuardDuty.

Puede activar o desactivar la característica de protección RDS para cualquier Cuenta de AWS en cualquier Región de AWS donde esté disponible. Cuando Protección de RDS no está habilitado, GuardDuty no detecta las bases de datos de Aurora que puedan estar comprometidas ni proporciona detalles sobre el problema.

Una cuenta de GuardDuty existente puede habilitar la protección de RDS con un periodo de prueba de 30 días. En una cuenta nueva de GuardDuty, la protección de RDS ya está habilitada e incluida en el periodo de prueba gratuito de 30 días. Para obtener más información, consulte Estimating GuardDuty cost (Cálculo del coste de GuardDuty) en la Guía del usuario de Amazon GuardDuty.

Para obtener información sobre las Regiones de AWS en las que GuardDuty aún no admite la protección de RDS, consulte Disponibilidad de características específicas para cada región en la Guía del usuario de Amazon GuardDuty.

Para obtener información sobre las versiones de la base de datos de Aurora compatibles con GuardDuty RDS Protection, consulte Bases de datos ilimitadas compatibles con Amazon Aurora, Amazon RDS y Aurora en la Guía del usuario de Amazon GuardDuty.

Uso de Amazon Aurora MySQL

Amazon Aurora MySQL es un motor de base de datos relacional completamente administrado, compatible con MySQL que combina la velocidad y la fiabilidad de las bases de datos comerciales de tecnología avanzada con la sencillez y la rentabilidad de las bases de datos de código abierto. Aurora MySQL es un reemplazo instántaneo para MySQL que simplifica y hace más rentable configurar, usar y escalar las implementaciones de MySQL nuevas y existentes, lo que le permitirá centrarse en su negocio y sus aplicaciones. Amazon RDS proporciona la administración de Aurora gestionando las tareas de base de datos rutinarias como el aprovisionamiento, la aplicación de parches, las copias de seguridad, la recuperación, la detección de errores y la reparación. Amazon RDS también brinda herramientas de migración que le permiten convertir sus aplicaciones de Amazon RDS for MySQL a Aurora MySQL con un solo botón.

Temas

- Información general de Amazon Aurora MySQL
- Seguridad con Amazon Aurora MySQL
- Actualización de aplicaciones para la conexión a los clústeres de base de datos de MySQL de Aurora con los nuevos certificados TLS
- Uso de la autenticación Kerberos para Aurora MySQL
- Migración de datos a un clúster de base de datos de Amazon Aurora MySQL
- Administración de Amazon Aurora MySQL
- Ajuste de Aurora MySQL
- Consulta paralela para Amazon Aurora MySQL
- Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL
- Replicación con Amazon Aurora MySQL
- Uso del reenvío de escritura local en un clúster de base de datos de Amazon Aurora MySQL
- Integración de Amazon Aurora MySQL con otros servicios de AWS
- Modo lab de Amazon Aurora MySQL
- Prácticas recomendadas con Amazon Aurora MySQL
- Solución de problemas de rendimiento de las bases de datos Amazon Aurora MySQL
- Referencia de Amazon Aurora MySQL

- Actualizaciones del motor de base de datos de Amazon Aurora MySQL

Información general de Amazon Aurora MySQL

En las siguientes secciones, encontrará información general de Amazon Aurora MySQL.

Temas

- Mejoras del rendimiento de Amazon Aurora MySQL
- Amazon Aurora MySQL y los datos espaciales
- Aurora MySQL versión 3 compatible con MySQL 8.0
- Aurora MySQL versión 2 compatible con MySQL 5.7

Mejoras del rendimiento de Amazon Aurora MySQL

Amazon Aurora incluye mejoras del desempeño para responder a las distintas necesidades de las bases de datos comerciales de gama alta.

Inserción rápida

La inserción rápida acelera las inserciones paralelas ordenadas por clave principal y se aplica específicamente a las declaraciones `LOAD DATA` e `INSERT INTO ... SELECT ...`. La inserción rápida almacena en caché la posición de un cursor en un recorrido del índice mientras se ejecuta la declaración. Esto evita tener que recorrer el índice de nuevo sin necesidad.

La inserción rápida solo se ha habilitado para tablas InnoDB estándar en Aurora MySQL (versión 3.03.2 o superior). Esta optimización no funciona para las tablas temporales de InnoDB. Se ha deshabilitado en la versión 2 de Aurora MySQL para todas las versiones 2.11 y 2.12. La optimización de la inserción rápida solo funciona si se ha deshabilitado la optimización del índice hash adaptativo.

Puede monitorizar las siguientes métricas para determinar la eficacia de la inserción rápida para su clúster de bases de datos:

- `aurora_fast_insert_cache_hits`: un contador que se incrementa cuando el cursor en caché se recupera y se verifica correctamente.
- `aurora_fast_insert_cache_misses`: un contador que se incrementa cuando el cursor en caché deja de ser válido y Aurora realiza un recorrido normal del índice.

Puede recuperar el valor actual de las métricas de inserción rápida con el siguiente comando:

```
mysql> show global status like 'Aurora_fast_insert%';
```

Obtendrá un resultado similar al siguiente:

```
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| Aurora_fast_insert_cache_hits   | 3598300   |
| Aurora_fast_insert_cache_misses | 436401336 |
+---------------------------------+-----------+
```

Amazon Aurora MySQL y los datos espaciales

La siguiente lista resume las principales características espaciales de Aurora MySQL y explica cómo se corresponden con las características espaciales en MySQL:

- La versión 2 de Aurora MySQL admite los mismos tipos de datos espaciales y funciones de relaciones espaciales que MySQL 5.7. Para obtener más información sobre estos tipos de datos y funciones, consulte Tipos de datos espaciales y Funciones de relaciones espaciales en la documentación de MySQL 5.7.
- La versión 3 de Aurora MySQL admite los mismos tipos de datos espaciales y funciones de relaciones espaciales que MySQL 8.0. Para obtener más información sobre estos tipos de datos y funciones, consulte Tipos de datos espaciales y Funciones de relaciones espaciales en la documentación de MySQL 8.0.
- Aurora MySQL admite la indexación espacial en tablas InnoDB. La indexación espacial mejora el desempeño de las consultas en conjuntos de datos grandes, para consultas sobre datos espaciales. En MySQL, la indexación espacial para tablas InnoDB está disponible en MySQL 5.7 y 8.0.

  Aurora MySQL usa una estrategia de indexación espacial diferente a MySQL para un alto rendimiento con consultas espaciales. La implementación del índice espacial de Aurora utiliza una curva de relleno de espacio en un árbol B, que está destinada a proporcionar un rendimiento mayor para los escaneos de rango espacial que un árbol R.

> **Note**
>
> En Aurora MySQL, una transacción de una tabla con un índice espacial definido en una columna con un identificador de referencia espacial (SRID) no puede realizar inserciones en un área seleccionada para que la actualice otra transacción.

Las siguientes declaraciones en el lenguaje de definición de datos (DDL) se admiten para crear índices en columnas que usan tipos de datos espaciales.

CREATE TABLE

Puede usar las palabras clave `SPATIAL INDEX` en una declaración `CREATE TABLE` para añadir un índice espacial a una columna en una tabla nueva. A continuación se muestra un ejemplo.

```
CREATE TABLE test (shape POLYGON NOT NULL, SPATIAL INDEX(shape));
```

ALTER TABLE

Puede usar las palabras clave `SPATIAL INDEX` en una declaración `ALTER TABLE` para añadir un índice espacial a una columna en una tabla existente. A continuación se muestra un ejemplo.

```
ALTER TABLE test ADD SPATIAL INDEX(shape);
```

CREATE INDEX

Puede usar la palabra clave `SPATIAL` en una declaración `CREATE INDEX` para añadir un índice espacial a una columna en una tabla existente. A continuación se muestra un ejemplo.

```
CREATE SPATIAL INDEX shape_index ON test (shape);
```

Aurora MySQL versión 3 compatible con MySQL 8.0

Puede utilizar Aurora MySQL versión 3 para obtener las últimas funciones compatibles con MySQL, mejoras de rendimiento y correcciones de errores. A continuación, puede obtener información sobre Aurora MySQL versión 3, con compatibilidad MySQL 8.0. Puede obtener información sobre cómo actualizar sus clústeres y aplicaciones a Aurora MySQL versión 3.

Algunas características de Aurora, como Aurora Serverless v2, requieren Aurora MySQL versión 3.

Temas

- Características de MySQL 8.0 Community Edition
- Requisito previo de Aurora MySQL versión 3 para Aurora MySQL Serverless v2
- Notas de la versión 3 de Aurora MySQL
- Nuevas optimizaciones de consultas paralelas
- Optimizaciones para reducir el tiempo de reinicio de la base de datos
- Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3
- Comparación entre Aurora MySQL versión 2 y Aurora MySQL versión 3
- Comparación de Aurora MySQL versión 3 y MySQL 8.0 Community Edition
- Actualización a Aurora MySQL versión 3

Características de MySQL 8.0 Community Edition

La versión inicial de Aurora MySQL versión 3 es compatible con MySQL 8.0.23 Community Edition. MySQL 8.0 presenta varias funciones nuevas, entre ellas las siguientes:

- Compatibilidad con el lenguaje de definición de datos (DDL) atómicos. Para obtener más información, consulte Compatibilidad con el lenguaje de definición de datos (DDL) atómicos.
- Funciones JSON. Para obtener más información, consulte Funciones JSON en el Manual de referencia de MySQL.
- Funciones de ventana. Para obtener más información, consulte Funciones de ventana en el Manual de referencia de MySQL.
- Expresiones de tabla comunes (CTE), utilizando la cláusula `WITH`. Para obtener más información, consulte WITH (expresiones de tabla comunes) en el Manual de referencia de MySQL.
- Cláusulas `ADD COLUMN` y `RENAME COLUMN` optimizadas para la instrucción `ALTER TABLE`. Estas optimizaciones se denominan “DDL instantáneo”. Aurora MySQL versión 3 es compatible con la característica DDL instantánea de la comunidad MySQL. No se usa la antigua característica DDL rápida de Aurora. Para obtener información de uso de DDL instantáneo, consulte DDL instantáneo (Aurora MySQL versión 3).
- Índices descendentes, funcionales e invisibles. Para obtener más información, consulte Índices invisibles, Índices descendentes, e Instrucción CREATE INDEX en el Manual de referencia de MySQL.

- Privilegios basados en roles controlados mediante instrucciones SQL. Para obtener más información sobre los cambios en el modelo de privilegios, consulte Modelo de privilegios basado en roles.
- Cláusulas `NOWAIT` y `SKIP LOCKED` con la instrucción `SELECT ... FOR SHARE`. Estas cláusulas evitan esperar a que otras transacciones liberen bloqueos de fila. Para obtener más información, consulte Lecturas de bloqueo en el Manual de referencia de MySQL Reference.
- Mejoras en la replicación del registro binario (binlog). Para obtener información detallada sobre Aurora MySQL, consulte Reproducción de registros binarios. En particular, puede realizar una replicación filtrada. Para obtener información de uso sobre la replicación filtrada, consulte Cómo evalúan los servidores las reglas de filtrado de replicación en el Manual de referencia de MySQL.
- Sugerencias. Algunas de las sugerencias compatibles con MySQL 8.0 ya estaban respaldadas en Aurora MySQL versión 2. Para obtener información sobre el uso de las sugerencias con Aurora MySQL, consulte Sugerencias de Aurora MySQL. Para obtener lista completa de sugerencias en la comunidad MySQL 8.0, consulte Sugerencias del optimizador en el Manual de referencia de MySQL.

Para obtener la lista completa de funciones añadidas a la edición de la comunidad MySQL 8.0, consulte la entrada del blog Lista completa de las nuevas características de MySQL 8.0.

Aurora MySQL versión 3 también incluye cambios en las palabras clave para un lenguaje inclusivo, respaldado desde la comunidad MySQL 8.0.26. Para obtener más detalles acerca de estos cambios, consulte Cambios de idioma inclusivos para Aurora MySQL versión 3.

Requisito previo de Aurora MySQL versión 3 para Aurora MySQL Serverless v2

Tener Aurora MySQL versión 3 es un requisito previo para todas las instancias de base de datos de un clúster de Aurora MySQL Serverless v2. Aurora MySQL Serverless v2 es compatible con instancias del lector en un clúster de base de datos y otras características de Aurora que no están disponibles para Aurora MySQL Serverless v1. También tiene un escalado más rápido y granular que Aurora MySQL Serverless v1.

Notas de la versión 3 de Aurora MySQL

Para ver las notas de todas las versiones de Aurora MySQL versión 3, consulte el tema sobre actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 3 en las notas de la versión de Aurora MySQL.

Nuevas optimizaciones de consultas paralelas

La optimización de consultas paralelas de Aurora se aplica ahora a más operaciones SQL:

- La consulta paralela se aplica ahora a las tablas que contienen los tipos de datos `TEXT`, `BLOB`, `JSON`, `GEOMETRY`, y `VARCHAR` y `CHAR` de más de 768 bytes.
- La consulta paralela puede optimizar las consultas que implican tablas particionadas.
- Las consultas paralelas pueden optimizar consultas que implican llamadas a funciones agregadas en la lista de selección y en la cláusula `HAVING`.

Para obtener más información acerca de estas mejoras, consulte Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL. Para obtener información general sobre la consulta paralela de Aurora, consulte Consulta paralela para Amazon Aurora MySQL.

Optimizaciones para reducir el tiempo de reinicio de la base de datos

Su clúster de base de datos Aurora MySQL debe tener alta disponibilidad durante las interrupciones previstas e imprevistas.

Los administradores de bases de datos deben realizar un mantenimiento ocasional de la base de datos. Estas tareas de mantenimiento incluyen la aplicación de parches en la base de datos, las actualizaciones, las modificaciones de parámetros de la base de datos que requieren un reinicio manual, la realización de una conmutación por error para reducir el tiempo que tardan los cambios de clase de instancia, etc. Estas acciones previstas requieren un tiempo de inactividad.

Sin embargo, el tiempo de inactividad también puede deberse a acciones imprevistas, como una conmutación por error inesperada debida a un fallo de hardware subyacente o a la limitación de recursos de la base de datos. Todas estas acciones previstas e imprevistas provocan el reinicio de la base de datos.

En la versión 3.05 de Aurora MySQL y versiones posteriores, hemos introducido optimizaciones que reducen el tiempo de reinicio de la base de datos. Estas optimizaciones reducen hasta en un 65 % el tiempo de inactividad y las interrupciones de las cargas de trabajo de la base de datos tras un reinicio.

Durante el inicio de la base de datos, se inicializan muchos componentes de la memoria interna. El componente de mayor tamaño es el grupo de búferes de InnoDB, que en Aurora MySQL ocupa el 75 % del tamaño de la memoria de la instancia de forma predeterminada. Nuestras pruebas

han revelado que el tiempo de inicialización es proporcional al tamaño de la reserva de búferes de InnoDB y, por tanto, aumenta con el tamaño de la clase de instancia de base de datos. Durante la fase de inicialización, la base de datos no puede aceptar conexiones, lo que prolonga el tiempo de inactividad durante los reinicios. La primera fase del reinicio rápido de Aurora MySQL optimiza la inicialización del grupo de búferes, lo que reduce el tiempo de inicialización de la base de datos y, por lo tanto, reduce el tiempo total de reinicio.

Para obtener más información, consulte el blog Reduzca el tiempo de inactividad con las optimizaciones del tiempo de reinicio de bases de datos Amazon Aurora MySQL.

Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3

La versión 3 de Aurora MySQL gestiona las tablas temporales de forma diferente a las versiones anteriores de Aurora MySQL. Este nuevo comportamiento se hereda de MySQL 8.0 Community Edition. Hay dos tipos de tablas temporales que se pueden crear con la versión 3 de Aurora MySQL:

- Tablas temporales internas (o implícitas): las crea el motor de Aurora MySQL para gestionar algunas operaciones, como agregación de ordenación, tablas derivadas o expresiones comunes de tabla (CTE).
- Tablas temporales creadas por el usuario (o explícitas): las crea el motor de Aurora MySQL cuando se utiliza la instrucción `CREATE TEMPORARY TABLE`.

Hay consideraciones adicionales para las tablas temporales, tanto internas como creadas por el usuario, en las instancias de base de datos del lector de Aurora. En las siguientes secciones se analizan estos cambios.

Temas

- Motor de almacenamiento para tablas temporales internas (implícitas)
- Limitación del tamaño de las tablas temporales internas en memoria
- Mitigación de los problemas de llenado de las tablas temporales internas en las réplicas de Aurora
- Optimización del parámetro temptable_max_mmap en instancias de base de datos de Aurora MySQL
- Tablas temporales creadas por el usuario (explícitas) en las instancias de base de datos del lector
- Errores de creación de tablas temporales y su mitigación

Motor de almacenamiento para tablas temporales internas (implícitas)

Al generar conjuntos de resultados intermedios, Aurora MySQL intenta inicialmente escribir en tablas temporales en memoria. Es posible que esto no se realice correctamente debido a tipos de datos incompatibles o a límites configurados. Si es así, la tabla temporal se convierte en una tabla temporal en el disco en lugar de mantenerse en la memoria. Puede encontrar más información al respecto en Internal Temporary Table Use in MySQL (Uso de tablas temporales internas en MySQL) de la documentación de MySQL.

En Aurora MySQL versión 3, la forma en que funcionan las tablas temporales internas es diferente de las versiones anteriores de Aurora MySQL. En lugar de elegir entre los motores de almacenamiento InnoDB y MyISAM para estas tablas temporales, ahora elige entre los motores de almacenamiento `TempTable` y `MEMORY`.

Con el motor de almacenamiento `TempTable`, puede tomar una opción adicional sobre cómo manejar ciertos datos. Los datos afectados desbordan el grupo de memoria que contiene todas las tablas temporales internas de la instancia de base de datos.

Estas opciones pueden influir en el rendimiento de las consultas que generan grandes volúmenes de datos temporales, por ejemplo, al realizar agregaciones tales como `GROUP BY` sobre tablas grandes.

> **Tip**
>
> Si la carga de trabajo incluye consultas que generan tablas temporales internas, confirme cómo funciona su aplicación con este cambio ejecutando puntos de referencia y monitoreando las métricas relacionadas con el rendimiento.
> En algunos casos, la cantidad de datos temporales se ajusta al grupo de memoria `TempTable` o solo desborda el grupo de memoria en una pequeña cantidad. En estos casos, le recomendamos que utilice la configuración `TempTable` para tablas temporales internas y archivos asignados por memoria para contener los datos de desbordamiento. Esta configuración es la predeterminada.

El motor de almacenamiento `TempTable` es el predeterminado. `TempTable` utiliza un grupo de memoria común para todas las tablas temporales que utilizan este motor, en lugar de un límite máximo de memoria por tabla. El tamaño de este grupo de memoria se especifica mediante el parámetro temptable_max_ram. El valor predeterminado es 1 GiB en instancias de base de datos con 16 GiB o más de memoria y 16 MB en instancias de base de datos con menos de 16 GiB de memoria. El tamaño del grupo de memoria influye en el consumo de memoria a nivel de sesión.

En algunos casos, al utilizar el motor de almacenamiento `TempTable`, los datos temporales pueden superar el tamaño del grupo de memoria. Si es así, Aurora MySQL almacena los datos de desbordamiento mediante un mecanismo secundario.

Puede establecer el parámetro temptable_max_mmap para especificar si los datos se desbordan a archivos temporales asignados a memoria o a tablas temporales internas de InnoDB en disco. Los distintos formatos de datos y criterios de desbordamiento de estos mecanismos de desbordamiento pueden afectar al rendimiento de las consultas. Lo hacen al influir en la cantidad de datos escritos en el disco y en la demanda de rendimiento del almacenamiento en disco.

La versión 3 de Aurora MySQL almacena los datos de desbordamiento de la siguiente manera:

- En la instancia de base de datos de escritor, los datos que se desbordan en las tablas temporales internas de InnoDB o en los archivos temporales asignados a la memoria residen en el almacenamiento local de la instancia.
- En las instancias de base de datos de lector, los datos de desbordamiento siempre residen en los archivos temporales asignados a la memoria del almacenamiento local.

  Las instancias de solo lectura no pueden almacenar datos en el volumen del clúster de Aurora.

Los parámetros de configuración relacionados con las tablas temporales internas se aplican de forma diferente a las instancias de escritor y lector del clúster:

- En instancias de lector, Aurora MySQL siempre utiliza el motor de almacenamiento `TempTable`.
- El tamaño de `temptable_max_mmap` tiene un valor predeterminado de 1 GiB, tanto para las instancias de escritor como de lector, independientemente del tamaño de la memoria de la instancia de base de datos. Puede ajustar este valor tanto en instancias de escritor como de lector.
- Al configurar `temptable_max_mmap` en `0`, se desactiva el uso de archivos temporales asignados a memoria en las instancias de escritor.
- No puede establecer `temptable_max_mmap` en `0` en instancias de lector.

> **Note**
>
> No recomendamos utilizar el parámetro temptable_use_mmap. Ha quedado obsoleto y se espera que se elimine en una futura versión de MySQL.

Limitación del tamaño de las tablas temporales internas en memoria

Como se explica en Motor de almacenamiento para tablas temporales internas (implícitas), puede controlar los recursos de la tabla temporal a nivel global mediante las configuraciones temptable_max_ram y temptable_max_mmap.

También puede limitar el tamaño de cualquier tabla temporal interna individual en memoria mediante el parámetro de base de datos tmp_table_size. Este límite tiene por objeto evitar que las consultas individuales consuman una cantidad desmesurada de recursos de las tablas temporales globales, lo que puede afectar al rendimiento de las consultas simultáneas que requieren estos recursos.

El parámetro `tmp_table_size` define el tamaño máximo de las tablas temporales creadas por el motor de almacenamiento `MEMORY` en la versión 3 y versiones posteriores de Aurora MySQL.

En la versión 3.04 y versiones posteriores de Aurora MySQL, `tmp_table_size` también define el tamaño máximo de las tablas temporales creadas por el motor de almacenamiento `TempTable` cuando el parámetro de base de datos `aurora_tmptable_enable_per_table_limit` está establecido en `ON`. Este comportamiento está deshabilitado de forma predeterminada (`OFF`), lo que es el mismo comportamiento que en la versión 3.03 y anteriores de Aurora MySQL.

- Cuándo `aurora_tmptable_enable_per_table_limit` está establecido en `OFF`, `tmp_table_size` no se tiene en cuenta para las tablas temporales internas en memoria creadas por el motor de almacenamiento `TempTable`.

  Sin embargo, se sigue aplicando el límite de recursos global de `TempTable`. Aurora MySQL tiene el siguiente comportamiento cuando se ha alcanzado el límite de recursos global de `TempTable`:

  - Instancias de base de datos del escritor: Aurora MySQL convierte automáticamente la tabla temporal en memoria en una tabla temporal en disco de InnoDB.
  - Instancias de base de datos del lector: la consulta finaliza con un error.

    ```
    ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
    ```

- Cuándo `aurora_tmptable_enable_per_table_limit` está establecido en `ON`, Aurora MySQL tiene el siguiente comportamiento cuando se alcanza el límite de `tmp_table_size`:
  - Instancias de base de datos del escritor: Aurora MySQL convierte automáticamente la tabla temporal en memoria en una tabla temporal en disco de InnoDB.
  - Instancias de base de datos del lector: la consulta finaliza con un error.

```
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
```

En este caso se aplica tanto el límite de recursos global como el límite por tabla de `TempTable`.

> **Note**
>
> El parámetro `aurora_tmptable_enable_per_table_limit` no tiene ningún efecto cuando internal_tmp_mem_storage_engine está configurado en `MEMORY`. En este caso, el tamaño máximo de una tabla temporal en memoria se define mediante el valor tmp_table_size o max_heap_table_size , el que sea menor.

Los siguientes ejemplos muestran el comportamiento del parámetro `aurora_tmptable_enable_per_table_limit` para instancias de base de datos del escritor y lector.

Example Ejemplo de una instancia de base de datos del escritor con **`aurora_tmptable_enable_per_table_limit`** establecido en **`OFF`**

La tabla temporal en memoria no se convierte en una tabla temporal en disco de InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+-----------------+----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+-----------------+----------------------------------------
+---------------------+----------------------+
|                  0 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+-----------------+----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> show status like '%created_tmp_disk%';
```

```
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Created_tmp_disk_tables | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (13.99 sec)

mysql> show status like '%created_tmp_disk%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Created_tmp_disk_tables | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)
```

Example Ejemplo de una instancia de base de datos del escritor con **aurora_tmptable_enable_per_table_limit** establecido en **ON**

La tabla temporal en memoria se convierte en una tabla temporal en disco de InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+------------------+----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+------------------+----------------------------------------
+------------------+
```

```
|                  0 | 3.04.0           |                                                1 |
      16777216 |
+--------------------+------------------+-------------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
+---------+
| max(n)  |
+---------+
| 6000000 |
+---------+
1 row in set (4.10 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 1     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example Ejemplo de una instancia de base de datos del lector con **aurora_tmptable_enable_per_table_limit** establecido en **OFF**

La consulta finaliza sin errores porque no se aplica `tmp_table_size`, y no se ha alcanzado el límite de recursos global de `TempTable`.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+------------------+----------------------------------------
+---------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+------------------+----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (14.05 sec)
```

Example Ejemplo de una instancia de base de datos del lector con **`aurora_tmptable_enable_per_table_limit`** establecido en **`OFF`**

Esta consulta alcanza el límite global de recursos global de TempTable con `aurora_tmptable_enable_per_table_limit` configurado en OFF. La consulta finaliza con un error en las instancias del lector.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
```

```
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
|                   1 | 3.04.0           |                                       0 |
       1073741824 |           1073741824 |
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.01 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 120000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_1586_2' is full
```

Example Ejemplo de una instancia de base de datos del lector con **`aurora_tmptable_enable_per_table_limit`** establecido en **`ON`**

La consulta finaliza con un error cuando se ha alcanzado el límite de `tmp_table_size`.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+------------------+-----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+------------------+-----------------------------------------
+------------------+
|                  1 | 3.04.0           |                                       1 |
      16777216 |
+--------------------+------------------+-----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_8_2' is full
```

Mitigación de los problemas de llenado de las tablas temporales internas en las réplicas de Aurora

Para evitar problemas de limitación del tamaño de las tablas temporales, establezca los parámetros `temptable_max_ram` y `temptable_max_mmap` a un valor combinado que pueda ajustarse a los requisitos de su carga de trabajo.

Tenga cuidado al establecer el valor del parámetro `temptable_max_ram`. Si se establece un valor demasiado alto, se reduce la memoria disponible en la instancia de base de datos, lo que puede provocar una situación de falta de memoria. Supervise el promedio de memoria que se puede liberar en la instancia de base de datos. A continuación, determine un valor adecuado para `temptable_max_ram`, de modo que le quede una cantidad razonable de memoria libre en la instancia. Para obtener más información, consulte Problemas de memoria que se puede liberar en Amazon Aurora.

También es importante supervisar el tamaño del almacenamiento local y el consumo de espacio de las tablas temporales. Puede supervisar el almacenamiento temporal disponible para una instancia de base de datos con la métrica `FreeLocalStorage` de Amazon CloudWatch, que se describe en Métricas de Amazon CloudWatch para Amazon Aurora.

> **Note**
>
> Este procedimiento no funciona cuando el parámetro `aurora_tmptable_enable_per_table_limit` está establecido en `ON`. Para obtener más información, consulte Limitación del tamaño de las tablas temporales internas en memoria.

Example 1

Sabe que sus tablas temporales crecen hasta un tamaño acumulado de 20 GiB. Desea establecer las tablas temporales en memoria a 2 GiB y que crezcan hasta un máximo de 20 GiB en disco.

Establezca `temptable_max_ram` en **`2,147,483,648`** y `temptable_max_mmap` en **`21,474,836,480`**. Estos valores están expresados en bytes.

Esta configuración de parámetros garantiza que las tablas temporales puedan crecer hasta alcanzar un total acumulado de 22 GiB.

Example 2

El tamaño de su instancia actual es 16xlarge o superior. No sabe el tamaño total de las tablas temporales que podría necesitar. Desea poder utilizar hasta 4 GiB en la memoria y hasta el tamaño máximo de almacenamiento disponible en el disco.

Establezca `temptable_max_ram` en **`4,294,967,296`** y `temptable_max_mmap` en **`1,099,511,627,776`**. Estos valores están expresados en bytes.

Aquí se establece `temptable_max_mmap` a 1 TiB, que es menos que el almacenamiento local máximo de 1,2 TiB en una instancia de base de datos 16xlarge de Aurora.

En una instancia de menor tamaño, ajuste el valor de `temptable_max_mmap` para que no llene el almacenamiento local disponible. Por ejemplo, una instancia 2xlarge solo dispone de 160 GiB de almacenamiento local. Por lo tanto, recomendamos establecer el valor a menos de 160 GiB. Para obtener más información sobre el almacenamiento local disponible para los tamaños de instancia de base de datos, consulte Límites de almacenamiento temporal de Aurora MySQL.

Optimización del parámetro temptable_max_mmap en instancias de base de datos de Aurora MySQL

El parámetro `temptable_max_mmap` de Aurora MySQL controla la cantidad máxima de espacio en disco local que pueden utilizar los archivos asignados por memoria antes de desbordarse en las tablas temporales de InnoDB en el disco (en las instancias de base de datos de escritor) o provocar un error (en las instancias de base de datos de lector). Establecer este parámetro de instancia de base de datos correctamente puede ayudar a optimizar el rendimiento de las instancias de base de datos.

Requisitos previos

1. Asegúrese de que el esquema de rendimiento esté habilitado. Para verificarlo, ejecute el siguiente comando SQL:

   ```
   SELECT @@performance_schema;
   ```

   Un valor de salida de 1 indica que está activado.
2. Confirme que la instrumentación de la memoria de tabla temporal esté habilitada. Para verificarlo, ejecute el siguiente comando SQL:

   ```
   SELECT name, enabled FROM performance_schema.setup_instruments WHERE name LIKE
    '%memory%temptable%';
   ```

La columna `enabled` muestra `YES` para las entradas de instrumentación de memoria de la tabla temporal pertinentes.

Supervisión del uso de las tablas temporales

Al establecer el valor inicial para `temptable_max_mmap`, le recomendamos que comience con el 80 % del tamaño de almacenamiento local de la clase de instancia de base de datos que esté utilizando. Esto garantiza que las tablas temporales tengan suficiente espacio en disco para funcionar de manera eficiente y, al mismo tiempo, dejar espacio para otros usos del disco en la instancia.

Para encontrar el tamaño de almacenamiento local de la clase de instancia de base de datos, consulte Límites de almacenamiento temporal de Aurora MySQL.

Por ejemplo, si utiliza la clase de instancia de base de datos db.r5.large, el tamaño del almacenamiento local será de 32 GiB. En este caso, establecería inicialmente el parámetro `temptable_max_mmap` en un 80 % de 32 GiB, lo que equivale a 25,6 GiB.

Tras establecer el valor inicial de `temptable_max_mmap`, ejecute la carga de trabajo máxima en las instancias de Aurora MySQL. Supervise el uso actual y elevado del disco de la tabla temporal mediante la siguiente consulta SQL:

```
SELECT event_name, current_count, current_alloc, current_avg_alloc, high_count,
 high_alloc, high_avg_alloc
FROM sys.memory_global_by_current_bytes WHERE event_name LIKE 'memory/temptable/%';
```

Esta consulta recupera la siguiente información:

- `event_name`: el nombre del evento de uso de disco o de memoria de la tabla temporal.
- `current_count`: el número actual de bloques de disco o de memoria de tabla temporal asignados.
- `current_alloc`: la cantidad actual de memoria o disco asignada a las tablas temporales.
- `current_avg_alloc`: el tamaño medio actual de bloques de disco o de memoria de tabla temporal.
- `high_count`: el número más alto de bloques de disco o de memoria de tabla temporal asignados.
- `high_alloc`: la cantidad mayor de memoria o disco asignada a las tablas temporales.
- `high_avg_alloc`: el tamaño medio más alto de bloques de disco o de memoria de tabla temporal.

Si las consultas producen el error La tabla está llena al utilizar esta configuración, esto indica que la carga de trabajo requiere más espacio en disco para las operaciones temporales de la tabla. En este caso, considere la posibilidad de aumentar el tamaño de la instancia de base de datos a una con más espacio de almacenamiento local.

Establecimiento del valor **`temptable_max_mmap`** óptimo

Utilice el siguiente procedimiento para supervisar y establecer el tamaño correcto para el parámetro `temptable_max_mmap`.

1. Revise el resultado de la consulta anterior e identifique el pico de uso temporal del disco de la tabla, como se indica en la columna `high_alloc`.
2. En función del uso máximo temporal del disco de la tabla, ajuste el parámetro `temptable_max_mmap` en el grupo de parámetros de base de datos para las instancias de base de datos de Aurora MySQL.

   Establezca el valor para que sea ligeramente superior al máximo de uso temporal del disco de la tabla para adaptarse al crecimiento futuro.
3. Aplique los cambios del grupo de parámetros a las instancias de base de datos.
4. Vuelva a supervisar el uso del disco de tabla temporal durante los picos de carga de trabajo para asegurarse de que el nuevo valor de `temptable_max_mmap` es el adecuado.
5. Repita los pasos anteriores según sea necesario para afinar el parámetro `temptable_max_mmap`.

Tablas temporales creadas por el usuario (explícitas) en las instancias de base de datos del lector

Puede crear tablas temporales explícitas mediante la palabra clave `TEMPORARY` en la instrucción `CREATE TABLE`. Las tablas temporales explícitas se admiten en la instancia de base de datos del escritor en un clúster de base de datos de Aurora. También puede utilizar tablas temporales explícitas en instancias de base de datos del lector, pero las tablas no pueden imponer el uso del motor de almacenamiento InnoDB.

Para evitar errores al crear tablas temporales explícitas en las instancias de la base de datos de escritor de Aurora MySQL, asegúrese de que ejecuta todas las instrucciones `CREATE TEMPORARY TABLE` de una o ambas de las siguientes maneras:

- No especifique la cláusula `ENGINE=InnoDB`.
- No establezca el modo SQL en `NO_ENGINE_SUBSTITUTION`.

Errores de creación de tablas temporales y su mitigación

El error que recibe difiere en función de si utiliza una simple instrucción `CREATE TEMPORARY TABLE` o variante `CREATE TEMPORARY TABLE AS SELECT`. Los siguientes ejemplos muestran los diferentes tipos de errores.

Este comportamiento de tabla temporal solo se aplica a instancias de solo lectura. Este primer ejemplo confirma que ese es el tipo de instancia a la que está conectada la sesión.

```
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
```

Para instrucciones `CREATE TEMPORARY TABLE` simples, la instrucción falla cuando el modo SQL `NO_ENGINE_SUBSTITUTION` está activado. Cuando `NO_ENGINE_SUBSTITUTION` está desactivado (de forma predeterminada), se realiza la sustitución del motor adecuado y la creación de la tabla temporal se lleva a cabo correctamente.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql>  CREATE TEMPORARY TABLE tt2 (id int) ENGINE=InnoDB;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> CREATE TEMPORARY TABLE tt4 (id int) ENGINE=InnoDB;

mysql> SHOW CREATE TABLE tt4\G
*************************** 1. row ***************************
       Table: tt4
Create Table: CREATE TEMPORARY TABLE `tt4` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Para instrucciones `CREATE TEMPORARY TABLE AS SELECT`, la instrucción falla cuando el modo SQL `NO_ENGINE_SUBSTITUTION` está activado. Cuando `NO_ENGINE_SUBSTITUTION` está desactivado (de forma predeterminada), se realiza la sustitución del motor adecuado y la creación de la tabla temporal se lleva a cabo correctamente.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql> CREATE TEMPORARY TABLE tt1 ENGINE=InnoDB AS SELECT * FROM t1;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> show create table tt3;
+-------+-----------------------------------------------------------------+
| Table | Create Table                                                    |
+-------+-----------------------------------------------------------------+
| tt3   | CREATE TEMPORARY TABLE `tt3` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

Para obtener más información sobre los aspectos de almacenamiento y las implicaciones de rendimiento de las tablas temporales en Aurora MySQL versión 3, consulte la entrada del blog Use the TempTable storage engine on Amazon RDS for MySQL and Amazon Aurora MySQL.

Comparación entre Aurora MySQL versión 2 y Aurora MySQL versión 3

Utilice lo siguiente para obtener información sobre los cambios que debe tener en cuenta al actualizar el clúster de Aurora MySQL versión 2 a la versión 3.

Temas

- Compatibilidad con el lenguaje de definición de datos (DDL) atómicos
- Diferencias de características entre las versiones 2 y 3 de Aurora MySQL
- Compatibilidad con clases de instancia
- Cambios de parámetros para Aurora MySQL versión 3
- Variables de estado
- Cambios de idioma inclusivos para Aurora MySQL versión 3
- Valores de AUTO_INCREMENT
- Reproducción de registros binarios

Compatibilidad con el lenguaje de definición de datos (DDL) atómicos

Uno de los cambios más importantes de MySQL 5.7 a 8.0 es la introducción del Atomic Data Dictionary. En las versiones anteriores a MySQL 8.0, el diccionario de datos de MySQL utilizaba un enfoque basado en archivos para almacenar metadatos, tales como definiciones de tablas (.frm), desencadenadores (.trg) y funciones por separado de los metadatos del motor de almacenamiento (como los de InnoDB). Esto tenía algunos problemas, como el riesgo de que las tablas quedaran huérfanas si se producía algo inesperado durante una operación de DDL, lo que provocaba que los metadatos basados en archivos y los del motor de almacenamiento no estuvieran sincronizados.

Para solucionar este problema, MySQL 8.0 ha introducido el Atomic Data Dictionary, que almacena todos los metadatos en un conjunto de tablas internas de InnoDB en el esquema de `mysql`. Esta nueva arquitectura proporciona una forma transaccional y compatible con ACID para administrar los metadatos de las bases de datos, lo que resuelve el problema de DDL atómico que planteaba el antiguo enfoque basado en archivos. Para obtener más información sobre el Atomic Data Dictionary, consulte las secciones Removal of File-based Metadata Storage y Atomic Data Definition Statement Support del MySQL Reference Manual.

Debido a este cambio de diseño, debe plantearse lo siguiente cuando tenga que actualizar de la versión 2 a la versión 3 de Aurora MySQL:

- Los metadatos basados en archivos de la versión 2 se deben migrar a las nuevas tablas del diccionario de datos durante el proceso de actualización a la versión 3. Según el número de objetos de la base de datos que se migren, este proceso puede tardar algún tiempo.
- Los cambios también han introducido algunas incompatibilidades nuevas que puede que tengan que abordarse antes de llevar a cabo la actualización de MySQL 5.7 a 8.0. Por ejemplo, la versión 8.0 tiene algunas palabras clave reservadas nuevas que podrían entrar en conflicto con los nombres de objetos existentes de la base de datos.

Para ayudarle a identificar estas incompatibilidades antes de actualizar el motor, Aurora MySQL ejecuta una serie de comprobaciones de compatibilidad de actualizaciones (comprobaciones previas) para determinar si hay algún objeto incompatible en el diccionario de la base de datos antes de actualizar el diccionario de datos. Para obtener más información sobre las comprobaciones previas, consulte Comprobaciones previas de actualización de versiones principales para Aurora MySQL.

Diferencias de características entre las versiones 2 y 3 de Aurora MySQL

Las siguientes características de Amazon Aurora MySQL son compatibles en Aurora MySQL para MySQL 5.7, pero dichas características no se admiten actualmente en Aurora MySQL para MySQL 8.0:

- No puede usar Aurora MySQL versión 3 para clústeres de Aurora Serverless v1. Aurora MySQL versión 3 funciona con Aurora Serverless v2.
- El modo lab no se aplica a Aurora MySQL versión 3. No hay funciones de modo lab en Aurora MySQL versión 3. DDL instantáneo sustituye a la característica rápida DDL en línea que antes estaba disponible en modo lab. Para ver un ejemplo, consulte DDL instantáneo (Aurora MySQL versión 3).
- La caché de consultas se elimina de la comunidad MySQL 8.0 y también de Aurora MySQL versión 3.
- Aurora MySQL versión 3 es compatible con la característica de unión hash de la comunidad MySQL. No se utiliza la implementación específica de Aurora de uniones hash en Aurora MySQL versión 2. Para obtener información sobre el uso de combinaciones hash con una consulta paralela de Aurora, consulte Activación de una combinación hash para clústeres de consultas paralelas y Sugerencias de Aurora MySQL. Para obtener información general de uso sobre las uniones hash, consulte Optimización de combinaciones hash en el Manual de referencia de MySQL.
- El procedimiento almacenado `mysql.lambda_async` que quedó obsoleto en Aurora MySQL versión 2 se elimina en la versión 3. Para la versión 3, utilice la función asíncrona `lambda_async` en su lugar.
- El conjunto de caracteres predeterminado en Aurora MySQL versión 3 es `utf8mb4`. En Aurora MySQL versión 2, el conjunto de caracteres predeterminado era `latin1`. Para obtener información sobre este conjunto de caracteres, consulte El conjunto de caracteres utf8mb4 (codificación Unicode UTF-8 de 4 bytes) en el Manual de referencia de MySQL.

Algunas funciones de Aurora MySQL están disponibles para ciertas combinaciones de versión del motor de base de datos y región AWS. Para obtener más información, consulte Funciones admitidas en Amazon Aurora por Región de AWS y el motor de base de datos de Aurora.

Compatibilidad con clases de instancia

La versión 3 de Aurora MySQL admite un conjunto de clases de instancia diferente al de la versión 2 de Aurora MySQL:

- Para instancias más grandes, puede utilizar las clases de instancias modernas, como `db.r5`, `db.r6g`, y `db.x2g`.
- Para instancias más pequeñas, puede utilizar las clases de instancias modernas, como `db.t3` y `db.t4g`.

> **Note**
>
> Recomendamos que se usen solo las clases de instancia de base de datos T para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más información sobre las clases de instancia T, consulte Utilización de clases de instancia T para el desarrollo y la prueba.

Las siguientes clases de instancia de Aurora MySQL versión 2 no están disponibles para Aurora MySQL versión 3:

- `db.r4`
- `db.r3`
- `db.t3.small`
- `db.t2`

Verifique los scripts de administración en busca de instrucciones CLI que crean instancias de base de datos de Aurora MySQL. Nombres de clase de instancia codificada que no están disponibles para Aurora MySQL, versión 3. Si es necesario, modifique los nombres de las clases de instancia a los que admite Aurora MySQL versión 3.

> **Tip**
>
> Para verificar las clases de instancias que puede utilizar para una combinación específica de la versión de Aurora MySQL y la región AWS, utilice el comando `describe-orderable-db-instance-options` AWS CLI.

Para obtener más detalles acerca de las clases de instancia Aurora, consulte Clases de instancia de base de datos de Amazon Aurora.

Cambios de parámetros para Aurora MySQL versión 3

Aurora MySQL versión 3 incluye nuevos parámetros de configuración a nivel de clúster y de instancia. Aurora MySQL versión 3 también elimina algunos parámetros que estaban presentes en Aurora MySQL versión 2. Algunos nombres de parámetros se modifican como resultado de la iniciativa de un lenguaje inclusivo. Para obtener compatibilidad con versiones anteriores, puede recuperar los valores de los parámetros utilizando los nombres antiguos o los nombres nuevos. Sin embargo, debe utilizar los nombres nuevos para especificar valores de parámetro en un grupo de parámetros personalizado.

En Aurora MySQL versión 3, el valor del parámetro `lower_case_table_names` se establece permanentemente en el momento en que se crea el clúster. Si utiliza un valor no predeterminado para esta opción, configure el grupo de parámetros personalizados Aurora MySQL versión 3 antes de actualizar. A continuación, especifique el grupo de parámetros durante la operación de creación de clúster o restauración de instantáneas.

> **Note**
>
> Con una base de datos global de Aurora basada en Aurora MySQL, no se puede realizar una actualización local desde la versión 2 a la versión 3 de Aurora MySQL si el parámetro `lower_case_table_names` está activado. Utilice el método de restauración de instantáneas en su lugar.

En la versión 3 de Aurora MySQL, los parámetros `init_connect` y `read_only` no se aplican a los usuarios que tienen el privilegio `CONNECTION_ADMIN`. Esto incluye al usuario maestro de Aurora. Para obtener más información, consulte Modelo de privilegios basado en roles.

Para obtener una lista de todos los parámetros de clúster de Aurora MySQL, consulte Parámetros de nivel de clúster. La tabla cubre todos los parámetros de las versiones 2 y 3 de Aurora MySQL. La tabla incluye notas que muestran qué parámetros son nuevos en Aurora MySQL versión 3 o se eliminaron de Aurora MySQL versión 3.

Para obtener la lista completa de los parámetros de instancia de Aurora MySQL, consulte Parámetros de nivel de instancia. La tabla cubre todos los parámetros de las versiones 2 y 3 de Aurora MySQL. La tabla incluye notas que muestran qué parámetros son nuevos en Aurora MySQL versión 3 y qué parámetros se eliminaron de Aurora MySQL versión 3. También incluye notas que muestran qué parámetros eran modificables en versiones anteriores, pero no en Aurora MySQL versión 3.

Para obtener información sobre los nombres de parámetros que han cambiado, consulte Cambios de idioma inclusivos para Aurora MySQL versión 3.

Variables de estado

Para obtener información sobre las variables de estado que no se aplican a Aurora MySQL, consulte Variables de estado de MySQL que no se aplican a Aurora MySQL.

Cambios de idioma inclusivos para Aurora MySQL versión 3

La versión inicial de Aurora MySQL versión 3 es compatible con la versión 8.0.23 de la edición de la comunidad MySQL. Aurora MySQL versión 3 también incluye cambios de MySQL 8.0.26 relacionados con palabras clave y esquemas del sistema para un lenguaje inclusivo. Por ejemplo, ahora se prefiere el comando `SHOW REPLICA STATUS` en lugar de `SHOW SLAVE STATUS`.

Las siguientes métricas de Amazon CloudWatch tienen nuevos nombres en Aurora MySQL versión 3.

En Aurora MySQL versión 3, solo están disponibles los nuevos nombres de métricas. Asegúrese de actualizar las alarmas u otra automatización que se basa en nombres de métricas cuando actualice a Aurora MySQL versión 3.

Nombre antiguo	Nombre nuevo
`ForwardingMasterDMLLatency`	`ForwardingWriterDMLLatency`
`ForwardingMasterOpenSessions`	`ForwardingWriterOpenSessions`
`AuroraDMLRejectedMasterFull`	`AuroraDMLRejectedWriterFull`
`ForwardingMasterDMLThroughput`	`ForwardingWriterDMLThroughput`

Las siguientes variables de estado tienen nombres nuevos en Aurora MySQL versión 3.

Para obtener compatibilidad, puede utilizar cualquiera de los dos nombres en la versión inicial de Aurora MySQL versión 3. Los nombres de variables de estado anteriores se eliminarán en una próxima versión.

Nombre que se va a eliminar	Nombre nuevo o preferido
`Aurora_fwd_master_dml_stmt_duration`	`Aurora_fwd_writer_dml_stmt_duration`
`Aurora_fwd_master_dml_stmt_count`	`Aurora_fwd_writer_dml_stmt_count`
`Aurora_fwd_master_select_stmt_duration`	`Aurora_fwd_writer_select_stmt_duration`
`Aurora_fwd_master_select_stmt_count`	`Aurora_fwd_writer_select_stmt_count`
`Aurora_fwd_master_errors_session_timeout`	`Aurora_fwd_writer_errors_session_timeout`
`Aurora_fwd_master_open_sessions`	`Aurora_fwd_writer_open_sessions`
`Aurora_fwd_master_errors_session_limit`	`Aurora_fwd_writer_errors_session_limit`
`Aurora_fwd_master_errors_rpc_timeout`	`Aurora_fwd_writer_errors_rpc_timeout`

Los siguientes parámetros de configuración tienen nombres nuevos en Aurora MySQL versión 3.

Para obtener compatibilidad, puede verificar los valores de los parámetros en el cliente `mysql` mediante cualquiera de los dos nombres de la versión inicial de Aurora MySQL versión 3. Solo podrá

utilizar los nuevos nombres cuando modifique los valores de un grupo de parámetros personalizado. Los nombres de los parámetros anteriores se eliminarán en una próxima versión.

Nombre que se va a eliminar	Nombre nuevo o preferido
aurora_fwd_master_idle_timeout	aurora_fwd_writer_idle_timeout
aurora_fwd_master_max_connections_pct	aurora_fwd_writer_max_connections_pct
master_verify_checksum	source_verify_checksum
sync_master_info	sync_source_info
init_slave	init_replica
rpl_stop_slave_timeout	rpl_stop_replica_timeout
log_slow_slave_statements	log_slow_replica_statements
slave_max_allowed_packet	replica_max_allowed_packet
slave_compressed_protocol	replica_compressed_protocol
slave_exec_mode	replica_exec_mode
slave_type_conversions	replica_type_conversions
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_parallel_type	replica_parallel_type

Nombre que se va a eliminar	Nombre nuevo o preferido	
slave_preserve_commit_order	replica_preserve_commit_order	
log_slave_updates	log_replica_updates	
slave_allow_batching	replica_allow_batching	
slave_load_tmpdir	replica_load_tmpdir	
slave_net_timeout	replica_net_timeout	
sql_slave_skip_counter	sql_replica_skip_counter	
slave_skip_errors	replica_skip_errors	
slave_checkpoint_period	replica_checkpoint_period	
slave_checkpoint_group	replica_checkpoint_group	
slave_transaction_retries	replica_transaction_retries	
slave_parallel_workers	replica_parallel_workers	
slave_pending_jobs_size_max	replica_pending_jobs_size_max	
pseudo_slave_mode	pseudo_replica_mode	

Los siguientes procedimientos almacenados tienen nombres nuevos en Aurora MySQL versión 3.

Para obtener compatibilidad, puede utilizar cualquiera de los dos nombres en la versión inicial de Aurora MySQL versión 3. Los nombres de procedimientos antiguos se eliminarán en una próxima versión.

Nombre que se va a eliminar	Nombre nuevo o preferido	
`mysql.rds_set_master_auto_position`	`mysql.rds_set_source_auto_position`	
`mysql.rds_set_external_master`	`mysql.rds_set_external_source`	
`mysql.rds_set_external_master_with_auto_position`	`mysql.rds_set_external_source_with_auto_position`	
`mysql.rds_reset_external_master`	`mysql.rds_reset_external_source`	
`mysql.rds_next_master_log`	`mysql.rds_next_source_log`	

Valores de AUTO_INCREMENT

En Aurora MySQL versión 3, Aurora conserva el valor `AUTO_INCREMENT` para cada tabla cuando reinicia cada instancia de base de datos. En Aurora MySQL versión 2, no se ha conservado el valor `AUTO_INCREMENT` después de un reinicio.

El valor `AUTO_INCREMENT` no se conserva cuando configura un nuevo clúster al restaurar desde una instantánea, realizar una recuperación a un momento dado y clonar un clúster. En estos casos, el valor `AUTO_INCREMENT` se inicializa en el valor basándose en el valor de columna más grande de la tabla en el momento de crear la instantánea. Este comportamiento es diferente al de RDS for MySQL 8.0, donde el valor `AUTO_INCREMENT` se conserva durante estas operaciones.

Reproducción de registros binarios

En la edición de la comunidad MySQL 8.0, la replicación de registros binarios está activada de forma predeterminada. En Aurora MySQL versión 3, la replicación de registros binarios está desactivada de forma predeterminada.

> **Tip**
>
> Si las funciones de replicación incorporadas de Aurora cumplen sus requisitos de alta disponibilidad, puede dejar desactivada la replicación de registros binarios. De esta forma, puede evitar la sobrecarga de rendimiento de la replicación de registros binarios. También puede evitar el monitoreo y la solución de problemas asociados que se necesitan para administrar la replicación de registros binarios.

Aurora admite la replicación de registros binarios desde un origen compatible con MySQL 5.7 a Aurora MySQL versión 3. El sistema de origen puede ser un clúster de bases de datos de Aurora MySQL, una instancia de base de datos de RDS for MySQL o una instancia de MySQL local.

Al igual que la comunidad MySQL, Aurora MySQL admite la replicación desde un origen que ejecuta una versión específica en un destino que ejecuta la misma versión principal o una versión principal superior. Por ejemplo, no se admite la replicación desde un sistema compatible con MySQL 5.6 a Aurora MySQL versión 3. No se admite la replicación de Aurora MySQL versión 3 a un sistema compatible con MySQL 5.7 o compatible con MySQL 5.6. Para obtener más detalles acerca de cómo utilizar replicación de registros binarios, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Aurora MySQL versión 3 incluye mejoras en la replicación de registros binarios en la comunidad MySQL 8.0, como la replicación filtrada. Para obtener más información sobre las mejoras de MySQL 8.0 de la comunidad, consulte Cómo evalúan los servidores las reglas de filtrado de replicación en el Manual de referencia de MySQL.

Compresión de transacciones para replicación de registros binarios

Para obtener información de uso sobre la compresión de registros binarios, consulte Compresión de transacciones de registros binarios en el Manual de referencia de MySQL.

Las siguientes limitaciones se aplican a la compresión de registros binarios en Aurora MySQL versión 3:

- Las transacciones cuyos datos de registro binario superan el tamaño máximo permitido del paquete no se comprimen. Esto se aplica independientemente de si la configuración de compresión de registro binario de Aurora MySQL está activada. Estas transacciones se replican sin comprimirse.

- Si utiliza un conector para la captura de datos de cambios (CDC) que aún no admite MySQL 8.0, no puede utilizar esta característica. Le recomendamos que pruebe exhaustivamente cualquier conector de terceros con compresión de registro binario. Le recomendamos que lo haga antes de activar la compresión binlog en sistemas que utilizan replicación binlog para CDC.

Comparación de Aurora MySQL versión 3 y MySQL 8.0 Community Edition

Puede utilizar la siguiente información para obtener información sobre los cambios que debe tener en cuenta al convertir de otro sistema compatible con MySQL 8.0 a Aurora MySQL versión 3.

En general, Aurora MySQL versión 3 admite el conjunto de características de la comunidad MySQL 8.0.23. Algunas características nuevas de la edición de la comunidad MySQL 8.0 no se aplican a Aurora MySQL. Algunas de esas funciones no son compatibles con algún aspecto de Aurora, como la arquitectura de almacenamiento Aurora. No se necesitan otras funciones porque el servicio de administración de Amazon RDS proporciona una funcionalidad equivalente. Las siguientes características de la comunidad MySQL 8.0 no son compatibles o funcionan de forma diferente en Aurora MySQL versión 3.

Para ver las notas de todas las versiones de Aurora MySQL versión 3, consulte el tema sobre actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 3 en las notas de la versión de Aurora MySQL.

Temas

- Las funciones de MySQL 8.0 no están disponibles en Aurora MySQL versión 3
- Modelo de privilegios basado en roles
- Búsqueda del ID de servidor de base de datos
- Autenticación

Las funciones de MySQL 8.0 no están disponibles en Aurora MySQL versión 3

Las siguientes características de la comunidad MySQL 8.0 no son compatibles o funcionan de forma diferente en Aurora MySQL versión 3.

- Los grupos de recursos y las instrucciones SQL asociadas no son compatibles con Aurora MySQL.
- Aurora MySQL no admite los espacios de tabla de deshacer definidos por el usuario ni las instrucciones SQL asociadas, como `CREATE UNDO TABLESPACE`, `ALTER UNDO TABLESPACE ... SET INACTIVE` y `DROP UNDO TABLESPACE`.

- Aurora MySQL no admite deshacer el truncado de espacios de tablas de deshacer en las versiones de Aurora MySQL anteriores a la 3.06. En la versión 3.06 y posteriores de Aurora MySQL, se admite el truncado automático de espacios de tablas de deshacer.
- Se admite el complemento de validación de contraseñas.
- No puede modificar la configuración de ningún complemento de MySQL, incluido el complemento de validación de contraseñas.
- El complemento X no es compatible.
- No se admite la replicación automática.

Modelo de privilegios basado en roles

Con Aurora MySQL versión 3, no se pueden modificar las tablas de base de datos de `mysql` directamente. En particular, no se pueden configurar usuarios insertándolos en la tabla de `mysql.user`. En su lugar, se utilizan sentencias SQL para otorgar privilegios basados en roles. Tampoco puede crear otros tipos de objetos, como procedimientos almacenados en la base de datos `mysql`. Aún puede consultar las tablas de `mysql`. Si utiliza la replicación de registros binarios, los cambios realizados directamente en la tabla de `mysql` del clúster de origen no se replican en el clúster destino.

En algunos casos, la aplicación puede utilizar accesos directos para crear usuarios u otros objetos insertándolos en las tablas de `mysql`. Si es así, cambie el código de la aplicación para utilizar las declaraciones correspondientes, como `CREATE USER`. Si la aplicación crea procedimientos almacenados u otros objetos en la base de datos `mysql`, utilice otra base de datos en su lugar.

Para exportar metadatos para usuarios de bases de datos durante la migración desde una base de datos MySQL externa, puede utilizar el comando del intérprete de comandos de MySQL en lugar de `mysqldump`. Para obtener más información, consulte Instance Dump Utility, Schema Dump Utility, and Table Dump Utility.

Para simplificar la administración de permisos para muchos usuarios o aplicaciones, puede utilizar la instrucción `CREATE ROLE` para crear un rol que tenga un conjunto de permisos. A continuación, puede utilizar las instrucciones `GRANT` y `SET ROLE` y la función `current_role` para asignar roles a usuarios o aplicaciones, cambiar el rol actual y verificar qué roles están en vigor. Para obtener más información sobre el sistema de permisos basado en roles de MySQL 8.0, consulte Uso de roles en el Manual de referencia de MySQL.

> **Important**
>
> Le recomendamos encarecidamente que no utilice el usuario maestro directamente en sus aplicaciones. En lugar de ello, es mejor ceñirse a la práctica recomendada de utilizar un usuario de base de datos creado con los privilegios mínimos necesarios para su aplicación.

Temas

- rds_superuser_role
- Usuario de comprobaciones de privilegios para replicación de registros binarios
- Roles para acceder a otros servicios de AWS

rds_superuser_role

Aurora MySQL versión 3 incluye un rol especial que tiene todos los siguientes privilegios. El rol se denomina `rds_superuser_role`. El usuario administrativo principal de cada clúster ya tiene asignada este rol. El rol `rds_superuser_role` incluye los siguientes privilegios para todos los objetos de base de datos:

- `ALTER`
- `APPLICATION_PASSWORD_ADMIN`
- `ALTER ROUTINE`
- `CONNECTION_ADMIN`
- `CREATE`
- `CREATE ROLE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`
- `DELETE`
- `DROP`
- `DROP ROLE`
- `EVENT`

- EXECUTE
- FLUSH_OPTIMIZER_COSTS (versión 3.09 y posteriores de Aurora MySQL)
- FLUSH_STATUS (versión 3.09 y posteriores de Aurora MySQL)
- FLUSH_TABLES (versión 3.09 y posteriores de Aurora MySQL)
- FLUSH_USER_RESOURCES (versión 3.09 y posteriores de Aurora MySQL)
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- ROLE_ADMIN
- SET_USER_ID
- SELECT
- SHOW DATABASES
- SHOW_ROUTINE (versión 3.04 y posteriores de Aurora MySQL)
- SHOW VIEW
- TRIGGER
- UPDATE
- XA_RECOVER_ADMIN

La definición de rol también incluye WITH GRANT OPTION para que un usuario administrativo pueda conceder ese rol a otros usuarios. En particular, el administrador debe conceder los privilegios necesarios para realizar la replicación de registros binarios con el clúster Aurora MySQL como destino.

> **Tip**
>
> Para ver todos los detalles de los permisos, ingrese las siguientes instrucciones.

```
SHOW GRANTS FOR rds_superuser_role@'%';
SHOW GRANTS FOR name_of_administrative_user_for_your_cluster@'%';
```

Usuario de comprobaciones de privilegios para replicación de registros binarios

La versión 3 de Aurora MySQL incluye un usuario de comprobaciones de privilegios para la replicación de registros binarios (binlog), `rdsrepladmin_priv_checks_user`. Además de los privilegios de `rds_superuser_role`, este usuario tiene el privilegio `replication_applier`.

Cuando se activa la replicación de binlog mediante una llamada al procedimiento almacenado `mysql.rds_start_replication`, se crea `rdsrepladmin_priv_checks_user`.

El usuario `rdsrepladmin_priv_checks_user@localhost` es un usuario reservado. No lo modifique.

Roles para acceder a otros servicios de AWS

Aurora MySQL versión 3 también incluye roles que puede utilizar para acceder a otros servicios de AWS. Puede establecer muchos de estos roles como alternativa a la concesión de privilegios. Por ejemplo, especifica `GRANT AWS_LAMBDA_ACCESS TO user` en lugar de `GRANT INVOKE LAMBDA ON *.* TO user`. Para ver los procedimientos de acceso a otros servicios de AWS, consulte Integración de Amazon Aurora MySQL con otros servicios de AWS. Aurora MySQL versión 3 incluye los siguientes roles relacionados con el acceso a otros servicios de:AWS

- `AWS_LAMBDA_ACCESS`: una alternativa al privilegio `INVOKE LAMBDA`. Para obtener más información, consulte Invocación de una función de Lambda desde un clúster de base de datos de Amazon Aurora MySQL.
- `AWS_LOAD_S3_ACCESS`: una alternativa al privilegio `LOAD FROM S3`. Para obtener más información, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.
- `AWS_SELECT_S3_ACCESS`: una alternativa al privilegio `SELECT INTO S3`. Para obtener más información, consulte Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3.
- `AWS_COMPREHEND_ACCESS`: una alternativa al privilegio `INVOKE COMPREHEND`. Para obtener más información, consulte Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora.

- `AWS_SAGEMAKER_ACCESS`: una alternativa al privilegio `INVOKE SAGEMAKER`. Para obtener más información, consulte Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora.
- `AWS_BEDROCK_ACCESS`: no hay un privilegio análogo a `INVOKE` para Amazon Bedrock. Para obtener más información, consulte Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora.

Cuando concede acceso mediante roles en Aurora MySQL versión 3, también activa el rol mediante la instrucción `SET ROLE` *`role_name`* o `SET ROLE ALL`. El siguiente ejemplo muestra cómo. Sustituya el nombre de rol apropiado por `AWS_SELECT_S3_ACCESS`.

```
# Grant role to user.

mysql> GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'

# Check the current roles for your user. In this case, the AWS_SELECT_S3_ACCESS role
 has not been activated.
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-----------------------------------------------------+
| CURRENT_ROLE()                                      |
+-----------------------------------------------------+
| `AWS_SELECT_S3_ACCESS`@`%`,`rds_superuser_role`@`%` |
+-----------------------------------------------------+
```

Búsqueda del ID de servidor de base de datos

El ID de servidor de base de datos (`server_id`) es necesario para la replicación de registro binario (binlog). La búsqueda del ID del servidor en Aurora MySQL se realiza de forma diferente que en Community MySQL.

En Community MySQL, el ID del servidor es un número que se obtiene mediante la siguiente sintaxis al registrarse en el servidor:

```
mysql> select @@server_id;

+-------------+
| @@server_id |
+-------------+
| 2           |
+-------------+
1 row in set (0.00 sec)
```

En Aurora MySQL, el ID del servidor es el ID de la instancia de base de datos, que se obtiene mediante la siguiente sintaxis al registrarse en la instancia de base de datos:

```
mysql> select @@aurora_server_id;

+------------------------+
| @@aurora_server_id     |
+------------------------+
| mydbcluster-instance-2 |
+------------------------+
1 row in set (0.00 sec)
```

Para obtener más información sobre la replicación binlog, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Autenticación

En la comunidad MySQL 8.0, el complemento de autenticación predeterminado es `caching_sha2_password`. Aurora MySQL versión 3 sigue utilizando el complemento `mysql_native_password`. No se puede cambiar la configuración de `default_authentication_plugin`. Sin embargo, puede crear nuevos usuarios y modificar los usuarios actuales, y las contraseñas individuales utilizarán el nuevo complemento de autenticación. A continuación se muestra un ejemplo.

```
mysql> CREATE USER 'testnewsha'@'%' IDENTIFIED WITH caching_sha2_password BY
 'aNewShaPassword';
Query OK, 0 rows affected (0.74 sec)
```

Actualización a Aurora MySQL versión 3

Para obtener información sobre cómo actualizar su base de datos de la versión 2 a la versión 3 de Aurora MySQL, consulte Actualización de la versión principal de un clúster de base de datos de Amazon Aurora MySQL.

Aurora MySQL versión 2 compatible con MySQL 5.7

En este tema, se describen las diferencias entre la versión 2 de Aurora MySQL y MySQL 5.7 Community Edition.

> **Important**
>
> La versión 2 de Aurora MySQL ya no recibe soporte estándar desde el 31 de octubre de 2024. Para obtener más información, consulte Preparación para el final del soporte estándar de la versión 2 de Amazon Aurora MySQL-Compatible Edition.

Características no compatibles con Aurora MySQL versión 2

Las siguientes características se admiten en MySQL 5.7, pero no se admiten actualmente en Aurora MySQL versión 2:

- `CREATE TABLESPACE` Instrucción SQL
- Complemento de replicación de grupo
- Tamaño de página incrementado
- Carga de grupo de búfer de InnoDB al inicio
- Complemento de analizador de texto completo de InnoDB
- Replicación de varios orígenes
- Cambio de tamaño de grupo de búfer online
- Complemento de validación de contraseñas: puede instalar el complemento, pero no es compatible. No se puede personalizar el complemento.
- Complementos de reescritura de consulta

- Filtrado de replicación
- Protocolo X

Para obtener más información sobre estas características, consulte la documentación de MySQL 5.7.

Comportamiento de los espacios de tabla temporales en Aurora MySQL versión 2

En MySQL 5.7, el espacio de tablas temporal se amplía automáticamente y aumenta de tamaño según sea necesario para dar cabida a las tablas temporales en el disco. Cuando se eliminan tablas temporales, el espacio libre se puede reutilizar para nuevas tablas temporales, pero el espacio de tablas temporal sigue teniendo el tamaño ampliado y no se reduce. El espacio de tablas temporal se elimina y se vuelve a crear cuando se reinicia el motor.

En Aurora MySQL versión 2, se aplica el siguiente comportamiento:

- En el caso de nuevos clústeres de base de datos de Aurora MySQL creados con la versión 2.10 o posteriores, el espacio de tablas temporal se elimina y se vuelve a crear al reiniciar la base de datos. Esto permite que la función de cambio de tamaño dinámico recupere el espacio de almacenamiento.
- Para clústeres de base de datos de Aurora MySQL existentes actualizados a las siguientes versiones:
  - Versión 2.10 o posteriores: el espacio de tablas temporal se elimina y se vuelve a crear al reiniciar la base de datos. Esto permite que la función de cambio de tamaño dinámico recupere el espacio de almacenamiento.
  - Versión 2.09: el espacio de tablas temporal no se elimina al reiniciar la base de datos.

Puede comprobar el tamaño del espacio de tablas temporal de su clúster de base de datos de Aurora MySQL versión 2 mediante la siguiente consulta:

```
SELECT
    FILE_NAME,
    TABLESPACE_NAME,
    ROUND((TOTAL_EXTENTS * EXTENT_SIZE) / 1024 / 1024 / 1024, 4) AS SIZE
FROM
    INFORMATION_SCHEMA.FILES
WHERE
    TABLESPACE_NAME = 'innodb_temporary';
```

Para obtener más información, consulte The Temporary Tablespace (El espacio de tablas temporal) en la documentación de MySQL.

Motor de almacenamiento para tablas temporales en disco

La versión 2 de Aurora MySQL usa diferentes motores de almacenamiento para las tablas temporales internas en disco, según el rol de la instancia.

- En la instancia de escritor, las tablas temporales en disco utilizan el motor de almacenamiento InnoDB de forma predeterminada. Están almacenadas en el espacio de tablas temporal del volumen del clúster de Aurora.

  Puede cambiar este comportamiento en la instancia de escritor modificando el valor del parámetro de base de datos `internal_tmp_disk_storage_engine`. Para obtener más información, consulte Parámetros de nivel de instancia.
- En las instancias de lector, las tablas temporales en disco utilizan el motor de almacenamiento MyISAM, que utiliza el almacenamiento local. Esto se debe a que las instancias de solo lectura no pueden almacenar datos en el volumen del clúster de Aurora.

Seguridad con Amazon Aurora MySQL

La seguridad de Amazon Aurora MySQL se administra en tres niveles:

- Para controlar quién puede realizar acciones de administración de Amazon RDS en clústeres de base de datos e instancias de base de datos de Aurora MySQL, se usa AWS Identity and Access Management (IAM). Cuando se conecta a AWS con credenciales de IAM, la cuenta de AWS debe tener políticas de IAM que concedan los permisos necesarios para realizar operaciones de administración de Amazon RDS. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora

  Si usa IAM para acceder a la consola de Amazon RDS, primero inicie sesión en la Consola de administración de AWS con sus credenciales de usuario de IAM. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
- Asegúrese de crear clústeres de base de datos de Aurora MySQL en una nube pública virtual (VPC) basada en el servicio Amazon VPC. Para controlar qué dispositivos e instancias Amazon EC2 pueden abrir conexiones al punto de conexión y al puerto de la instancia de base de datos para los clústeres de base de datos de Aurora MySQL en una VPC, debe usar un grupo de seguridad de VPC. Puede establecer estas conexiones de puerto y punto de conexión mediante la seguridad de la capa de transporte (TLS). Además, las reglas del firewall de su compañía pueden controlar si los dispositivos que se ejecutan en ella pueden abrir conexiones a una instancia de base de datos. Para obtener más información acerca de las VPC, consulte VPC de Amazon y Amazon Aurora.

  La tenencia de VPC admitida depende de la clase de instancia de base de datos que utilicen los clústeres de base de datos de Aurora MySQL. En el caso de la tenencia de VPC `default`, la VPC se ejecuta en hardware compartido. En el caso de la tenencia de una VPC `dedicated`, la VPC se ejecuta en una instancia de hardware dedicada. Las clases de instancia de base de datos de rendimiento ampliable solo admiten el arrendamiento de VPC predeterminado. Las clases de instancia de base de datos de rendimiento ampliable incluyen las clases de instancia de base de datos db.t2, db.t3 y db.t4g. Todas las demás clases de instancia de base de datos de Aurora MySQL admiten la tenencia de VPC predeterminada y dedicada.

  > **Note**
  >
  > Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la

producción. Para obtener más detalles sobre las clases de instancia T, consulte Utilización de clases de instancia T para el desarrollo y la prueba.

Para obtener más información sobre las clases de instancias, consulte Clases de instancia de base de datos de Amazon Aurora. Para obtener más información acerca de la tenencia de VPC `default` y `dedicated`, consulte Instancias dedicadas en la Guía del usuario de Amazon Elastic Compute Cloud.

- Para autenticar el inicio de sesión y los permisos de un clúster de base de datos de Amazon Aurora MySQL, puede usar cualquiera de los siguientes procedimientos o una combinación de ellos:
  - Puede seguir el mismo procedimiento que con una instancia independiente de MySQL.

    Los comandos como `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` y `SET PASSWORD` funcionan de la misma forma que en las bases de datos locales, al igual que la modificación directa de las tablas de los esquemas de las bases de datos. Para obtener más información, consulte Control de acceso y administración de cuentas en la documentación de MySQL.
  - También puede utilizar la autenticación de base de datos de IAM.

    Con la autenticación de bases de datos de IAM, se autentica en el clúster de base de datos con un usuario o un rol de IAM y un token de autenticación. Un token de autenticación es un valor único que se genera utilizando el proceso de firma Signature Version 4. Mediante la autenticación de base de datos de IAM, puede utilizar las mismas credenciales para controlar el acceso a AWS los recursos y a las bases de datos. Para obtener más información, consulte Autenticación de bases de datos de IAM .

> **Note**
>
> Para obtener más información, consulte Seguridad en Amazon Aurora.

En las siguientes secciones, consulte la información sobre los permisos de usuario para las conexiones Aurora MySQL y TLS con clústeres de base de datos de Aurora MySQL.

Temas

- Privilegios de la cuenta de usuario maestro con Amazon Aurora MySQL.

- Conexiones TLS a clústeres de base de datos de Aurora MySQL

Privilegios de la cuenta de usuario maestro con Amazon Aurora MySQL.

Cuando se crea una instancia de base de datos de Amazon Aurora MySQL, el usuario maestro tiene los privilegios predeterminados que se indican en Privilegios de la cuenta de usuario maestro.

Para proporcionar servicios de administración para cada clúster de base de datos, se crean los usuarios `admin` y `rdsadmin` cuando se crea el clúster de base de datos. Al intentar borrar, cambiar de nombre, modificar la contraseña o cambiar los privilegios de la cuenta `rdsadmin`, se producirá un error.

En los clústeres de base de datos de la versión 2 de Aurora MySQL, los usuarios `admin` y `rdsadmin` se crean cuando se crea el clúster de base de datos. En los clústeres de base de datos de la versión 3 de Aurora MySQL, se crean los usuarios `admin`, `rdsadmin` y `rds_superuser_role`.

> **Important**
>
> Le recomendamos encarecidamente que no utilice el usuario maestro directamente en sus aplicaciones. En lugar de ello, es mejor ceñirse a la práctica recomendada de utilizar un usuario de base de datos creado con los privilegios mínimos necesarios para su aplicación.

Para la administración del clúster de base de datos Aurora MySQL, los comandos estándar `kill` y `kill_query` se han restringido. En su lugar, use los comandos de Amazon RDS `rds_kill` y `rds_kill_query` para terminar las consultas o las sesiones de usuario en las instancias de base de datos Aurora MySQL.

> **Note**
>
> El cifrado de instantáneas de una instancia de base de datos no se admite en la región China (Ningxia).

Conexiones TLS a clústeres de base de datos de Aurora MySQL

Los clústeres de base de datos de Amazon Aurora MySQL admiten conexiones de seguridad de la capa de transporte (TLS) desde aplicaciones con el mismo proceso y la misma clave pública que las instancias de base de datos de RDS para MySQL.

Amazon RDS crea un certificado de TLS e instala el certificado en la instancia de base de datos cuando Amazon RDS aprovisiona la instancia. Estos certificados están firmados por una autoridad de certificación. El certificado TLS incluye el punto de conexión de la instancia de base de datos como nombre común (CN) que el certificado de TLS debe proteger frente a los ataques de suplantación. Como resultado, solo puede usar el punto de conexión del clúster de base de datos para conectarse a un clúster de base de datos con TLS si su cliente admite nombres alternativos de firmante (SAN). De lo contrario, tendrá que usar el punto de enlace de la instancia de una instancia de escritor.

Para obtener más información acerca de cómo descargar certificados, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

Le recomendamos el controlador JDBC de AWS como cliente que admite SAN con TLS. Para obtener más información sobre el controlador JDBC de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del controlador JDBC de Amazon Web Services (AWS).

Temas

- Requerir una conexión TLS a un clúster de base de datos de Aurora MySQL
- Versiones de TLS para Aurora MySQL
- Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora MySQL
- Cifrado de conexiones a un clúster de base de datos de Aurora MySQL

Requerir una conexión TLS a un clúster de base de datos de Aurora MySQL

Puede requerir que todas las conexiones de usuario al clúster de base de datos de Aurora MySQL utilicen TLS mediante el parámetro de clúster de base de datos `require_secure_transport`. De forma predeterminada, el parámetro `require_secure_transport` está definido como `OFF`. Puede establecer el parámetro `require_secure_transport` en `ON` para exigir TLS para las conexiones al clúster de base de datos.

Puede definir el valor del parámetro `require_secure_transport` actualizando el grupo de parámetros de su clúster de base de datos para su clúster de base de datos. No es necesario

reiniciar el clúster de base de datos para que el cambio surta efecto. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.

> **Note**
>
> El parámetro `require_secure_transport` está disponible para las versiones 2 y 3 de Aurora MySQL. Puede establecer este parámetro en un grupo de parámetros de clúster de base de datos personalizado. El parámetro no está disponible en grupos de parámetros de instancia de base de datos.

Cuando el parámetro `require_secure_transport` se establece en `ON` para un clúster de base de datos, un cliente de base de datos puede conectarse al mismo si puede establecer una conexión cifrada. De lo contrario, se devuelve al cliente un mensaje de error similar al siguiente:

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

Versiones de TLS para Aurora MySQL

Aurora MySQL admite las versiones 1.0, 1.1, 1.2 y 1.3 de seguridad de la capa de transporte (TLS). A partir de la versión 3.04.0 y versiones posteriores de Aurora MySQL, puede utilizar el protocolo TLS 1.3 para proteger sus conexiones. En la tabla siguiente se muestra la compatibilidad de TLS para versiones de Aurora MySQL.

Aurora MySQL version	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Predeterminado
Aurora MySQL versión	Soportado	Soportado	Soportado	No compatible	Todas las versiones de TLS admitidas
Aurora MySQL versión (anterior a 3.04.0)	Soportado	Soportado	Soportado	No compatible	Todas las versiones de TLS admitidas

Aurora MySQL version	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Predeterminado
Aurora MySQL versión (3.04.0 y anteriores)	No admitido	No admitido	Soportado	Soportado	Todas las versiones de TLS admitidas

> **Important**
>
> Si utiliza grupos de parámetros personalizados para sus clústeres de Aurora MySQL con la versión 2 y versiones anteriores a la 3.04.0, le recomendamos que utilice TLS 1.2 porque TLS 1.0 y 1.1 son menos seguros. La edición de la comunidad de MySQL 8.0.26 y Aurora MySQL 3.03 y sus versiones secundarias dejaron de admitir las versiones de TLS 1.1 y 1.0. La edición de la comunidad de MySQL 8.0.28 y las versiones 3.04.0 y posteriores compatibles de Aurora MySQL no admiten TLS 1.1 ni TLS 1.0. Si utiliza las versiones 3.04.0 y posteriores de Aurora MySQL, no defina el protocolo TLS en 1.0 y 1.1 en su grupo de parámetros personalizado.
> Para las versiones 3.04.0 y posteriores de Aurora MySQL, la configuración predeterminada es TLS 1.3 y TLS 1.2.

Puede utilizar el parámetro de clúster de base de datos `tls_version` para indicar las versiones de protocolo permitidas. Existen parámetros de cliente similares para la mayoría de las herramientas de cliente o controladores de base de datos. Es posible que algunos clientes anteriores no admitan las versiones de TLS más recientes. De forma predeterminada, el clúster de base de datos intenta utilizar la versión más reciente del protocolo TLS permitida por la configuración del servidor y del cliente.

Establezca el parámetro de clúster de base de datos de `tls_version` en uno de los siguientes valores:

- `TLSv1.3`
- `TLSv1.2`
- `TLSv1.1`

- `TLSv1`

También puede configurar el parámetro `tls_version` como una cadena de lista separada por comas. Si desea utilizar los protocolos TLS 1.2 y TLS 1.0, el parámetro `tls_version` debe incluir todos los protocolos, desde el más bajo hasta el más alto. En este caso, `tls_version` se establece como:

```
tls_version=TLSv1,TLSv1.1,TLSv1.2
```

Para obtener información acerca de cómo modificar parámetros en un clúster de base de datos o un grupo de parámetros de base de datos, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora. Si utiliza la AWS CLI para modificar el parámetro de clúster de base de datos `tls_version`, `ApplyMethod` se debe establecer en `pending-reboot`. Cuando el método de aplicación es `pending-reboot`, los cambios en los parámetros se aplican después de detener y reiniciar los clústeres de base de datos asociados al grupo de parámetros.

Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora MySQL

Mediante el uso de conjuntos de cifrado configurables, puede tener más control sobre la seguridad de las conexiones de su base de datos. Puede especificar una lista de conjuntos de cifrado que desea permitir para proteger las conexiones TLS del cliente a su base de datos. Con conjuntos de cifrado configurables, puede controlar el cifrado de conexión que acepta el servidor de base de datos. Esto evita el uso de cifrados inseguros u obsoletos.

Los conjuntos de cifrado configurables son compatibles con la versión 3 de Aurora MySQL y la versión 2 de Aurora MySQL. Con el fin de especificar la lista de cifrados TLS 1.2, TLS 1.1, TLS 1.0 permitidos para cifrar conexiones, modifique el parámetro del clúster `ssl_cipher`. Establezca el parámetro `ssl_cipher` en un grupo de parámetros de clúster mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Establezca el parámetro `ssl_cipher` en una cadena de valores de cifrado separados por comas para su versión de TLS. Para la aplicación cliente, puede especificar los cifrados que se utilizarán para las conexiones cifradas mediante la opción `--ssl-cipher` cuando se conecta a la base de datos. Para obtener más información acerca de la conexión a la base de datos, consulte Conexión a un clúster de base de datos Amazon Aurora MySQL.

A partir de la versión 3.04.0 y versiones posteriores de Aurora MySQL, puede especificar conjuntos de cifrado TLS 1.3. Para especificar los conjuntos de cifrado TLS 1.3 permitidos, modifique el parámetro `tls_ciphersuites` de su grupo de parámetros. TLS 1.3 ha reducido el número de conjuntos de cifrado disponibles debido a los cambios en la convención de nomenclatura que elimina el mecanismo de intercambio de claves y certificado utilizado. Establezca el parámetro `tls_ciphersuites` en una cadena de valores de cifrado separados por comas para TLS 1.3.

En la tabla siguiente, se muestran los cifrados compatibles junto con el protocolo de cifrado TLS y las versiones del motor Aurora MySQL válidas para cada cifrado.

Cifrado	Protocolo de cifrado	Versiones compatibles con Aurora MySQL
`DHE-RSA-AES128-SHA`	TLS 1.0	3.01.0 y versiones posteriores, todas las versiones anteriores a 2.11.0
`DHE-RSA-AES128-SHA256`	TLS 1.2	3.01.0 y versiones posteriores, todas las versiones anteriores a 2.11.0
`DHE-RSA-AES128-GCM-SHA256`	TLS 1.2	3.01.0 y versiones posteriores, todas las versiones anteriores a 2.11.0
`DHE-RSA-AES256-SHA`	TLS 1.0	3.03.0 y versiones anteriores, todas las versiones anteriores a 2.11.0
`DHE-RSA-AES256-SHA256`	TLS 1.2	3.01.0 y versiones posteriores, todas las versiones anteriores a 2.11.0
`DHE-RSA-AES256-GCM-SHA384`	TLS 1.2	3.01.0 y versiones posteriores, todas las versiones anteriores a 2.11.0

Cifrado	Protocolo de cifrado	Versiones compatibles con Aurora MySQL
ECDHE-RSA-AES128-SHA	TLS 1.0	3.01.0 y versiones posteriores, 2.09.3 y versiones posteriores, 2.10.2 y versiones posteriores
ECDHE-RSA-AES128-SHA256	TLS 1.2	3.01.0 y versiones posteriores, 2.09.3 y versiones posteriores, 2.10.2 y versiones posteriores
ECDHE-RSA-AES128-GCM-SHA256	TLS 1.2	3.01.0 y versiones posteriores, 2.09.3 y versiones posteriores, 2.10.2 y versiones posteriores
ECDHE-RSA-AES256-SHA	TLS 1.0	3.01.0 y versiones posteriores, 2.09.3 y versiones posteriores, 2.10.2 y versiones posteriores
ECDHE-RSA-AES256-SHA384	TLS 1.2	3.01.0 y versiones posteriores, 2.09.3 y versiones posteriores, 2.10.2 y versiones posteriores
ECDHE-RSA-AES256-GCM-SHA384	TLS 1.2	3.01.0 y versiones posteriores, 2.09.3 y versiones posteriores, 2.10.2 y versiones posteriores
TLS_AES_128_GCM_SHA256	TLS 1.3	3.04.0 y versiones posteriores
TLS_AES_256_GCM_SHA384	TLS 1.3	3.04.0 y versiones posteriores
TLS_CHACHA20_POLY1305_SHA256	TLS 1.3	3.04.0 y versiones posteriores

> **Note**
>
> Los cifrados `DHE-RSA` solo son compatibles con las versiones de Aurora MySQL anteriores a la 2.11.0. Las versiones 2.11.0 y posteriores solo admiten cifrados `ECDHE`.

Para obtener información acerca de cómo modificar parámetros en un clúster de base de datos o un grupo de parámetros de base de datos, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora. Si utiliza la CLI para modificar el parámetro de clúster de base de datos de `ssl_cipher`, asegúrese de configurar el `ApplyMethod` en `pending-reboot`. Cuando el método de aplicación es `pending-reboot`, los cambios en los parámetros se aplican después de detener y reiniciar los clústeres de base de datos asociados al grupo de parámetros.

También puede utilizar el comando de la CLI describe-motor-default-clúster-parameters para determinar qué conjuntos de cifrado se admiten actualmente para una familia de grupos de parámetros específica. El siguiente ejemplo muestra cómo obtener los valores permitidos para el parámetro de clúster `ssl_cipher` para la versión 2 de Aurora MySQL.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
mysql5.7

        ...some output truncated...
 {
  "ParameterName": "ssl_cipher",
  "ParameterValue": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-
AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-
SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-SHA384",
  "Description": "The list of permissible ciphers for connection encryption.",
  "Source": "system",
  "ApplyType": "static",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-
RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-
SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-SHA384",
  "IsModifiable": true,
  "SupportedEngineModes": [
   "provisioned"
  ]
```

```
},
      ...some output truncated...
```

Para obtener más información sobre los cifrados, consulte la variable ssl_cipher en la documentación de MySQL. Para obtener más información sobre los formatos de conjuntos de cifrado, consulte la documentación sobre Formato de cadena openssl-ciphers y Formato de cadena openssl-ciphers en el sitio web de OpenSSL.

Cifrado de conexiones a un clúster de base de datos de Aurora MySQL

Para cifrar las conexiones utilizando el cliente `mysql` predeterminado, lance el cliente mysql utilizando el parámetro `--ssl-ca` para hacer referencia a la clave pública. Por ejemplo:

Para MySQL 5.7 y 8.0:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-mode=VERIFY_IDENTITY
```

Para MySQL 5.6:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-verify-server-cert
```

Reemplace *`full_path_to_CA_certificate`* por la ruta completa del certificado de entidad de certificación (CA). Para obtener más información acerca de cómo descargar un certificado, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

Puede exigir conexiones TLS para determinadas cuentas de usuarios. Por ejemplo, puede utilizar una de las siguientes instrucciones, dependiendo de su versión de MySQL, para exigir el uso de conexiones TLS en la cuenta de usuario `encrypted_user`.

Para MySQL 5.7 y 8.0:

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

Para MySQL 5.6:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

Cuando utiliza RDS Proxy, se conecta al punto de conexión del proxy en lugar del punto de conexión del clúster habitual. Puede hacer que SSL/TLS sea obligatorio u opcional para las conexiones al proxy, de la misma manera que para las conexiones directamente al clúster de base de datos de Aurora. Para obtener información sobre el uso RDS Proxy, consulte Amazon RDS Proxy para Aurora.

> **Note**
>
> Para obtener más información acerca de las conexiones TLS con MySQL, consulte la documentación de MySQL.

Actualización de aplicaciones para la conexión a los clústeres de base de datos de MySQL de Aurora con los nuevos certificados TLS

El 13 de enero de 2023, Amazon RDS publicó nuevos certificados de entidades de certificación (CA) para la conexión a sus clústeres de base de datos de Aurora mediante la seguridad de la capa de transporte (TLS). Después, puede encontrar la información sobre la actualización de sus aplicaciones para utilizar los nuevos certificados.

Este tema puede ayudarle a determinar las aplicaciones de cualquier cliente utilizan TLS para conectarse a sus clústeres de base de datos. Si lo hacen, puede comprobar de manera adicional si esas aplicaciones precisan una verificación de certificados para conectarse.

> **Note**
>
> Algunas aplicaciones están configuradas para conectarse a los clústeres de base de datos de MySQL de Aurora solo si pueden verificar con éxito el certificado del servidor.
> Para esas aplicaciones, debe actualizar los almacenes de confianza de la aplicación de su cliente para incluir los nuevos certificados de CA.

Después de que actualice sus certificados de CA en los almacenes de confianza de la aplicación de su cliente, puede rotar los certificados en sus clústeres de base de datos. Recomendamos encarecidamente probar estos procedimientos en un entorno de desarrollo o ensayo antes de implementarlos en sus entornos de producción.

Para obtener más información acerca de la rotación de certificados, consulte Rotar certificados SSL/TLS. Para obtener más información acerca de cómo descargar certificados, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos. Para obtener información sobre el uso de TLS con los clústeres de base de datos de MySQL de Aurora, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

Temas

- Determinación de si alguna aplicación se conecta a su clúster de base de datos de MySQL de Aurora mediante TLS
- Determinación de si un cliente necesita una verificación de certificados para conectarse
- Actualización del almacén de confianza de su aplicación

- Ejemplo de código Java para el establecimiento de conexiones TLS

Determinación de si alguna aplicación se conecta a su clúster de base de datos de MySQL de Aurora mediante TLS

Si utiliza la versión 2 de MySQL de Aurora (compatible con MySQL 5.7) y el esquema de rendimiento se activa, ejecute la siguiente consulta para comprobar si las conexiones utilizan TLS. Para obtener información sobre la habilitación del esquema de rendimiento, consulte Inicio rápido del esquema de rendimiento en la documentación de MySQL.

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

En estos resultados de ejemplo, puede ver que su propia sesión (`admin`) y una aplicación con sesión iniciada como `webapp1` utilizan TLS.

```
+----+-----------------+------------------+-----------------+
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

Determinación de si un cliente necesita una verificación de certificados para conectarse

Puede comprobar si los cliente de JDBC y MySQL precisan la verificación de certificados para conectarse.

JDBC

El siguiente ejemplo con el conector de MySQL/J 8.0 muestra una manera de comprobar las propiedades de conexión de JDBC de una aplicación para determinar si las conexiones exitosas precisan un certificado válido. Para obtener más información sobre todas las opciones de conexión de JDBC para MySQL, consulte Propiedades de la configuración en la documentación de MySQL.

Al utilizar el conector de MySQL/J 8.0, una conexión de TLS precisa una verificación frente al certificado de CA del servidor si sus propiedades de conexión han configurado `sslMode` como `VERIFY_CA` o `VERIFY_IDENTITY`, como en el siguiente ejemplo.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> Si utiliza MySQL Java Connector v5.1.38 o posterior, o MySQL Java Connector v8.0.9 o posterior para conectarse a sus bases de datos, incluso si no ha configurado explícitamente sus aplicaciones para usar TLS al conectarse a sus bases de datos, estos controladores cliente utilizan de forma predeterminada TLS. Además, al utilizar TLS, realizan una verificación parcial del certificado y producen un error al conectarse si el certificado del servidor de base de datos ha caducado.

MySQL

Los siguientes ejemplos con el cliente de MySQL muestran dos maneras de comprobar la conexión a MySQL de un script para determinar si las conexiones exitosas precisan un certificado válido. Para obtener más información sobre todas las opciones de conexión con el cliente de MySQL, consulte Configuración del lado del cliente para las conexiones cifradas en la documentación de MySQL.

Al utilizar el cliente de MySQL 5.7 o 8.0, una conexión TLS precisa una verificación frente al certificado de CA del servidor si para la opción de `--ssl-mode` especifica `VERIFY_CA` o `VERIFY_IDENTITY`, como en el siguiente ejemplo.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

Al utilizar el cliente de MySQL 5.6, una conexión SSL precisa una verificación frente al certificado de CA del servidor si especifica la opción `--ssl-verify-server-cert`, como en el siguiente ejemplo.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

Actualización del almacén de confianza de su aplicación

Para obtener información sobre la actualización del almacén de confianza para las aplicaciones de MySQL, consulte Instalación de certificados de SSL en la documentación de MySQL.

> **Note**
>
> Cuando actualice el almacén de confianza, puede retener certificados antiguos además de añadir los nuevos certificados.

Actualización del almacén de confianza de su aplicación para JDBC

Puede actualizar el almacén de confianza para las aplicaciones que utilizan JDBC para las conexiones TLS.

Para obtener información sobre la descarga del certificado raíz, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

Para obtener secuencias de comandos de ejemplo que importan certificados, consulte Script de muestra para la importación de certificados en su almacén de confianza.

Si utiliza el controlador de JDBC de MySQL en una aplicación, establezca las siguientes propiedades en la aplicación.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

> **Note**
>
> Especifique una contraseña distinta de la que se muestra aquí como práctica recomendada de seguridad.

Cuando inicie la aplicación, establezca las siguientes propiedades.

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

Ejemplo de código Java para el establecimiento de conexiones TLS

El siguiente ejemplo de código muestra cómo configurar la conexión SSL que valida el certificado del servidor mediante JDBC.

```
public class MySQLSSLTest {

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void test(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");


        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.setProperty("sslMode", "VERIFY_IDENTITY");
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);

        Connection connection = DriverManager.getConnection("jdbc:mysql://jagdeeps-ssl-
test.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306",properties);
        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
    }
}
```

> **Important**
>
> Después de que haya determinado que sus conexiones de base de datos utilizan TLS y haya actualizado el almacén de confianza de su aplicación, puede actualizar su base de datos para que utilice los certificados de rds-ca-rsa2048-g1. Para obtener instrucciones, consulte el

paso 3 en Actualización del certificado de entidad de certificación modificando la instancia de base de datos.

Uso de la autenticación Kerberos para Aurora MySQL

Puede usar la autenticación Kerberos para autenticar a los usuarios cuando estos se conecten a su clúster de base de datos de Aurora MySQL. Para ello, configure el clúster de base de datos para utilizar AWS Directory Service for Microsoft Active Directory para la autenticación Kerberos. AWS Directory Service for Microsoft Active Directorytambién se denomina AWS Managed Microsoft AD. Es una función disponible con Directory Service. Para obtener más información, consulte ¿Qué esDirectory Service? en la Guía de administración de AWS Directory Service.

Para empezar, cree un directorio de AWS Managed Microsoft AD para almacenar las credenciales de usuario. A continuación, proporcione a su clúster de base de datos de Aurora MySQL el dominio de Active Directory y otra información. Cuando los usuarios se autentican con el clúster de base de datos de Aurora MySQL, las solicitudes de autenticación se reenvían al directorio AWS Managed Microsoft AD.

Mantener todas las credenciales en el mismo directorio puede ahorrarle tiempo y esfuerzo. Con este método, dispone de un lugar centralizado para almacenar y administrar credenciales de numerosos clústeres de bases de datos. El uso de un directorio también puede mejorar su perfil de seguridad general.

Además, puede acceder a las credenciales desde su propio Microsoft Active Directory en las instalaciones. Para ello, cree una relación de dominio de confianza para que el directorio de AWS Managed Microsoft AD confíe en su Microsoft Active Directory en las instalaciones. De esta manera, los usuarios pueden acceder a los clústeres de base de datos de Aurora MySQL con la misma experiencia de inicio de sesión único (SSO) de Windows que cuando acceden a cargas de trabajo de la red en las instalaciones.

Una base de datos puede usar la autenticación de Kerberos, de AWS Identity and Access Management (IAM), o ambas. Sin embargo, dado que la autenticación Kerberos y de IAM proporcionan diferentes métodos de autenticación, un usuario específico puede iniciar sesión en una base de datos con solo uno u otro método de autenticación, pero no ambos. Para obtener más información acerca de la autenticación IAM, consulte Autenticación de bases de datos de IAM .

Contenido

- Información general de la autenticación Kerberos para clústeres de base de datos de Aurora MySQL
- Limitaciones de la autenticación Kerberos para Aurora MySQL
- Configuración de la autenticación Kerberos para clústeres de base de datos de Aurora MySQL

- Paso 1: crear un directorio con AWS Managed Microsoft AD
- Paso 2: (opcional) crear una relación de confianza para un Active Directory en las instalaciones
- Paso 3: crear un rol de IAM para que lo use Amazon Aurora
- Paso 4: crear y configurar usuarios
- Paso 5: crear o modificar un clúster de base de datos de Aurora MySQL
- Paso 6: crear usuarios de Aurora MySQL que usen la autenticación Kerberos
  - Modificación de un inicio de sesión de Aurora MySQL existente
- Paso 7: configurar un cliente MySQL
- Paso 8: (opcional) configurar la comparación de nombres de usuario que no distinga mayúsculas de minúsculas
- Conexión a Aurora MySQL con autenticación Kerberos
  - Uso del inicio de sesión de Kerberos en Aurora MySQL para conectarse al clúster de base de datos
  - Autenticación Kerberos con bases de datos globales Aurora
  - Migración desde RDS para MySQL a Aurora MySQL
  - Evitar el almacenamiento en caché de tickets
  - Registro para la autenticación Kerberos
- Administración de un clúster de base de datos en un dominio
  - Descripción de la pertenencia a los dominios

Información general de la autenticación Kerberos para clústeres de base de datos de Aurora MySQL

Para configurar la autenticación Kerberos para un clúster de base de datos de Aurora MySQL, realice los siguientes pasos generales. Estos pasos se describen con más detalle más adelante.

1. Utilice AWS Managed Microsoft AD para crear un directorio de AWS Managed Microsoft AD. Puede utilizar la Consola de administración de AWS, la AWS CLI o Directory Service para crear el directorio. Para obtener más detalles, consulte Create your AWS Managed Microsoft AD directory (Creación de su directorio de AWS Managed Microsoft AD) en la Guía de administración de AWS Directory Service.

2. Cree un rol de AWS Identity and Access Management (IAM) que utilice la política de IAM administrada `AmazonRDSDirectoryServiceAccess`. El rol permite a Amazon Aurora realizar llamadas al directorio.

   Para que el rol permita el acceso, el punto de conexión AWS Security Token Service (AWS STS) debe activarse en la Región de AWS para su cuenta de AWS. Los puntos de conexión de AWS STS están activos de forma predeterminada en todas Regiones de AWS y puede usarlos sin ninguna acción posterior. Para obtener más información, consulte Activación y desactivación de AWS STS en una región de Región de AWS en la Guía del usuario de IAM.

3. Cree y configure usuarios en el directorio de AWS Managed Microsoft AD usando las herramientas de Microsoft Active Directory. Para obtener más información sobre la creación de usuarios en su Active Directory, consulte Administrar usuarios y grupos en AWS Managed Microsoft AD en la guía de administración de AWS Directory Service.

4. Cree o modifique un clúster de base de datos de Aurora MySQL. Si utiliza la CLI o la API de RDS en la solicitud de creación, especifique un identificador de dominio con el parámetro `Domain`. Utilice el identificador `d-*` que se ha generado al crear el directorio y el nombre del rol de IAM que ha creado.

   Si modifica un clúster de base de datos de Aurora MySQL ya existente para utilizar la autenticación Kerberos, establezca los parámetros de dominio y rol de IAM para el clúster de base de datos. Busque el clúster de base de datos en la misma VPC que el directorio de dominio.

5. Use las credenciales de usuario principal de Amazon RDS para conectarse al clúster de base de datos de Aurora MySQL. Cree el usuario de base de datos en Aurora MySQL siguiendo las instrucciones de Paso 6: crear usuarios de Aurora MySQL que usen la autenticación Kerberos.

   Los usuarios que cree de esta manera pueden iniciar sesión en el clúster de base de datos de Aurora MySQL con la autenticación Kerberos. Para obtener más información, consulte Conexión a Aurora MySQL con autenticación Kerberos.

Para utilizar la autenticación Kerberos con un Microsoft Active Directory en las instalaciones o autoalojado, cree una relación de confianza entre bosques. Una relación de confianza entre bosques es una relación de confianza entre dos grupos de dominios. La confianza puede ser unidireccional o bidireccional. Para obtener más información acerca de la configuración de relaciones de confianza entre bosques con Directory Service, consulte Cuándo crear una relación de confianza en la Guía de administración de AWS Directory Service.

Limitaciones de la autenticación Kerberos para Aurora MySQL

Las siguientes limitaciones se aplican a la autenticación Kerberos para Aurora MySQL:

- La autenticación Kerberos es compatible con la versión 3.03 de Aurora MySQL y versiones posteriores.

  Para obtener más información sobre la compatibilidad en Región de AWS, consulte Autenticación Kerberos con Aurora MySQL.
- Para usar la autenticación Kerberos con Aurora MySQL, su cliente o conector de MySQL debe usar la versión 8.0.26 o versiones posteriores en las plataformas Unix, y la versión 8.0.27 o versiones posteriores en Windows. De lo contrario, el complemento `authentication_kerberos_client` del lado del cliente no estará disponible y no podrá autenticarse.
- Solo se admite AWS Managed Microsoft AD en Aurora MySQL. Sin embargo, puede unir clústeres de base de datos de Aurora MySQL a dominios compartidos de Managed Microsoft AD propiedad de distintas cuentas de la misma Región de AWS.

  También puede usar su propio Active Directory en las instalaciones. Para obtener más información, consulte Paso 2: (opcional) crear una relación de confianza para un Active Directory en las instalaciones.
- Al utilizar Kerberos para autenticar a un usuario que se conecta al clúster de Aurora MySQL desde clientes MySQL o desde controladores del sistema operativo Windows, de forma predeterminada, las mayúsculas y minúsculas del nombre de usuario de la base de datos deben se iguales que las mayúsculas y minúsculas del usuario de Active Directory. Por ejemplo, si el usuario de Active Directory aparece como `Admin`, el nombre de usuario de la base de datos debe ser `Admin`.

  Sin embargo, ahora puede usar la comparación de nombres de usuario que no distinga mayúsculas de minúsculas con el complemento `authentication_kerberos`. Para obtener más información, consulte Paso 8: (opcional) configurar la comparación de nombres de usuario que no distinga mayúsculas de minúsculas.
- Debe reiniciar las instancias de base de datos del lector después de activar la característica para instalar el complemento `authentication_kerberos`.
- La replicación en instancias de base de datos que no admiten el complemento `authentication_kerberos` puede provocar un error de replicación.
- Para que las bases de datos globales de Aurora utilicen la autenticación Kerberos, debe configurarla para cada clúster de base de datos de la base de datos global.

- El nombre de dominio debe tener menos de 62 caracteres.
- No modifique el puerto del clúster de base de datos después de activar la autenticación Kerberos. Si modifica el puerto, la autenticación Kerberos dejará de funcionar.

Configuración de la autenticación Kerberos para clústeres de base de datos de Aurora MySQL

Utilice AWS Managed Microsoft AD para configurar la autenticación Kerberos para un clúster de base de datos de Aurora MySQL. Para configurar la autenticación Kerberos, siga los pasos que se indican a continuación:

Temas

- Paso 1: crear un directorio con AWS Managed Microsoft AD
- Paso 2: (opcional) crear una relación de confianza para un Active Directory en las instalaciones
- Paso 3: crear un rol de IAM para que lo use Amazon Aurora
- Paso 4: crear y configurar usuarios
- Paso 5: crear o modificar un clúster de base de datos de Aurora MySQL
- Paso 6: crear usuarios de Aurora MySQL que usen la autenticación Kerberos
- Paso 7: configurar un cliente MySQL
- Paso 8: (opcional) configurar la comparación de nombres de usuario que no distinga mayúsculas de minúsculas

Paso 1: crear un directorio con AWS Managed Microsoft AD

Directory Service crea un directorio de Active Directory completamente administrado en la nube de AWS. Cuando crea un directorio de AWS Managed Microsoft AD, Directory Service crea dos controladores de dominio y servidores del sistema de nombres de dominio (DNS) en su nombre. Los servidores de directorios se crean en diferentes subredes de una VPC. Esta redundancia ayuda a garantizar que su directorio permanezca accesible incluso si ocurre un error.

Cuando crea un directorio de AWS Managed Microsoft AD, Directory Service realiza en su nombre las siguientes tareas:

- Configurar un Active Directory dentro de la VPC.

- Crea una cuenta de administrador del directorio con el nombre de usuario `Admin` y la contraseña especificada. Esta cuenta le permite administrar el directorio.

  > **Note**
  >
  > Asegúrese de guardar esta contraseña. Directory Service no la almacena. Es posible restablecerla, pero no recuperarla.

- Crea un grupo de seguridad para los controladores del directorio.

Al lanzar AWS Managed Microsoft AD, AWS crea una unidad organizativa (OU) que contiene todos los objetos del directorio. Esta unidad organizativa tiene el nombre de NetBIOS que introdujo al crear el directorio. Se encuentra en la raíz del dominio, que es propiedad y está administrada por AWS.

La cuenta `Admin` que se creó con el directorio de AWS Managed Microsoft AD dispone de permisos para realizar las actividades administrativas más habituales para la unidad organizativa, entre las que se incluyen:

- Crear, actualizar o eliminar usuarios
- Añadir recursos a su dominio, como servidores de archivos o de impresión y, a continuación, asignar permisos para esos recursos a usuarios dentro de la unidad organizativa
- Crear unidades organizativas y contenedores adicionales
- Delegar autoridad
- Restaurar objetos eliminados de la papelera de reciclaje de Active Directory
- Ejecutar módulos de AD y DNS de Windows PowerShell en el servicio web de Active Directory

La cuenta `Admin` también tiene derechos para realizar las siguientes actividades en todo el dominio:

- Administrar configuraciones DNS (agregar, quitar o actualizar registros, zonas y programas de envío).
- Ver logs de eventos DNS
- Ver logs de eventos de seguridad

Para crear un directorio con AWS Managed Microsoft AD

1. Inicie sesión en Consola de administración de AWS y abra la consola de Directory Service en https://console.aws.amazon.com/directoryservicev2/.
2. En el panel de navegación, elija Directories (Directorios) y, a continuación, Set up Directory (Configurar directorio).
3. Elija AWS Managed Microsoft AD. AWS Managed Microsoft AD es la única opción que puede usar actualmente con Amazon RDS.
4. Introduzca la información siguiente:

   Nombre de DNS del directorio

   El nombre completo del directorio, como por ejemplo **`corp.example.com`**.

   Nombre NetBIOS del directorio

   El nombre abreviado del directorio, como **`CORP`**.

   Descripción del directorio

   (Opcional) Descripción del directorio.

   Contraseña de administrador

   Contraseña del administrador del directorio. El proceso de creación de directorios crea una cuenta de administrador con el nombre de usuario Admin y esta contraseña.

   La contraseña del administrador del directorio no puede contener la palabra "admin". La contraseña distingue entre mayúsculas y minúsculas y debe tener un mínimo de 864 caracteres y un máximo de 64. También debe contener al menos un carácter de tres de las siguientes categorías:

   - Letras minúsculas (a–z)
   - Letras mayúsculas (A–Z)
   - Números (0–9)
   - Caracteres no alfanuméricos (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

   Confirm password

   Se vuelve a introducir la contraseña del administrador.

5. Elija Next (Siguiente).

6. Escriba la siguiente información en la sección Networking (Redes) y luego seleccione Next (Siguiente):

   VPC

   VPC del directorio. Cree el clúster de base de datos de Aurora MySQL en esta misma VPC.

   Subredes

   Subredes de los servidores del directorio. Las dos subredes deben estar en diferentes zonas de disponibilidad.

7. Revise la información del directorio y haga los cambios necesarios. Cuando la información sea correcta, seleccione Create directory (Crear directorio).
   Página de detalles del directorio durante la creación

El directorio tarda varios minutos en crearse. Cuando se haya creado correctamente, el valor de Status (Estado) cambiará a Active (Activo).

Para consultar información de su directorio, seleccione el nombre del directorio en la descripción de directorios. Anote el valor de ID del directorio porque lo necesitará cuando cree o modifique su clúster de base de datos de Aurora MySQL.
ID del directorio en la página de detalles del directorio

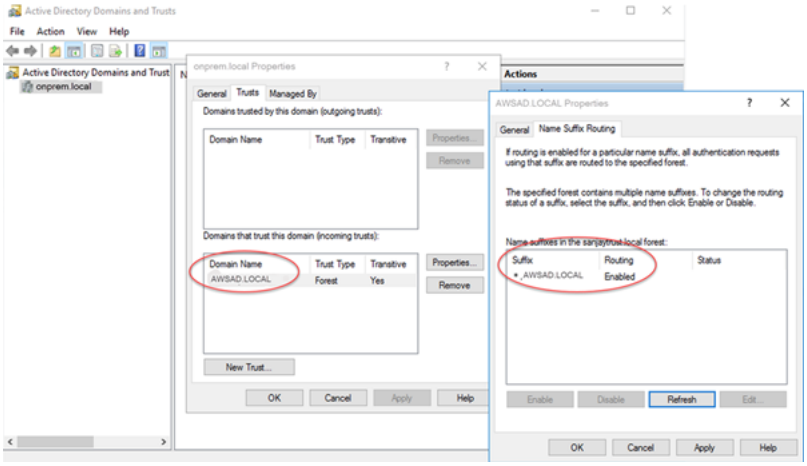
Paso 2: (opcional) crear una relación de confianza para un Active Directory en las instalaciones

Si no planea usar su propio Microsoft Active Directory local, vaya a Paso 3: crear un rol de IAM para que lo use Amazon Aurora.

Para usar la autenticación Kerberos con Active Directory en las instalaciones, debe crear una relación de dominio de confianza entre Microsoft Active Directory en las instalaciones y el directorio AWS Managed Microsoft AD (creado en Paso 1: crear un directorio con AWS Managed Microsoft AD). La confianza puede ser unidireccional, donde el directorio AWS Managed Microsoft AD confía en Microsoft Active Directory local. La confianza también puede ser bidireccional, donde ambos Active Directories confían entre sí. Para obtener más información acerca de la configuración de relaciones de confianza con Directory Service, consulte Cuándo crear una relación de confianza en la guía de administración de AWS Directory Service.

> **Note**
>
> Si utiliza Microsoft Active Directory en las instalaciones:
>
> - Los clientes de Windows no pueden conectarse con utilizando punto de conexión personalizados de Aurora. Para obtener más información, consulte Conexiones de puntos de conexión de Amazon Aurora.
> - Para bases de datos globales:
>   - Los clientes de Windows solo pueden conectarse mediante los puntos de enlace de la instancia o los del clúster en la Región de AWS principal de la base de datos global.
>   - Los clientes de Windows no pueden conectarse mediante los puntos de enlace del clúster en Regiones de AWS secundarias.

Asegúrese de que el nombre de dominio local de Microsoft Active Directory incluya un enrutamiento de sufijo DNS que corresponda a la relación de confianza recién creada. En la siguiente captura de pantalla, se muestra un ejemplo.



Paso 3: crear un rol de IAM para que lo use Amazon Aurora

Para que Amazon Aurora llame a Directory Service en su nombre, necesita un rol de AWS Identity and Access Management (IAM) que utilice la política de IAM administrada `AmazonRDSDirectoryServiceAccess`. Este rol permite a Aurora realizar llamadas a Directory Service.

Cuando crea un clúster de base de datos con la Consola de administración de AWS y tiene el permiso `iam:CreateRole`, la consola crea este rol automáticamente. En este caso, el nombre del

rol es `rds-directoryservice-kerberos-access-role`. De no ser así, debe crear el rol de IAM manualmente. Cuando cree este rol de IAM, elija `Directory Service` y asocie la política administrada de AWS `AmazonRDSDirectoryServiceAccess` a este.

A fin de obtener más información acerca de la creación de roles de IAM para un servicio, consulte Creación de un rol para delegar permisos a un servicio de AWS en la guía del usuario de IAM.

Opcionalmente, puede crear políticas con los permisos requeridos en vez de utilizar la política de IAM administrad `AmazonRDSDirectoryServiceAccess`. En este caso, el rol de IAM debe tener la siguiente política de confianza de IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

El rol debe también tener la siguiente política de rol de IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
```

```
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Paso 4: crear y configurar usuarios

Puede crear usuarios con la herramienta Usuarios y equipos de Active Directory. Esta herramienta forma parte de las herramientas Active Directory Domain Services y Active Directory Lightweight Directory Services. Los usuarios representan a las personas físicas o entidades que tienen acceso al directorio.

Para crear usuarios en un directorio de Directory Service, tiene que usar una instancia en las instalaciones o de Amazon EC2 basada en Microsoft Windows que esté unida a su directorio de Directory Service. Debe iniciar sesión en la instancia como usuario con privilegios para crear usuarios. Para obtener más información, consulte Administrar usuarios y grupos de AWS Managed Microsoft AD en la Guía de administración de AWS Directory Service.

Paso 5: crear o modificar un clúster de base de datos de Aurora MySQL

Cree o modifique un clúster de base de datos de Aurora MySQL para usarlo con su directorio. Puede utilizar la consola, la AWS CLI o la API de RDS para asociar un clúster de base de datos con un directorio. Puede realizar esta tarea de una de las siguientes formas:

- Cree un nuevo clúster de base de datos de Aurora MySQL con la consola, el comando create-db-cluster de la CLI o la operación CreateDBCluster de la API de RDS.

  Para obtener instrucciones, consulte Creación de un clúster de base de datos de Amazon Aurora.
- Modifique un clúster de base de datos de Aurora MySQL existente con la consola, el comando modify-db-cluster de la CLI o la operación ModifyDBCluster de la API de RDS.

  Para obtener instrucciones, consulte Modificación de un clúster de base de datos de Amazon Aurora.

- Restaure un clúster de base de datos de Aurora MySQL a partir de una instantánea de base de datos con la consola, el comando restore-db-cluster-from-snapshot de la CLI o la operación RestoreDBClusterFromSnapshot de la API de RDS.

  Para obtener instrucciones, consulte Restauración de una instantánea de clúster de base de datos.
- Restaure un clúster de base de datos de Aurora MySQL a un punto en el tiempo con la consola, el comando restore-db-cluster-to-point-in-time de la CLI o la operación RestoreDBClusterToPointInTime de la API de RDS.

  Para obtener instrucciones, consulte Restauración de un clúster de base de dato a un momento indicado.

La autenticación Kerberos solo es compatible con clústeres de base de datos de Aurora MySQL en una VPC. El clúster de base de datos puede estar en la misma VPC que el directorio o en una VPC diferente. La VPC del clúster de base de datos tiene que tener un grupo de seguridad de VPC que permita la comunicación saliente con su directorio.

Consola

Si utiliza la consola para crear, modificar o restaurar un clúster de base de datos, elija Kerberos authentication (Autenticación de Kerberos) en la sección Database authentication (Autenticación de base de datos). Elija Browse Directory (Examinar directorio) y, a continuación, seleccione el directorio o elija Create a new directory (Crear un nuevo directorio).
Configuración de la autenticación Kerberos al crear un clúster de base de datos

AWS CLI

Puede utilizar la AWS CLI o la API de RDS para asociar un clúster de base de datos con un directorio. Es necesario incluir los parámetros siguientes para que el clúster de base de datos pueda usar el directorio de dominio que ha creado:

- Para el parámetro `--domain`, utilice el identificador de dominio (identificador "d-*") que se generó cuando creó el directorio.
- Para el parámetro `--domain-iam-role-name`, utilice el rol que creó que usa la política `AmazonRDSDirectoryServiceAccess` de IAM administrada.

Por ejemplo, el siguiente comando de la CLI modifica un clúster de base de datos para que use un directorio.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Si modifica un clúster de base de datos para activar la autenticación Kerberos, reinicie las instancias de base de datos del lector después de hacer el cambio.

Paso 6: crear usuarios de Aurora MySQL que usen la autenticación Kerberos

El clúster de base de datos está unido al dominio de AWS Managed Microsoft AD. Por lo tanto, puede crear usuarios de Aurora MySQL a partir de usuarios de Active Directory en su dominio. Los permisos de base de datos se administran mediante permisos estándar de Aurora MySQL que se conceden y revocan a estos usuarios.

Puede permitir que un usuario de Active Directory se autentique con Aurora MySQL. Para ello, primero use las credenciales del usuario principal de Amazon RDS para conectarse al clúster de base de datos de Aurora MySQL igual que con cualquier otro clúster de base de datos. Después de iniciar sesión, cree un usuario autenticado externamente con la autenticación Kerberos en Aurora MySQL, tal y como se muestra a continuación:

```
CREATE USER user_name@'host_name' IDENTIFIED WITH 'authentication_kerberos' BY
 'realm_name';
```

- Sustituya *user_name* por el nombre de usuario. Los usuarios (tanto humanos como aplicaciones) del dominio pueden conectarse ahora al clúster de base de datos desde un equipo cliente unido al dominio utilizando la autenticación Kerberos.
- Sustituya *host_name* por el nombre del host. Puede utilizar `%` como comodín. También puede usar direcciones IP específicas para el nombre de host.
- Sustituya *realm_name* por el nombre del ámbito del directorio del dominio. El nombre del ámbito suele ser el mismo que el nombre de dominio DNS en mayúsculas (por ejemplo, `CORP.EXAMPLE.COM`). Un ámbito es un grupo de sistemas que utilizan el mismo centro de distribución de claves de Kerberos.

En el siguiente ejemplo, se crea un usuario de base de datos con el nombre `Admin` que se autentica en Active Directory con el nombre de ámbito `MYSQL.LOCAL`.

```
CREATE USER Admin@'%' IDENTIFIED WITH 'authentication_kerberos' BY 'MYSQL.LOCAL';
```

Modificación de un inicio de sesión de Aurora MySQL existente

También puede modificar un inicio de sesión de Aurora MySQL existente para usar la autenticación Kerberos mediante la siguiente sintaxis:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Paso 7: configurar un cliente MySQL

Para configurar un cliente MySQL, siga estos pasos:

1. Cree un archivo `krb5.conf` (o equivalente) para apuntar al dominio.
2. Verifique que el tráfico puede fluir entre el host cliente y Directory Service. Use una utilidad de red como, por ejemplo, Netcat, para lo siguiente:
   - Verificar el tráfico sobre DNS para el puerto 53.
   - Verificar el tráfico sobre TCP/UDP para el puerto 52 y para Kerberos, lo que incluye los puertos 88 y 464 para Directory Service.
3. Verifique que el tráfico puede fluir entre el host cliente y la instancia de base de datos sobre el puerto de base de datos. Por ejemplo, utilice `mysql` para conectarse a la base de datos y acceder a ella.

A continuación, se muestra un contenido de `krb5.conf` de ejemplo para AWS Managed Microsoft AD.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

A continuación, se muestra un contenido de `krb5.conf` de ejemplo para un Microsoft Active Directory en las instalaciones.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Paso 8: (opcional) configurar la comparación de nombres de usuario que no distinga mayúsculas de minúsculas

De forma predeterminada, las mayúsculas y minúsculas del nombre de usuario de la base de datos de MySQL deben ser iguales que las del inicio de sesión en Active Directory. Sin embargo,

ahora puede usar la comparación de nombres de usuario que no distinga mayúsculas de minúsculas con el complemento `authentication_kerberos`. Para ello, defina el parámetro `authentication_kerberos_caseins_cmp` del clúster de base de datos en `true`.

Utilización de la comparación de nombres de usuario que no distingue mayúsculas de minúsculas

1. Cree un grupo de parámetros de clúster de base de datos personalizado. Siga los procedimientos indicados en Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.
2. Edite el nuevo grupo de parámetros para establecer el valor de `authentication_kerberos_caseins_cmp` en `true`. Siga los procedimientos indicados en Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.
3. Asocie el grupo de parámetros del clúster de base de datos con el clúster de base de datos de MySQL. Siga los procedimientos indicados en Asociación de un grupo de parámetros de clúster de base de datos con un clúster de base de datos en Amazon Aurora.
4. Reinicie el clúster de base de datos.

Conexión a Aurora MySQL con autenticación Kerberos

Para evitar errores, utilice un cliente MySQL con la versión 8.0.26 o posterior en plataformas Unix y la versión 8.0.27 o posterior en Windows.

Uso del inicio de sesión de Kerberos en Aurora MySQL para conectarse al clúster de base de datos

Para conectarse a Aurora MySQL con la autenticación Kerberos, inicie sesión como un usuario de la base de datos que haya creado siguiendo las instrucciones que se indican en Paso 6: crear usuarios de Aurora MySQL que usen la autenticación Kerberos.

En el símbolo del sistema, conéctese a uno de los puntos de conexión asociados a su clúster de base de datos de Aurora MySQL. Cuando se le pida la contraseña, escriba la contraseña de Kerberos asociada a ese nombre de usuario.

Al autenticarse con Kerberos, se genera un ticket de concesión de tickets (TGT) si aún no existe ninguno. El complemento `authentication_kerberos` usa el TGT para obtener un ticket de servicio, que luego se presenta al servidor de base de datos Aurora MySQL.

Puede utilizar el cliente MySQL para conectarse a Aurora MySQL con la autenticación Kerberos mediante Windows o Unix.

Unix

Puede conectarse con uno de los siguientes métodos:

- Obtenga el TGT manualmente. En este caso, no es necesario proporcionar la contraseña al cliente MySQL.
- Proporcione la contraseña para iniciar sesión en Active Directory directamente al cliente MySQL.

El complemento del lado del cliente se admite en plataformas Unix para las versiones del cliente MySQL 8.0.26 y posteriores.

Para conectarse obteniendo el TGT de forma manual

1. En la interfaz de la línea de comandos, utilice el siguiente comando para obtener el TGT.

   ```
   kinit user_name
   ```

2. Utilice el siguiente comando `mysql` para iniciar sesión en el punto de conexión de la instancia de base de datos de su clúster de base de datos.

   ```
   mysql -h DB_instance_endpoint -P 3306 -u user_name -p
   ```

   > **Note**
   >
   > La autenticación puede fallar si el keytab se rota en la instancia de base de datos. En este caso, vuelva a ejecutar `kinit` para obtener un nuevo TGT.

Para conectarse directamente

1. En la interfaz de la línea de comandos, utilice el siguiente comando `mysql` para iniciar sesión en el punto de conexión de la instancia de base de datos de su clúster de base de datos.

   ```
   mysql -h DB_instance_endpoint -P 3306 -u user_name -p
   ```

2. Introduzca la contraseña del usuario de Active Directory.

Windows

En Windows, la autenticación se realiza normalmente al iniciar sesión, por lo que no es necesario obtener el TGT manualmente para conectarse al clúster de base de datos de Aurora MySQL. Las mayúsculas y minúsculas del nombre de usuario de la base de datos deben ser iguales que las mayúsculas y minúsculas del usuario en Active Directory. Por ejemplo, si el usuario de Active Directory aparece como `Admin`, el nombre de usuario de la base de datos debe ser `Admin`.

El complemento del lado del cliente se admite en Windows para las versiones 8.0.27 y posteriores del cliente MySQL.

Para conectarse directamente

- En la interfaz de la línea de comandos, utilice el siguiente comando `mysql` para iniciar sesión en el punto de conexión de la instancia de base de datos de su clúster de base de datos.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name
```

Autenticación Kerberos con bases de datos globales Aurora

La autenticación Kerberos para Aurora MySQL se admite en bases de datos globales de Aurora. Para autenticar a los usuarios del clúster de base de datos secundario mediante el Active Directory del clúster de base de datos principal, replique el Active Directory en la Región de AWS secundaria. La autenticación Kerberos se activa en el clúster secundario con el mismo ID de dominio que en el clúster principal. La replicación de AWS Managed Microsoft AD solo se admite con la versión Enterprise de Active Directory. Para obtener más información, consulte Multi-Region replication (Replicación en varias regiones) en la Guía de administración de AWS Directory Service.

Migración desde RDS para MySQL a Aurora MySQL

Después de migrar desde RDS para MySQL con la autenticación Kerberos habilitada a Aurora MySQL, modifique los usuarios creados con el complemento `auth_pam` para que usen el complemento `authentication_kerberos`. Por ejemplo:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Evitar el almacenamiento en caché de tickets

Si no existe un TGT válido cuando se inicia la aplicación cliente de MySQL, la aplicación puede obtener el TGT y almacenarlo en caché. Si desea evitar que el TGT se almacene en caché, defina un parámetro de configuración en el archivo `/etc/krb5.conf`.

> **Note**
>
> Esta configuración solo se aplica a los hosts de cliente que ejecutan Unix, no Windows.

Para evitar el almacenamiento en caché de TGT

- Añada una sección `[appdefaults]` a `/etc/krb5.conf` de la manera siguiente:

```
[appdefaults]
  mysql = {
    destroy_tickets = true
  }
```

Registro para la autenticación Kerberos

La variable de entorno `AUTHENTICATION_KERBEROS_CLIENT_LOG` establece el nivel de registro para la autenticación Kerberos. Puede utilizar los registros para realizar la depuración del lado del cliente.

Los valores permitidos son del 1 al 5. Los mensajes de registro se escriben en la salida de error estándar. En la tabla siguiente se describe cada nivel de registro.

Nivel de registro	Descripción
1 o no configurado	Sin registro
2	Mensajes de error
3	Mensajes de error y advertencia
4	Mensajes de error, advertencia e información

Nivel de registro	Descripción
5	Mensajes de error, advertencia, información y depuración

Administración de un clúster de base de datos en un dominio

Puede usar la AWS CLI o la API de RDS para administrar el clúster de base de datos y su relación con su Active Directory administrado. Por ejemplo, puede asociar un Active Directory para la autenticación Kerberos y desasociar un Active Directory para desactivar la autenticación Kerberos. También puede mover un clúster de base de datos para que lo autentique externamente un Active Directory en otro.

Por ejemplo, con la API de Amazon RDS puede hacer lo siguiente:

- Para volver a intentar activar la autenticación Kerberos en una pertenencia que ha dado un error, use la operación de API `ModifyDBInstance` y especifique el ID de directorio de la pertenencia actual.
- Para actualizar el nombre del rol de IAM para la suscripción, use la operación `ModifyDBInstance` de la API y especifique el ID del directorio de la suscripción actual y el nuevo rol de IAM.
- Para desactivar la autenticación Kerberos en un clúster de base de datos, utilice la operación de API `ModifyDBInstance` y especifique none como parámetro de dominio.
- Para mover un clúster de base de datos de un dominio a otro, use la operación de API `ModifyDBInstance` y especifique el identificador del nuevo dominio como parámetro del dominio.
- Para generar una lista de pertenencias de cada clúster de base de datos, utilice la operación de API `DescribeDBInstances`.

Descripción de la pertenencia a los dominios

Una vez que haya creado o modificado un clúster de base de datos, este se convierte en miembro del dominio. Si desea ver el estado de pertenencia al dominio del clúster de base de datos, ejecute el comando de la CLI describe-db-clusters. El estado del clúster de base de datos puede ser uno de los siguientes:

- `kerberos-enabled`: el clúster de base de datos tiene activada la autenticación Kerberos.
- `enabling-kerberos`: AWS está realizando el proceso de activación de la autenticación Kerberos en este clúster de base de datos.
- `pending-enable-kerberos`: la activación de la autenticación Kerberos está pendiente en este clúster de base de datos.
- `pending-maintenance-enable-kerberos`: AWS intentará activar la autenticación Kerberos en el clúster de base de datos durante el próximo periodo de mantenimiento programado.
- `pending-disable-kerberos`: la desactivación de la autenticación Kerberos está pendiente en este clúster de base de datos.
- `pending-maintenance-disable-kerberos`: AWS intentará desactivar la autenticación Kerberos en el clúster de base de datos durante el próximo periodo de mantenimiento programado.
- `enable-kerberos-failed`: un problema de configuración ha impedido que AWS active la autenticación Kerberos en el clúster de base de datos. Compruebe y corrija la configuración antes de volver a ejecutar el comando para modificar el clúster de base de datos.
- `disabling-kerberos`: AWS está realizando el proceso de desactivación de la autenticación Kerberos en este clúster de base de datos.

Una solicitud para activar la autenticación Kerberos puede generar un error a causa de un problema de conectividad de la red o de un rol de IAM incorrecto. Por ejemplo, supongamos que crea un clúster de base de datos o modifica un clúster de base de datos ya existente y se produce un error en el intento de activar la autenticación Kerberos. Si esto sucede, vuelva a ejecutar el comando modify o modifique el clúster de base de datos recién creado para unirse al dominio.

Migración de datos a un clúster de base de datos de Amazon Aurora MySQL

Tiene varias opciones para migrar datos desde una base de datos existente a un clúster de base de datos MySQL en Amazon Aurora. Las opciones de migración dependen también de la base de datos desde la que se realiza la migración y del tamaño de los datos que se van a migrar.

Existen dos tipos de trabajos distintos de migración: física y lógica. La migración física supone que las copias físicas de archivos de base de datos se utilizan para migrar la base de datos. La migración lógica supone que la migración se lleva a cabo aplicando cambios lógicos en la base de datos, tales como inserciones, actualizaciones y eliminaciones.

La migración física tiene las siguientes ventajas:

- La migración física es más rápida que la migración lógica, especialmente para bases de datos grandes.
- El desempeño de la base de datos no sufre cuando se toma un backup para migración física.
- La migración física puede migrar todo en la base de datos origen, incluidos componentes de base de datos complejos.

La migración física tiene las siguientes limitaciones:

- El parámetro `innodb_page_size` se debe establecer en su valor predeterminado (`16KB`).
- El parámetro `innodb_data_file_path` debe configurarse con un solo archivo de datos que utilice el nombre de archivo de datos predeterminado `"ibdata1:12M:autoextend"`. Las bases de datos con dos archivos de datos, o con un archivo de datos con un nombre diferente, no se pueden migrar con este método.

  A continuación se muestran ejemplos de nombres de archivo no permitidos: `"innodb_data_file_path=ibdata1:50M; ibdata2:50M:autoextend"` y `"innodb_data_file_path=ibdata01:50M:autoextend"`.
- El parámetro `innodb_log_files_in_group` se debe establecer en su valor predeterminado (`2`).

La migración lógica tiene las siguientes ventajas:

- Puede migrar subconjuntos de la base de datos, tales como tablas específicas o partes de una tabla.

- Los datos se pueden migrar con independencia de la estructura de almacenamiento físico.

La migración lógica tiene las siguientes limitaciones:

- La migración lógica es habitualmente más lenta que la migración física.
- Los componentes de base de datos complejos pueden ralentizar el proceso de migración lógica. En algunos casos, los componentes de base de datos complejos pueden incluso bloquear la migración lógica.

En la siguiente tabla se describen sus opciones y el tipo de migración para cada opción.

Migrar desde	Migration type	Solución
Una instancia de base de datos de RDS for MySQL	Física	Puede migrar desde una instancia de base de datos de RDS for MySQL si crea primero una réplica de lectura de Aurora MySQL de una instancia de base de datos de MySQL. Cuando el retardo de la réplica entre la instancia de base de datos MySQL y la réplica de lectura de Aurora MySQL sea 0, podrá indicar a las aplicaciones de cliente que lean de la réplica de lectura de Aurora y, a continuación, detener la replicación para convertir la réplica de lectura de Aurora MySQL en un clúster de base de datos de Aurora MySQL independiente para lectura y escritura. Para obtener más información, consulte Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora.
Una instantánea de base de datos de RDS for MySQL	Física	Puede migrar datos directamente de una instantánea de base de datos de RDS for MySQL a un clúster de base de datos de Amazon Aurora MySQL. Para obtener más información, consulte Migración de una instantánea de RDS for MySQL a Aurora.
Una base de datos MySQL externa a Amazon RDS	Lógica	Puede crear un volcado de los datos con la utilidad `mysqldump` y, a continuación, importar esos datos a un

Migrar desde	Migration type	Solución
		clúster de base de datos de Amazon Aurora MySQL. Para obtener más información, consulte Migración lógica de MySQL a Amazon Aurora MySQL mediante mysqldump. Para exportar metadatos para usuarios de bases de datos durante la migración desde una base de datos MySQL externa, también puede utilizar el comando del intérprete de comandos de MySQL en lugar de `mysqldump` . Para obtener más información, consulte Instance Dump Utility, Schema Dump Utility, and Table Dump Utility. **Note** La utilidad mysqlpump está en desuso a partir de MySQL 8.0.34.
Una base de datos MySQL externa a Amazon RDS	Física	Puede copiar los archivos de copia de seguridad de la base de datos en un bucket de Amazon Simple Storage Service (Amazon S3) y, a continuac ión, restaurar un clúster de base de datos de Amazon Aurora MySQL a partir de esos archivos. Esta opción puede ser bastante más rápida que migrar los datos con `mysqldump` . Para obtener más información, consulte Migración física de MySQL con Percona XtraBackup y Amazon S3.

Migrar desde	Migration type	Solución
Una base de datos MySQL externa a Amazon RDS	Lógica	Puede guardar los datos de la base de datos como archivos de texto y copiar esos archivos en un bucket de Amazon S3. A continuación, puede cargar esos datos en un clúster de base de datos de Aurora MySQL con el comando `LOAD DATA FROM S3` de MySQL. Para obtener más información, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.
Una bases de datos que no sea compatible con MySQL	Logical (Lógica)	Puede utilizar AWS Database Migration Service (AWS DMS) para migrar datos desde una base de datos que no sea compatible con MySQL. Para obtener más información acerca de AWS DMS, consulte ¿Qué es AWS Database Migration Service?

> **Note**
>
> Si está migrando una base de datos de MySQL externa a Amazon RDS, las opciones de migración descritas en la tabla solo se admiten si su base de datos admite los espacios de tabla InnoDB o MyISAM.
> Si la base de datos MySQL que va a migrar a Aurora MySQL usa `memcached`, quite `memcached` antes de migrarla.
> No se puede migrar a la versión 3.05 o posteriores de Aurora MySQL desde algunas versiones anteriores de MySQL 8.0, como 8.0.11, 8.0.13 y 8.0.15. Le recomendamos que actualice a la versión 8.0.28 de MySQL antes de realizar la migración.

Migración de datos desde una base de datos MySQL externa a un clúster de base de datos de Amazon Aurora MySQL

Si la base de datos admite espacios de tablas InnoDB o MyISAM, dispone de estas opciones para migrar los datos a un clúster de base de datos de Amazon Aurora MySQL:

- Puede crear un volcado de los datos con la utilidad `mysqldump` y, a continuación, importar esos datos a un clúster de base de datos de Amazon Aurora MySQL. Para obtener más información, consulte Migración lógica de MySQL a Amazon Aurora MySQL mediante mysqldump.
- Puede copiar los archivos de copia de seguridad completos e incrementales de la base de datos a un bucket de Amazon S3 y, a continuación, restaurar un clúster de base de datos de Amazon Aurora MySQL a partir de dichos archivos. Esta opción puede ser bastante más rápida que migrar los datos con `mysqldump`. Para obtener más información, consulte Migración física de MySQL con Percona XtraBackup y Amazon S3.

Temas

- Migración física de MySQL con Percona XtraBackup y Amazon S3
- Migración lógica de MySQL a Amazon Aurora MySQL mediante mysqldump

Migración física de MySQL con Percona XtraBackup y Amazon S3

Puede copiar los archivos de copia de seguridad completos e incrementales de la base de datos MySQL 5.7 o 8.0 de origen a un bucket de Amazon S3. A continuación, puede realizar la restauración a un clúster de base de datos de Amazon Aurora MySQL con la misma versión principal del motor de base de datos de esos archivos.

Esta opción puede ser bastante más rápida que migrar datos mediante `mysqldump`, ya que `mysqldump` repite todos los comandos para volver a crear el esquema y los datos a partir de la base de datos origen en el nuevo clúster de base de datos de Aurora MySQL. Al copiar los archivos de datos de MySQL de origen, Aurora MySQL puede utilizar inmediatamente esos archivos como datos para el clúster de base de datos de Aurora MySQL.

También puede reducir el tiempo de inactividad mediante el uso de la replicación de registros binarios durante el proceso de migración. Si usa la replicación de logs binarios, la base de datos MySQL externa permanece abierta a las transacciones mientras se migran los datos al clúster de base de datos de Aurora MySQL. Una vez creado el clúster de base de datos de Aurora MySQL, se

usa la replicación de registros binarios para sincronizar el clúster de base de datos de Aurora MySQL con las transacciones que han tenido lugar después de la copia de seguridad. Cuando el clúster de base de datos de Aurora MySQL DB esté sincronizado con la base de datos MySQL, se termina la migración y se produce el cambio al clúster de base de datos de Aurora MySQL para las nuevas transacciones. Para obtener más información, consulte Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL mediante replicación.

Contenido

- Limitaciones y consideraciones
- Antes de empezar
  - Instalación de Percona XtraBackup
  - Permisos necesarios
  - Creación del rol de servicio de IAM
- Copia de seguridad de archivos para restaurarlos como un clúster de base de datos de Amazon Aurora MySQL
  - Creación de una copia de seguridad completa con Percona XtraBackup
  - Uso de copias de seguridad incrementales con Percona XtraBackup
  - Consideraciones sobre copias de seguridad
- Restauración de un clúster de base de datos de Amazon Aurora MySQL desde un bucket de Amazon S3
- Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL mediante replicación
  - Configuración de la base de datos MySQL externa y el clúster de base de datos de Aurora MySQL para la replicación cifrada
  - Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL externa
- Reducir el tiempo de la migración física a Amazon Aurora MySQL
  - Tipos de tablas no admitidas
  - Cuentas de usuario con privilegios no admitidos
  - Privilegios dinámicos de la versión 3 de Aurora MySQL
  - Objetos almacenados con 'rdsadmin'@'localhost' como definidor

Limitaciones y consideraciones

Las siguientes limitaciones y consideraciones se aplican a la restauración a un clúster de base de datos de Amazon Aurora MySQL a partir de un bucket de Amazon S3:

- Solo puede migrar los datos a un nuevo clúster de base de datos, no a un clúster de base de datos existente.
- Debe utilizar Percona XtraBackup para realizar copias de seguridad de sus datos en S3. Para obtener más información, consulte Instalación de Percona XtraBackup.
- El bucket de Amazon S3 y el clúster de base de datos de Aurora MySQL deben estar en la misma región de AWS.
- No puede restaurar desde lo siguiente:
  - Una exportación de instantáneas del clúster de base de datos a Amazon S3. Tampoco puede migrar datos desde una exportación de instantáneas del clúster de base de datos a su bucket de S3.
  - Una base de datos de origen cifrada, pero puede cifrar los datos que se están migrando. También puede dejar los datos sin cifrar durante el proceso de migración.
  - Una base de datos MySQL 5.5 o 5.6.
- Percona Server for MySQL no se admite como base de datos de origen, ya que puede contener tablas `compression_dictionary*` en el esquema `mysql`.
- No puede llevar a cabo la restauración en un clúster de base de datos de Aurora Serverless.
- No se admite la migración a versiones anteriores ni en las versiones principales ni en las secundarias. Por ejemplo, no puede migrar de la versión 8.0 de MySQL a la versión 2 de Aurora MySQL (compatible con MySQL 5.7) y no puede migrar de la versión 8.0.32 de MySQL a la versión 3.03 de Aurora MySQL, que es compatible con la versión de la comunidad 8.0.26 de MySQL.
- No se puede migrar a la versión 3.05 o posteriores de Aurora MySQL desde algunas versiones anteriores de MySQL 8.0, como 8.0.11, 8.0.13 y 8.0.15. Le recomendamos que actualice a la versión 8.0.28 de MySQL antes de realizar la migración.
- La importación de Amazon S3 no es compatible en la clase de instancia de base de datos db.t2.micro. Sin embargo, puede restaurar en otra clase de instancia de base de datos y, a continuación, cambiar la clase de instancia de base de datos posteriormente. Para obtener más información sobre las clases de instancias de bases de datos, consulte Clases de instancia de base de datos de Amazon Aurora.

- Amazon S3 limita el tamaño de un archivo cargado en un bucket de S3 a 5 TB. Si un archivo de copia de seguridad supera los 5 TB, debe dividir el archivo de copia de seguridad en archivos más pequeños.
- Amazon RDS limita el número de archivos cargados en un bucket de S3 a 1 millón. Si los datos de copia de seguridad de su base de datos, con todas las copias de seguridad completas e incrementales, superan 1 millón de archivos, use un archivo Gzip (.gz), tar (.tar.gz) o Percona xbstream (.xbstream) para almacenar archivos de copia de seguridad completas e incrementales en el bucket de S3. Percona XtraBackUp 8.0 solo admite Percona xbstream para la compresión.
- Para proporcionar servicios de administración para cada clúster de base de datos, se crea el usuario `rdsadmin` cuando se crea el clúster de base de datos. Como se trata de un usuario reservado en RDS, se aplican las siguientes limitaciones:
  - Las funciones, procedimientos, vistas, eventos y activadores con el definidor `'rdsadmin'@'localhost'` no se importan. Para obtener más información, consulte Objetos almacenados con 'rdsadmin'@'localhost' como definidor y Privilegios de la cuenta de usuario maestro con Amazon Aurora MySQL..
  - Cuando se crea el clúster de base de datos de Aurora MySQL, se crea un usuario maestro con los privilegios máximos admitidos. Al restaurar desde una copia de seguridad, los privilegios no admitidos asignados a los usuarios que se importan se eliminan automáticamente durante la importación.

    Para identificar a los usuarios que podrían verse afectados por esto, consulte Cuentas de usuario con privilegios no admitidos. Para obtener más información sobre los privilegios admitidos en Aurora MySQL, consulte Modelo de privilegios basado en roles.
- En el caso de la versión 3 de Aurora MySQL, no se importan los privilegios dinámicos. Los privilegios dinámicos compatibles con Aurora se pueden importar después de la migración. Para obtener más información, consulte Privilegios dinámicos de la versión 3 de Aurora MySQL.
- Las tablas creadas por el usuario en el esquema de `mysql` no se migran.
- El parámetro `innodb_data_file_path` debe configurarse con un solo archivo de datos que utilice el nombre de archivo de datos predeterminado `ibdata1:12M:autoextend`. Las bases de datos con dos archivos de datos, o con un archivo de datos con un nombre diferente, no se pueden migrar con este método.

  A continuación se muestran ejemplos de nombres de archivo no permitidos: `innodb_data_file_path=ibdata1:50M`, `ibdata2:50M:autoextend` y `innodb_data_file_path=ibdata01:50M:autoextend`.

- No puede migrar de una base de datos de origen que tenga tablas definidas fuera del directorio de datos de MySQL predeterminado.
- Actualmente, el tamaño máximo admitido para las copias de seguridad sin comprimir que utilizan este método está limitado a 64 TiB. En el caso de las copias de seguridad comprimidas, este límite se reduce para ajustarse a los requisitos de espacio de descompresión. En esos casos, el tamaño máximo admitido de la copia de seguridad sería (`64 TiB – compressed backup size`).
- Aurora MySQL no admite la importación de MySQL ni de otros componentes y complementos externos.
- Aurora MySQL no restaura todo lo que contiene la base de datos. Se recomienda guardar el esquema y los valores de la base de datos correspondientes para los siguientes elementos de la base de datos MySQL de origen y añadirlos al clúster de base de datos de Aurora MySQL restaurado una vez creado:
  - Cuentas de usuario
  - Funciones
  - Procedimientos almacenados
  - Información de zona horaria. Esta información se carga desde el sistema operativo local del clúster de base de datos de Aurora MySQL. Para obtener más información, consulte Zona horaria local para los clústeres de base de datos de Amazon Aurora.

Antes de empezar

Antes de copiar los datos en un bucket de Amazon S3 y restaurar a un clúster de base de datos a partir de esos archivos, debe hacer lo siguiente:

- Instale Percona XtraBackup en su servidor local.
- Permita que Aurora MySQL obtenga acceso a su bucket de Amazon S3 en su nombre.

Instalación de Percona XtraBackup

Amazon Aurora puede restaurar un clúster de base de datos a partir de archivos creados con Percona XtraBackup. Puede instalar Percona XtraBackup desde las descargas de software: Percona.

Para la migración de MySQL 5.7, utilice Percona XtraBackup 2.4.

Para la migración de MySQL 8.0, utilice Percona XtraBackup 8.0. Asegúrese de que la versión de Percona XtraBackup sea compatible con la versión del motor de su base de datos de origen.

Permisos necesarios

Para migrar los datos de MySQL a un clúster de base de datos de Amazon Aurora MySQL, se requieren varios permisos:

- El usuario que solicita que Aurora cree un nuevo clúster a partir de un bucket de Amazon S3 debe tener permiso para mostrar los buckets de su cuenta de AWS. Puede conceder este permiso al usuario mediante una política de AWS Identity and Access Management (IAM).
- Aurora requiere permiso para realizar acciones en su nombre y acceder al bucket de Amazon S3 en el que se almacenan los archivos utilizados para crear el clúster de base de datos de Amazon Aurora MySQL. Puede conceder los permisos necesarios a Aurora mediante un rol de servicio de IAM.
- El usuario que realiza la solicitud también debe tener permiso para enumerar los roles de IAM para la cuenta de AWS.
- Si el usuario que ha realizado la solicitud va a crear la función del servicio de IAM o va a solicitar que Aurora cree la función del servicio de IAM (mediante la consola), el usuario debe tener permiso para crear un rol de IAM para la cuenta de AWS.
- Si tiene previsto cifrar los datos durante el proceso de migración, actualice la política de IAM del usuario que va a realizar la migración para conceder a RDS acceso a las AWS KMS keys para cifrar los backups. Para obtener instrucciones, consulte Creación de una política de IAM para acceder a los recursos de AWS KMS.

Por ejemplo, la siguiente política de IAM concede a un usuario los permisos mínimos necesarios para que pueda utilizar la consola para mostrar los roles de IAM, crear un rol de IAM, mostrar los buckets de Amazon S3 de la cuenta y mostrar las claves de KMS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:CreatePolicy",
```

```
                "iam:AttachRolePolicy",
                "s3:ListBucket",
                "kms:ListKeys"
            ],
            "Resource": "*"
        }
    ]
}
```

Además, para que un usuario pueda asociar un rol de IAM a un bucket de Amazon S3, el usuario de IAM debe disponer del permiso `iam:PassRole` para ese rol de IAM. Este permiso hace que un administrador pueda restringir los roles de IAM que un usuario puede asociar a buckets de Amazon S3.

Por ejemplo, la siguiente política de IAM permite a un usuario asociar el rol denominado `S3Access` a un bucket de Amazon S3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::123456789012:role/S3Access"
        }
    ]
}
```

Para obtener más información sobre los permisos de usuario de IAM, consulte Administración de acceso mediante políticas.

Creación del rol de servicio de IAM

Puede hacer que la Consola de administración de AWS cree un rol para usted eligiendo la opción Create a New Role (Crear una función) (se explica más adelante en este tema). Si selecciona esta

opción y especifica un nombre para el nuevo rol, Aurora crea el rol de servicio de IAM necesario para que Aurora acceda al bucket de Amazon S3 con el nombre que usted indique.

Como alternativa, puede crear el rol manualmente completando el siguiente procedimiento.

Para crear un rol de IAM para que Aurora pueda acceder a Amazon S3

1. Realice los pasos que se indican en Creación de una política de IAM para acceder a los recursos de Amazon S3.
2. Realice los pasos que se indican en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.
3. Realice los pasos que se indican en Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.

Copia de seguridad de archivos para restaurarlos como un clúster de base de datos de Amazon Aurora MySQL

Puede crear un backup completo de los archivos de la base de datos MySQL usando Percona XtraBackup y cargar los archivos de backup en un bucket de Amazon S3. Como alternativa, si ya usa Percona XtraBackup para realizar los backups de los archivos de la base de datos MySQL, puede cargar los archivos y los directorios de backup completos e incrementales en un bucket de Amazon S3.

Temas

- Creación de una copia de seguridad completa con Percona XtraBackup
- Uso de copias de seguridad incrementales con Percona XtraBackup
- Consideraciones sobre copias de seguridad

Creación de una copia de seguridad completa con Percona XtraBackup

Para crear una copia de seguridad completa de los archivos de la base de datos MySQL que se puedan restaurar desde Amazon S3 para crear un clúster de base de datos de Aurora MySQL, use la utilidad Percona XtraBackup (`xtrabackup`) para realizar una copia de seguridad de la base de datos.

Por ejemplo, el siguiente comando crea un backup de una base de datos MySQL y almacena los archivos en la carpeta `/on-premises/s3-restore/backup`.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/
s3-restore/backup>
```

Si desea comprimir su backup en un solo archivo (que se puede dividir si es necesario), puede utilizar la opción `--stream` para guardar el backup en uno de los siguientes formatos:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

El siguiente comando crea una copia de seguridad de la base de datos MySQL dividido en varios archivos Gzip.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar.gz
```

El siguiente comando crea una copia de seguridad de la base de datos MySQL dividido en varios archivos tar.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar
```

El siguiente comando crea una copia de seguridad de la base de datos MySQL dividido en varios archivos xbstream.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.xbstream
```

> **Note**
>
> Si aparece el siguiente error, puede deberse a que se han mezclado formatos de archivo en el comando:
>
> ```
> ERROR:/bin/tar: This does not look like a tar archive
> ```

Una vez realizado el backup de la base de datos MySQL con la utilidad Percona XtraBackup, ya podrá copiar los directorios y los archivos del backup en un bucket de Amazon S3.

Para obtener más información acerca de cómo crear y cargar un archivo en un bucket de Amazon S3, consulte Introducción a Amazon Simple Storage Service en la Guía de introducción a Amazon S3.

Uso de copias de seguridad incrementales con Percona XtraBackup

Amazon Aurora MySQL admite backups completos e incrementales creados con Percona XtraBackup. Si ya usa Percona XtraBackup para realizar copias de seguridad completas e incrementales de sus archivos de base de datos MySQL, no tiene que crear una copia de seguridad completa y cargar los archivos del backup en Amazon S3. En lugar de eso, puede ahorrar una cantidad considerable de tiempo copiando los directorios y archivos de backup de sus backups completos e incrementales en un bucket de Amazon S3. Para obtener más información, consulte el artículo sobre creación de una copia de seguridad incremental en el sitio web de Percona.

Cuando copie los archivos de backup completos o incrementales en un bucket de Amazon S3, debe copiar repetidamente el contenido del directorio base. Esos contenidos incluyen el backup completo y también todos los directorios y archivos del backup incremental. Esta copia debe mantener la estructura de directorios en el bucket de Amazon S3. Aurora realiza iteraciones por todos los archivos y directorios. Aurora utiliza el archivo `xtrabackup-checkpoints` incluido con cada copia de seguridad progresiva para identificar el directorio base y ordenar las copias de seguridad progresivas por rango del número de secuencia del registro (LSN).

Para obtener más información acerca de cómo crear y cargar un archivo en un bucket de Amazon S3, consulte Introducción a Amazon Simple Storage Service en la Guía de introducción a Amazon S3.

Consideraciones sobre copias de seguridad

Aurora no admite copias de seguridad parciales creados con Percona XtraBackup. No puede utilizar las siguientes opciones para crear una copia de seguridad parcial al realizar copias de seguridad de los archivos de origen de su base de datos: `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude` o `--databases-file`.

Para obtener más información acerca de cómo realizar una copia de seguridad de su base de datos con Percona XtraBackup, consulte la documentación de Percona XtraBackup y el artículo sobre el uso de registros binarios en el sitio web de Percona.

Aurora admite copias de seguridad incrementales creadas con Percona XtraBackup. Para obtener más información, consulte el artículo sobre creación de una copia de seguridad incremental en el sitio web de Percona.

Aurora consume sus archivos de copia de seguridad en función del nombre de archivo. Asegúrese de asignar la extensión de archivo adecuada a los archivos de copia de seguridad según el formato de archivo: por ejemplo, `.xbstream` para archivos almacenados con el formato xbstream de Percona.

Aurora consume sus archivos de backup en orden alfabético, así como según la numeración natural. Utilice siempre la opción `split` al ejecutar el comando `xtrabackup` para asegurarse de que la escritura y la asignación de nombre de sus archivos de backup se realice en el orden correcto.

Amazon S3 limita el tamaño de un archivo cargado en un bucket de Amazon S3 a 5 TB. Si los datos del backup de su base de datos superan los 5 TB, use el comando `split` para dividir los archivos de backup en varios archivos que ocupen menos de 5 TB.

Aurora limita el número de archivos de origen cargados en un bucket de Amazon S3 a 1 millón de archivos. En algunos casos, los datos de backup de su base de datos con todos los backups completos e incrementales incluyen un número elevado de archivos. En estos casos, usa un archivo tarball (.tar.gz) para almacenar los archivos de backups completos e incrementales en el bucket de Amazon S3.

Cuando carga un archivo en un bucket de Amazon S3, puede usar el cifrado del lado del servidor para cifrar los datos. Luego, puede restaurar un clúster de base de datos de Amazon Aurora MySQL a partir de estos archivos cifrados. Amazon Aurora MySQL puede restaurar un clúster de base de datos con archivos cifrados mediante los siguientes tipos de cifrado del lado del servidor:

- Cifrado en el servidor con claves de cifrado administradas por Amazon S3 (SSE-S3): cada objeto está cifrado con una clave exclusiva que utiliza un cifrado multifactor seguro.
- Cifrado en el servidor con claves administradas por AWS KMS (SSE-KMS): similar a SSE-S3, pero tiene la opción de crear y administrar usted mismo las claves de cifrado, así como otras diferencias.

Para obtener información sobre cómo usar el cifrado en el servidor al cargar archivos en un bucket de Amazon S3, consulte Protección de datos con el cifrado del lado del servidor en la Guía para desarrolladores de Amazon S3.

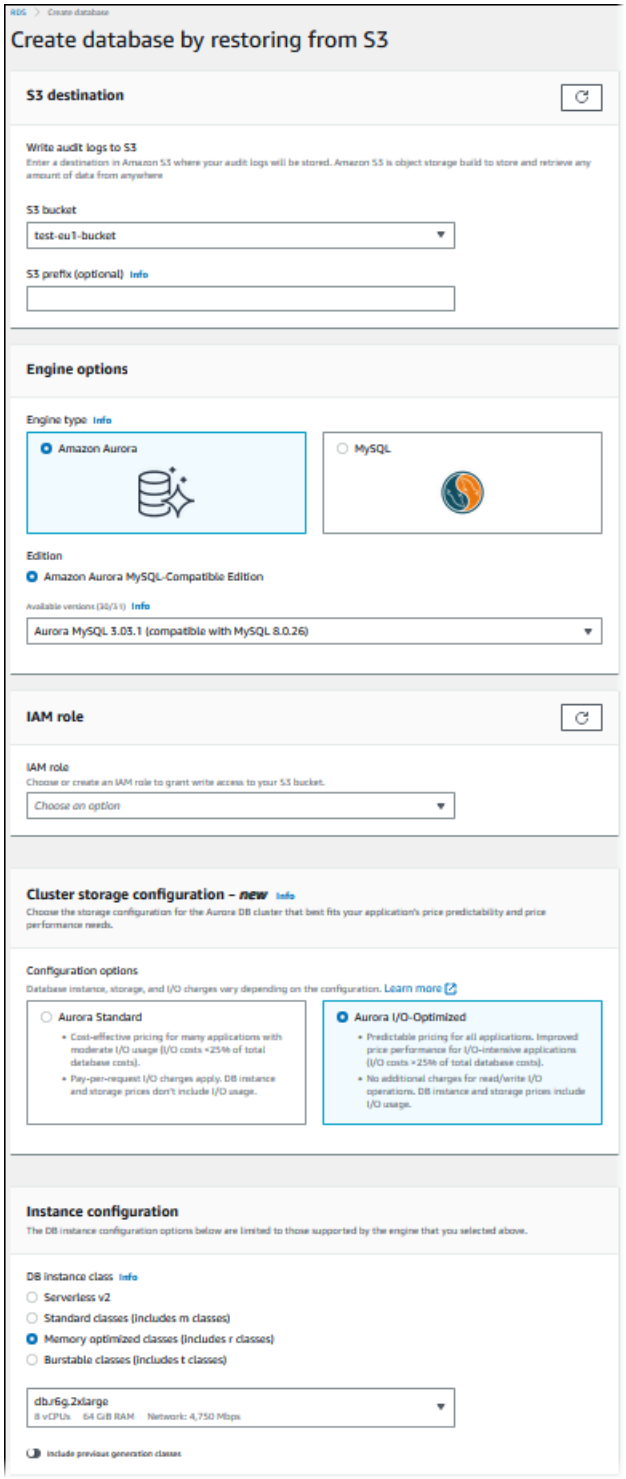
Restauración de un clúster de base de datos de Amazon Aurora MySQL desde un bucket de Amazon S3

Puede restaurar los archivos de copia de seguridad desde el bucket de Amazon S3 para crear un nuevo clúster de base de datos de Amazon Aurora MySQL mediante la consola de Amazon RDS.

Para restaurar un clúster de base de datos de Amazon Aurora MySQL a partir de los archivos de un bucket de Amazon S3

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la consola de Amazon RDS, elija la región de AWS en la que se va a crear su clúster de base de datos. Elija la misma región de AWS que el bucket de Amazon S3 que contiene su copia de seguridad de base de datos.
3. En el panel de navegación, elija Databases (Bases de datos) y después Restore from S3 (Restaurar desde S3).
4. Elija Restore from S3 (Restaurar de S3).

   Aparecerá la página Create database by restoring from S3 (Crear base de datos restaurando desde S3).



5. En S3 destination (destino de S3):

   a. Elija el bucket de S3 que contiene sus archivos de copia de seguridad.

   b. (Opcional) En S3 folder path prefix (Prefijo de ruta de carpeta de S3), escriba un prefijo de ruta de archivo para los archivos almacenados en el bucket de Amazon S3.

Si no especifica un prefijo, RDS creará su instancia de base de datos con todos los archivos y carpetas de la carpeta raíz del bucket de S3. Si especifica un prefijo, RDS creará su instancia de base de datos con los archivos y carpetas del bucket de S3 cuya ruta completa del archivo empieza con el prefijo especificado.

Por ejemplo, suponga que almacena los archivos de backup en S3 en una subcarpeta denominada copias de seguridad y que tiene varios conjuntos de archivos de backup, cada uno en su propio directorio (gzip_backup1, gzip_backup2, etc.). En este caso, debe especificar un prefijo de backups/gzip_backup1 para restaurar a partir de los archivos de la carpeta gzip_backup1.

6. En Engine options (Opciones del motor):

   a. Para Engine type (Tipo de motor), elija Amazon Aurora.

   b. En Version (Versión), elija la versión del motor Aurora MySQL para la instancia de base de datos restaurada.

7. En IAM Role (Rol de IAM), puede elegir un rol de IAM existente.

8. (Opcional) También puede hacer que se cree un nuevo rol de IAM eligiendo Create a new role (Crear un nuevo rol). Si es así:

   a. Introduzca elIAM role name (Nombre de rol de IAM).

   b. Elija si desea Allow access to KMS key (Permitir el acceso a la clave KMS):

      - Si no ha cifrado los archivos de copia de seguridad, elija No.
      - Si ha cifrado los archivos de copia de seguridad con AES-256 (SSE-S3) al cargarlos en Amazon S3, elija No. En este caso, los datos se encriptan de manera automática.
      - Si ha cifrado los archivos de copia de seguridad con cifrado en el servidor de AWS KMS (SSE-KMS) al cargarlos en Amazon S3, elija Yes (Sí). A continuación, elija la clave de KMS correcta para AWS KMS key.

        La Consola de administración de AWS crea una política de IAM que permite a Aurora descifrar los datos.

   Para obtener más información, consulte Protección de datos con el cifrado del lado del servidor en la Guía para desarrolladores de Amazon S3.

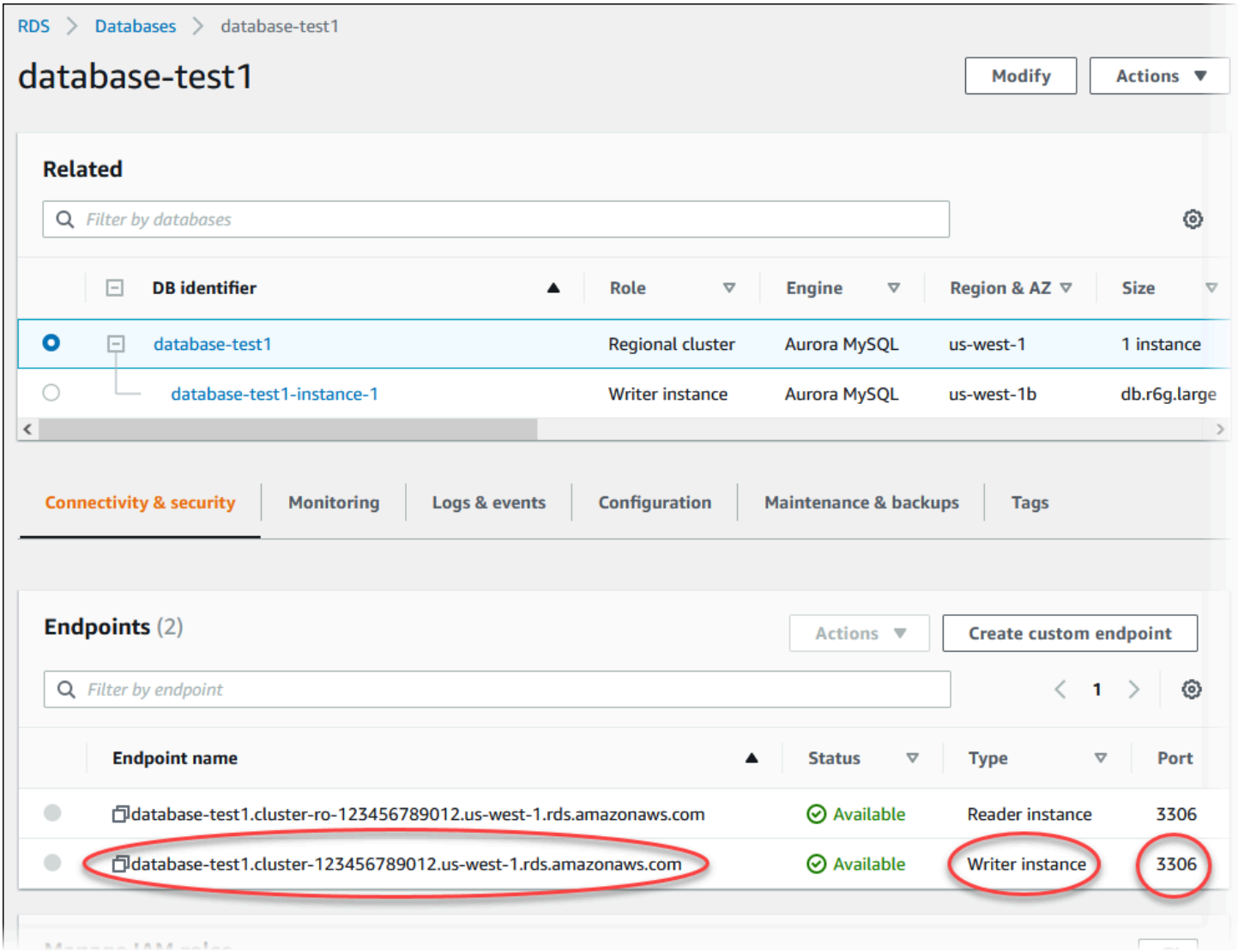
9. Elija la configuración del clúster de base de datos, como la configuración de almacenamiento del clúster de base de datos, la clase de instancia de base de datos, el identificador de clúster de

base de datos y las credenciales de inicio de sesión. Para obtener más información acerca de cada ajuste, consulte Configuración de clústeres de bases de datos de Aurora.

10. Personalice la configuración adicional del clúster de base de datos Aurora MySQL según sea necesario.
11. Elija Create database (Crear base de datos) para iniciar la instancia de base de datos Aurora.

En la consola de Amazon RDS, la nueva instancia de base de datos aparece en la lista de instancias de base de datos. La instancia de la base de datos tendrá el estado creating hasta que se cree la instancia y esté lista para el uso. Cuando el estado cambie a available, podrá conectarse a la instancia principal de su clúster de base de datos. Dependiendo de la clase de instancia de base de datos y del almacenamiento asignado, es posible que la nueva instancia tarde varios minutos en estar disponible.

Para ver el clúster que acaba de crear, elija la vista Databases (Bases de datos) en la consola de Amazon RDS y elija el clúster de base de datos. Para obtener más información, consulte Visualización de un clúster de base de datos de Amazon Aurora.



Anote el puerto y el punto de enlace del escritor del clúster de base de datos. Utilice el punto de enlace del escritor y el puerto del clúster de base de datos en sus cadenas de conexión JDBC y ODBC para cualquier aplicación que realice operaciones de lectura o escritura.

Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL mediante replicación

Para que no se produzcan interrupciones durante la migración, puede replicar las transacciones que se confirmaron en su base de datos MySQL en el clúster de base de datos de Aurora MySQL. La replicación permite al clúster de base de datos ponerse al día con las transacciones de la base de datos MySQL que se produjeron durante la migración. Cuando el clúster de base de datos esté totalmente sincronizado, puede detener la replicación y terminar la migración a Aurora MySQL.

Temas

- Configuración de la base de datos MySQL externa y el clúster de base de datos de Aurora MySQL para la replicación cifrada
- Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL externa

Configuración de la base de datos MySQL externa y el clúster de base de datos de Aurora MySQL para la replicación cifrada

Para replicar los datos de forma segura, puede usar la replicación cifrada.

> **Note**
>
> Si no necesita usar la replicación cifrada, puede omitir estos pasos y pasar a las instrucciones de Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL externa.

A continuación, se indican los requisitos previos para utilizar la replicación cifrada:

- La capa de conexión segura (SSL) debe estar habilitada en la base de datos principal de MySQL externa.
- Debe disponerse de una clave cliente y un certificado cliente para el clúster de base de datos de Aurora MySQL.

Durante la replicación cifrada, el clúster de base de datos Aurora MySQL actúa como un cliente en el servidor de base de datos MySQL. Los certificados y las claves del cliente de Aurora MySQL son archivos con formato .pem.

Para configurar su base de datos MySQL externa y su clúster de base de datos de Aurora MySQL para la replicación cifrada

1. Compruebe que esté preparado para la replicación cifrada:
   - Si no tiene SSL habilitado en la base de datos principal de MySQL externa y no dispone de una clave cliente ni de un certificado cliente, habilite SSL en el servidor de base de datos de MySQL y genere la clave cliente y el certificado cliente necesarios.
   - Si SSL está habilitado en el primario externo, proporcione una clave cliente y un certificado cliente para el clúster de base de datos de Aurora MySQL. Si no los tiene, genere una nueva

clave y certificado para el clúster de base de datos Aurora MySQL. Para firmar el certificado cliente, debe tener la clave de la entidad de certificación que usó para configurar SSL en la base de datos primaria de MySQL externa.

Para obtener más información, consulte Creating SSL Certificates and Keys Using openssl en la documentación de MySQL.

Necesita un certificado de la entidad de certificación, la clave cliente y el certificado cliente.

2. Conéctese al clúster de base de datos Aurora MySQL como usuario principal mediante SSL.

   Para obtener información acerca de la conexión a un clúster de base de datos Aurora MySQL con SSL, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

3. Ejecute el procedimiento almacenado mysql.rds_import_binlog_ssl_material para importar la información de SSL en el clúster de base de datos Aurora MySQL.

   Para el parámetro `ssl_material_value`, inserte la información de los archivos con formato .pem para el clúster de base de datos Aurora MySQL en la carga JSON correcta.

   En el siguiente ejemplo se importa la información de SSL en un clúster de base de datos Aurora MySQL. En los archivos con formato .pem, el código del cuerpo suele ser más grande que el que se muestra en el ejemplo.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
```

```
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Para obtener más información, consulte mysql.rds_import_binlog_ssl_material y Conexiones TLS a clústeres de base de datos de Aurora MySQL.

> **Note**
>
> Después de ejecutar el procedimiento, los secretos se almacenan en archivos. Para borrar los archivos más tarde, puede ejecutar el procedimiento almacenado mysql.rds_remove_binlog_ssl_material.

Sincronización del clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL externa

Puede sincronizar su clúster de base de datos de Amazon Aurora MySQL con la base de datos MySQL mediante replicación.

Para sincronizar su clúster de base de datos de Aurora MySQL con la base de datos MySQL mediante replicación

1. Asegúrese de que el archivo /etc/my.cnf de la base de datos MySQL externa tenga las entradas pertinentes.

   Si no se requiere la replicación cifrada, asegúrese de que la base de datos MySQL externa se inicia con los logs binarios (binlogs) habilitados y SSL deshabilitado. A continuación se indican las entradas pertinentes en el archivo /etc/my.cnf para los datos sin cifrar.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

   Si se requiere la replicación cifrada, asegúrese de que la base de datos MySQL externa se inicia con SSL y los binlogs habilitados. Las entradas del archivo /etc/my.cnf incluyen las ubicaciones del archivo .pem para el servidor de base de datos MySQL.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1

# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

Puede confirmar que SSL está habilitado con el siguiente comando.

```
mysql> show variables like 'have_ssl';
```

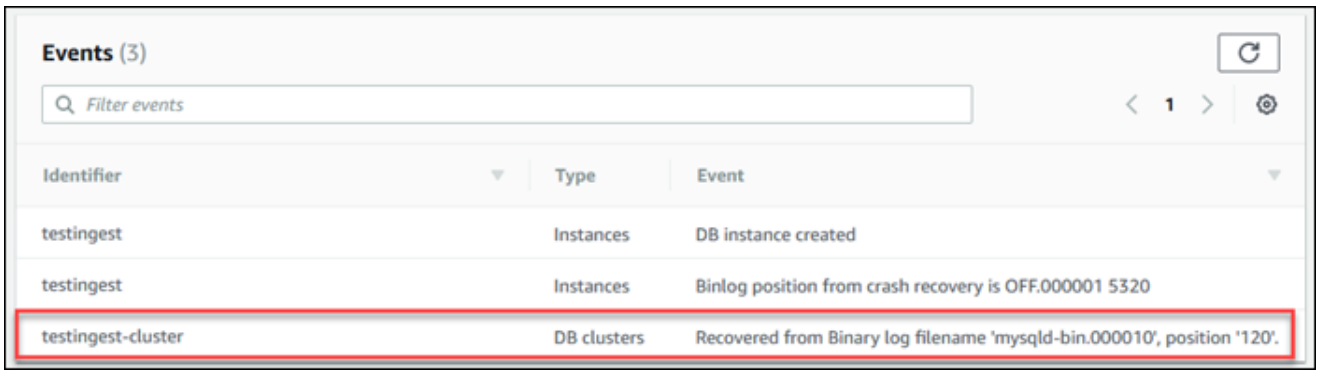
El resultado debería ser similar al siguiente.

```
+~-~-~-~-~-~-~-~-~-~-~-~-~-~--+~-~-~-~-~-~--+
| Variable_name | Value |
+~-~-~-~-~-~-~-~-~-~-~-~-~-~--+~-~-~-~-~-~--+
| have_ssl      | YES   |
+~-~-~-~-~-~-~-~-~-~-~-~-~-~--+~-~-~-~-~-~--+
1 row in set (0.00 sec)
```

2. Determine la posición del registro binario inicial para la replicación: Especificará la posición para iniciar la replicación en un paso posterior.

   Uso de Consola de administración de AWS

   a. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
   b. En el panel de navegación, seleccione Events.
   c. En la lista Events (Eventos), anote la posición del evento Recovered from Binary log filename (Recuperado del nombre de archivo de log binario).



Uso de AWS CLI

También puede obtener el nombre y la posición del archivo binlog utilizando el comando describe-events desde la AWS CLI. El siguiente es un ejemplo del comando `describe-events`.

```
PROMPT> aws rds describe-events
```

En la salida, identifique el evento que muestra la posición de binlog.

3. Mientras está conectado a la base de datos MySQL externa, cree el usuario que se va a usar para la replicación. Esta cuenta se usa únicamente para la replicación y debe estar limitada a su dominio para mejorar la seguridad. A continuación se muestra un ejemplo.

```
mysql> CREATE USER '<user_name>'@'<domain_name>' IDENTIFIED BY '<password>';
```

El usuario requiere los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE`. Conceda estos privilegios al usuario.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO
 '<user_name>'@'<domain_name>';
```

Si necesita usar la replicación cifrada, exija conexiones SSL al usuario de la replicación. Por ejemplo, puede utilizar la siguiente declaración para exigir el uso de conexiones SSL en la cuenta de usuario *<user_name>*.

```
GRANT USAGE ON *.* TO '<user_name>'@'<domain_name>' REQUIRE SSL;
```

> **Note**
>
> Si `REQUIRE SSL` no se incluye, la conexión de replicación podría cambiarse inadvertidamente por una conexión sin cifrar.

4. En la consola de Amazon RDS, agregue la dirección IP del servidor que aloja la base de datos MySQL externa al grupo de seguridad de la VPC para el clúster de base de datos de Aurora MySQL. Para obtener más información acerca de la modificación de un grupo de seguridad de VPC, consulte Grupos de seguridad de su VPC en la Guía del usuario de Amazon Virtual Private Cloud.

   Es posible que también necesite configurar su red local para permitir las conexiones desde la dirección IP de su clúster de base de datos de Aurora MySQL con el fin de que se pueda comunicar con la base de datos MySQL externa. Para encontrar la dirección IP del clúster de base de datos de Aurora MySQL, use el comando `host`.

   ```
   host <db_cluster_endpoint>
   ```

   El nombre de host es el nombre DNS del punto de enlace del clúster de base de datos de Aurora MySQL.

5. Habilite la replicación de registros binarios ejecutando el procedimiento almacenado mysql.rds_reset_external_master (Aurora MySQL versión 2) o mysql.rds_reset_external_source (Aurora MySQL versión 3). Este procedimiento almacenado tiene la siguiente sintaxis.

   ```
   CALL mysql.rds_set_external_master (
     host_name
     , host_port
     , replication_user_name
     , replication_user_password
     , mysql_binary_log_file_name
     , mysql_binary_log_file_location
     , ssl_encryption
   );

   CALL mysql.rds_set_external_source (
     host_name
   ```

```
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Para obtener más información acerca de los parámetros, consulte mysql.rds_reset_external_master (Aurora MySQL versión 2) y mysql.rds_reset_external_source (Aurora MySQL versión 3).

Para `mysql_binary_log_file_name` y `mysql_binary_log_file_location`, use la posición del evento Recovered from Binary log filename (Recuperado del nombre de archivo de log binario) que anotó antes.

Si los datos del clúster de base de datos de Aurora MySQL no están cifrados, el parámetro `ssl_encryption` debe establecerse en `0`. Si los datos están cifrados, el parámetro `ssl_encryption` debe establecerse en `1`.

El siguiente ejemplo ejecuta el procedimiento para un clúster de base de datos de Aurora MySQL que tiene datos cifrados.

```
CALL mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);

CALL mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);
```

Este procedimiento almacenado establece los parámetros que el clúster de base de datos de Aurora MySQL usa para conectarse a una base de datos MySQL externa y leer su log binario. Si los datos están cifrados, también descarga el certificado de la entidad de certificación de SSL, el certificado cliente y la clave cliente en el disco local.

6. Inicie la replicación de logs binarios ejecutando el procedimiento almacenado mysql.rds_start_replication.

```
CALL mysql.rds_start_replication;
```

7. Monitorice qué retardo tiene el clúster de base de datos de Aurora MySQL con respecto a la base de datos principal de replicación de MySQL. Para ello, conéctese al clúster de base de datos de Aurora MySQL y ejecute el siguiente comando.

```
Aurora MySQL version 2:
SHOW SLAVE STATUS;

Aurora MySQL version 3:
SHOW REPLICA STATUS;
```

   En la salida del comando, el campo `Seconds Behind Master` muestra cuánto retardo tiene el clúster de base de datos de Aurora MySQL con respecto al MySQL principal. Cuando este valor es `0` (cero), el clúster de base de datos de Aurora MySQL se ha sincronizado con el principal y puede continuar con el siguiente paso para detener la replicación.

8. Conéctese a la base de datos principal de replicación MySQL y detenga la replicación. Para ello, ejecute el procedimiento almacenado mysql.rds_stop_replication.

```
CALL mysql.rds_stop_replication;
```

Reducir el tiempo de la migración física a Amazon Aurora MySQL

Puede realizar las siguientes modificaciones en la base de datos de para acelerar el proceso de migrar una base de datos a Amazon Aurora MySQL.

> **Important**
>
> Asegúrese de realizar las actualizaciones en una copia de la base de datos de producción, en lugar de en una base de datos de producción. A continuación, puede hacer una copia de seguridad de la copia y restaurarla en su clúster de base de datos de Aurora MySQL para evitar interrupciones en el servicio en la base de datos de producción.

Tipos de tablas no admitidas

Aurora MySQL solo admite el motor InnoDB para tablas de bases de datos. Si hay tablas MyISAM en la base de datos, tendrá que convertir esas tables antes de migrar a Aurora MySQL. El proceso de conversión requiere más espacio para la conversión de MyISAM a InnoDB durante el procedimiento de migración.

Para reducir el riesgo de quedarse sin espacio o para acelerar el proceso de migración, convierta todas sus tablas MyISAM en tablas InnoDB antes de migrarlas. El tamaño de la tabla InnoDB resultante equivale al tamaño requerido por Aurora MySQL para esa tabla. Para convertir una tabla MyISAM a InnoDB, ejecute el siguiente comando:

```
ALTER TABLE schema.table_name engine=innodb, algorithm=copy;
```

Aurora MySQL no admite tablas comprimidas, es decir, tablas creadas con `ROW_FORMAT=COMPRESSED` o `COMPRESSION = {"zlib"|"lz4"}`.

Para reducir el riesgo de quedarse sin espacio o para acelerar el proceso de migración, amplíe las tablas comprimidas mediante la configuración de `ROW_FORMAT DEFAULT`, `COMPACT`, `DYNAMIC` o `REDUNDANT`. Para páginas comprimidas, configure `COMPRESSION="none"`.

Para obtener más información, consulte Formatos de fila de InnoDB y Compresión de tablas y páginas en InnoDB en la documentación de MySQL.

Puede utilizar el siguiente script de SQL en su instancia de base de datos MySQL para obtener una lista de las tablas MyISAM o comprimidas de su base de datos.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Aurora MySQL.
-- It must be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.
```

```
-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
```

```
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Cuentas de usuario con privilegios no admitidos

Las cuentas de usuario con privilegios no admitidos con Aurora MySQL se importan sin esos privilegios. Para ver la lista de los privilegios admitidos, consulte Modelo de privilegios basado en roles.

Puede ejecutar la siguiente consulta SQL en la base de datos de origen para enumerar las cuentas de usuario que tienen privilegios no admitidos.

```
SELECT
    user,
    host
FROM
    mysql.user
WHERE
    Shutdown_priv = 'y'
    OR File_priv = 'y'
    OR Super_priv = 'y'
    OR Create_tablespace_priv = 'y';
```

Privilegios dinámicos de la versión 3 de Aurora MySQL

Los privilegios dinámicos no se importan. La versión 3 de Aurora MySQL admite los siguientes privilegios dinámicos.

```
'APPLICATION_PASSWORD_ADMIN',
'CONNECTION_ADMIN',
'REPLICATION_APPLIER',
'ROLE_ADMIN',
'SESSION_VARIABLES_ADMIN',
'SET_USER_ID',
'XA_RECOVER_ADMIN'
```

El siguiente script de ejemplo concede los privilegios dinámicos admitidos a las cuentas de usuario del clúster de base de datos de Aurora MySQL.

```
-- This script finds the user accounts that have Aurora MySQL supported dynamic
 privileges
-- and grants them to corresponding user accounts in the Aurora MySQL DB cluster.

/home/ec2-user/opt/mysql/8.0.26/bin/mysql -uusername -pxxxxx -P8026 -h127.0.0.1 -BNe
 "SELECT
  CONCAT('GRANT ', GRANTS, ' ON *.* TO ', GRANTEE ,';') AS grant_statement
  FROM (select GRANTEE, group_concat(privilege_type) AS GRANTS FROM
 information_schema.user_privileges
      WHERE privilege_type IN (
        'APPLICATION_PASSWORD_ADMIN',
        'CONNECTION_ADMIN',
        'REPLICATION_APPLIER',
        'ROLE_ADMIN',
        'SESSION_VARIABLES_ADMIN',
        'SET_USER_ID',
        'XA_RECOVER_ADMIN')
      AND GRANTEE NOT IN (\"'mysql.session'@'localhost'\",
\"'mysql.infoschema'@'localhost'\",\"'mysql.sys'@'localhost'\") GROUP BY GRANTEE)
      AS PRIVGRANTS; " | /home/ec2-user/opt/mysql/8.0.26/bin/mysql -u master_username -
p master_password -h DB_cluster_endpoint
```

Objetos almacenados con 'rdsadmin'@'localhost' como definidor

Las funciones, procedimientos, vistas, eventos y activadores con `'rdsadmin'@'localhost'` como definidor no se importan.

Puede utilizar el siguiente script de SQL en la base de datos MySQL de origen para enumerar los objetos almacenados que tienen el definidor no admitido.

```
-- This SQL query lists routines with `rdsadmin`@`localhost` as the definer.

SELECT
    ROUTINE_SCHEMA,
    ROUTINE_NAME
FROM
    information_schema.routines
WHERE
    definer = 'rdsadmin@localhost';

-- This SQL query lists triggers with `rdsadmin`@`localhost` as the definer.

SELECT
```

```
    TRIGGER_SCHEMA,
    TRIGGER_NAME,
    DEFINER
FROM
    information_schema.triggers
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists events with `rdsadmin`@`localhost` as the definer.

SELECT
    EVENT_SCHEMA,
    EVENT_NAME
FROM
    information_schema.events
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists views with `rdsadmin`@`localhost` as the definer.
SELECT
    TABLE_SCHEMA,
    TABLE_NAME
FROM
    information_schema.views
WHERE
    DEFINER = 'rdsadmin@localhost';
```

Migración lógica de MySQL a Amazon Aurora MySQL mediante mysqldump

Dado que Amazon Aurora MySQL es una base de datos compatible con MySQL, puede usar la utilidad `mysqldump` para copiar datos de la base de datos MySQL o la utilidad `mariadb-dump` para copiar los datos de la base de datos MariaDB en un clúster de base de datos de Aurora MySQL existente.

Para consultar un análisis sobre cómo hacerlo con bases de datos de MySQL o MariaDB de tamaño muy grande, consulte los siguientes temas en la Guía del usuario de Amazon Relational Database Service:

- MySQL: importación de datos a una base de datos de Amazon RDS para MySQL con un tiempo de inactividad reducido
- MariaDB: importación de datos a una base de datos de Amazon RDS para MariaDB con un tiempo de inactividad reducido

Para las bases de datos de MySQL o MariaDB que tienen pequeñas cantidades de datos, consulte los siguientes temas en la Guía del usuario de Amazon Relational Database Service:

- MySQL: importación de datos de una base de datos MySQL externa a una instancia de base de datos de Amazon RDS para MySQL
- MariaDB: importación de datos de una base de datos de MariaDB externa a una instancia de base de datos de Amazon RDS para MariaDB

Migración de datos de una instancia de base de datos de RDS for MySQL a un clúster de base de datos de Amazon Aurora MySQL

Puede migrar (copiar) los datos a un clúster de base de datos de Amazon Aurora MySQL desde una instancia de base de datos de RDS for MySQL.

Temas

- Migración de una instantánea de RDS for MySQL a Aurora
- Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora

> **Note**
>
> Dado que Amazon Aurora MySQL es compatible con MySQL, puede migrar datos desde la base de datos MySQL configurando la replicación entre la base de datos MySQL y un clúster de base de datos de Amazon Aurora MySQL. Para obtener más información, consulte Replicación con Amazon Aurora.

Migración de una instantánea de RDS for MySQL a Aurora

Puede migrar una instantánea de base de datos de una instancia de base de datos de RDS for MySQL para crear un clúster de base de datos de MySQL Aurora. El nuevo clúster de base de datos Aurora MySQL se rellena con los datos de la instancia de base de datos de RDS for MySQL original. La instantánea de base de datos debe haberse obtenido a partir de una instancia de base de datos de Amazon RDS que ejecute una versión de MySQL compatible con Aurora MySQL.

Puede migrar una instantánea de base de datos manual o automatizada. Una vez creado el clúster de base de datos, podrá crear réplicas de Aurora opcionales.

> **Note**
>
> También puede migrar una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Aurora MySQL. Para ello, debe crear una réplica de lectura de Aurora de su instancia de base de datos de RDS para MySQL de origen. Para obtener más información, consulte Migración de datos desde una instancia de base de datos de RDS para

MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora.
No se puede migrar a la versión 3.05 o posteriores de Aurora MySQL desde algunas versiones anteriores de MySQL 8.0, como 8.0.11, 8.0.13 y 8.0.15. Le recomendamos que actualice a la versión 8.0.28 de MySQL antes de realizar la migración.

Los pasos generales que debe realizar son los siguientes:

1. Determine la cantidad de espacio que desea aprovisionar para el clúster de base de datos de Aurora MySQL. Para obtener más información, consulte ¿Cuánto espacio necesito?
2. Utilice la consola para crear la instantánea en la región de AWS en la que se encuentra la instancia de Amazon RDS MySQL. Para obtener más información acerca de la creación de una instantánea de base de datos, consulte Creación de una instantánea de base de datos.
3. Si la instantánea de base de datos no se encuentra en la misma región de AWS que su clúster de base de datos, utilice la consola de Amazon RDS; para copiar la instantánea de base de datos en esa región de AWS. Para obtener más información acerca de la copia de una instantánea de base de datos, consulte Copia de una instantánea de base de datos.
4. Utilice la consola para migrar la instantánea de base de datos y crear un clúster de base de datos de Aurora MySQL con las mismas bases de datos que la instancia de base de datos original de MySQL.

Warning

Amazon RDS limita cada cuenta de AWS a una copia de la instantánea en cada región de AWS en un momento dado.

¿Cuánto espacio necesito?

Al migrar una instantánea de una instancia de base de datos MySQL a un clúster de base de datos de Aurora MySQL, Aurora utiliza un volumen de Amazon Elastic Block Store (Amazon EBS) para formatear los datos de la instantánea antes de migrarlos. En algunos casos, se necesita espacio adicional para formatear los datos para la migración.

Las tablas que no son MyISAM y no están comprimidas pueden alcanzar un tamaño de 16 TB. Si tiene tablas MyISAM, Aurora deberá utilizar espacio adicional en el volumen para convertir las tablas

con el fin de que sean compatibles con MySQL de Aurora. Si hay tablas comprimidas, Aurora tendrá que utilizar espacio adicional en el volumen para ampliar esas tablas antes de almacenarlas en el volumen del clúster de Aurora. Debido a este requisito de espacio adicional, debe asegurarse de que ninguna de las tablas MyISAM y de las tablas comprimidas que se van a migrar desde su instancia de base de datos MySQL tiene un tamaño superior a 8 TB.

Reducción de la cantidad de espacio necesario para migrar datos a Amazon Aurora MySQL

Es posible que le interese modificar su esquema de base de datos antes de migrar a Amazon Aurora. Esta modificación puede ser útil en los siguientes casos:

- Si desea acelerar el proceso de migración.
- Si no está seguro de cuánto espacio necesita aprovisionar.
- Si ha intentado migrar los datos y la migración ha generado un error por falta de espacio aprovisionado.

Puede realizar los siguientes cambios para mejorar el proceso de migración de una base de datos a Amazon Aurora.

> **Important**
>
> Asegúrese de realizar estas actualizaciones en una instancia de base de datos nueva restaurada a partir de un snapshot de una base de datos de producción y no a partir de una instancia de producción. A continuación, puede migrar los datos de la instantánea de la nueva instancia de base de datos al clúster de base de datos de Aurora para evitar las interrupciones de servicio en la base de datos de producción.

Tipo de tabla	Limitación o directriz
Tablas MyISAM	Aurora MySQL solo admite tablas InnoDB. Si hay tablas MyISAM en la base de datos, tendrá que convertirlas antes de migrarlas a Aurora MySQL. El proceso de conversión requiere más espacio para la conversión de MyISAM a InnoDB durante el procedimiento de migración. Para reducir el riesgo de quedarse sin espacio o para acelerar el proceso de migración, convierta todas sus tablas MyISAM en

Tipo de tabla	Limitación o directriz
	tablas InnoDB antes de migrarlas. El tamaño de la tabla InnoDB resultante equivale al tamaño requerido por Aurora MySQL para esa tabla. Para convertir una tabla MyISAM a InnoDB, ejecute el siguiente comando: `alter table <schema>.<table_name> engine=innodb, algorithm=copy;`
Tablas comprimidas	Aurora MySQL no admite tablas comprimidas (es decir, tablas creadas con `ROW_FORMAT=COMPRESSED`). Para reducir el riesgo de quedarse sin espacio o para acelerar el proceso de migración, amplíe las tablas comprimidas mediante la configuración de `ROW_FORMAT` `DEFAULT`, `COMPACT`, `DYNAMIC` o `REDUNDANT`. Para obtener más información, consulte InnoDB row formats (Formatos de fila de InnoDB) en la documentación de MySQL.

Puede utilizar el siguiente script de SQL en su instancia de base de datos MySQL para obtener una lista de las tablas MyISAM o comprimidas de su base de datos.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Amazon Aurora.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
```

```
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

El script genera una salida similar a la del siguiente ejemplo. El ejemplo muestra dos tablas que se deben convertir de MyISAM a InnoDB. La salida también incluye el tamaño aproximado de cada tabla en megabytes (MB).

```
+---------------------------------+------------------+
| ==> MyISAM or Compressed Tables | Approx size (MB) |
```

```
+---------------------------------+------------------+
| test.name_table                 |          2102.25 |
| test.my_table                   |            65.25 |
+---------------------------------+------------------+
2 rows in set (0.01 sec)
```

Migración de una instantánea de base de datos RDS para MySQL a un clúster de base de datos Aurora MySQL

Puede migrar una instantánea de base de datos de una instancia de base de datos de RDS para MySQL a fin de crear un clúster de base de datos de Aurora MySQL mediante la Consola de administración de AWS o la AWS CLI. El nuevo clúster de base de datos Aurora MySQL se rellena con los datos de la instancia de base de datos de RDS for MySQL original. Para obtener más información acerca de la creación de una instantánea de base de datos, consulte Creación de una instantánea de base de datos.

Si la instantánea de base de datos no se encuentra en la región de AWS en la que desea ubicar sus datos, copie la instantánea de base de datos en dicha región de AWS. Para obtener más información acerca de la copia de una instantánea de base de datos, consulte Copia de una instantánea de base de datos.

Consola

Al migrar la instantánea de base de datos con la Consola de administración de AWS, esta realiza las acciones necesarias para crear solo el clúster de base de datos.

También puede elegir que el nuevo clúster de base de datos de Aurora MySQL se cifre en reposo mediante una AWS KMS key.

Para migrar una instantánea de base de datos MySQL con la Consola de administración de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Inicie la migración de datos desde la instancia de base de datos MySQL a desde la instantánea:

   Para iniciar la migración desde la instancia de base de datos:

   1. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, seleccione la instancia de base de datos de MySQL.
   2. En Actions (Acciones), elija Migrate latest snapshot (Migrar última instantánea).

Para iniciar la migración desde la instantánea:

1. Elija Snapshots (Instantáneas).
2. En la página Snapshots (Instantáneas), elija la instantánea que desea migrar a un clúster de base de datos de Aurora MySQL.
3. Elija Snapshot Actions y, a continuación, seleccione Migrate Snapshot.

Aparece la página Migrate Database.

3. Defina los siguientes valores en la página Migrate Database (Migrar base de datos):

   - Migrate to DB Engine (Migrar a motor de base de datos): seleccione `aurora`.
   - DB Engine Version (Versión de motor de base de datos): seleccione la versión del motor de base de datos para el clúster de base de datos de Aurora MySQL.
   - Clase de instancia de base de datos: elija una clase de instancia de base de datos que tenga el almacenamiento y la capacidad requeridos para la base de datos, por ejemplo `db.r3.large`. Los volúmenes de clúster de Aurora crecen automáticamente a medida que se incrementa la cantidad de datos de la base de datos. Un volumen de clúster de Aurora puede aumentar hasta un tamaño máximo de 128 tebibytes (TiB). Por lo tanto, solo tiene que seleccionar una clase de instancia de base de datos que se adapte a sus necesidades actuales de almacenamiento. Para obtener más información, consulte Información general del almacenamiento de Amazon Aurora.
   - DB Instance Identifier (Identificador de instancias de bases de datos): escriba un nombre para el clúster de base de datos que sea único para su cuenta en la región de AWS que ha seleccionado. Este identificador se utiliza en las direcciones de punto de enlace para las instancias del clúster de base de datos. Puede optar por agregar al nombre información como la región de AWS y el motor de base de datos que ha seleccionado, por ejemplo, **`aurora-cluster1`**.

     El identificador de instancias de bases de datos tiene las siguientes limitaciones:

     - Debe contener de 1 a 63 caracteres alfanuméricos o guiones.
     - El primer carácter debe ser una letra.
     - No puede terminar con un guion ni contener dos guiones consecutivos.
     - Debe ser único para todas las instancias de base de datos por cada cuenta de AWS y por cada región de AWS.

- Virtual Private Cloud (VPC): si ya dispone de una VPC, puede utilizarla con su clúster de base de datos de Aurora MySQL si selecciona el identificador de la VPC, por ejemplo, `vpc-a464d1c1`. Para obtener más información sobre la creación de una VPC., consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

  De lo contrario, puede elegir que Aurora cree una VPC automáticamente seleccionando Create a new VPC (Crear una nueva VPC).
- DB subnet group (Grupo de subredes de base de datos): si ya dispone de un grupo de subredes, puede utilizarlo con el clúster de base de datos de MySQL de Aurora si selecciona el identificador del grupo de subredes, por ejemplo, `gs-subnet-group1`.

  De lo contrario, puede elegir que Aurora cree un grupo de subredes automáticamente seleccionando Create a new subnet group (Crear un nuevo grupo de subredes).
- Public accessibility (Accesibilidad pública): seleccione No para especificar que a las instancias de su clúster de base de datos solo pueden obtener acceso los recursos que se encuentren dentro de su VPC. Seleccione Yes (Sí) para especificar que los recursos de la red pública pueden obtener acceso a las instancias de su clúster de base de datos. El valor predeterminado es Yes (Sí).

  > **Note**
  >
  > No es necesario que su clúster de base de datos de producción esté en una subred pública, ya que solo los servidores de su aplicación necesitan acceso a su clúster de base de datos. Si su clúster de base de datos no necesita estar en una subred pública, defina Publicly Accessible (Accesible públicamente) como No.
- Availability Zone (Zona de disponibilidad): seleccione la zona de disponibilidad para alojar la instancia principal del clúster de base de datos de Aurora MySQL. Para hacer que Aurora seleccione una zona de disponibilidad automáticamente, seleccione No Preference (Sin preferencias).
- Database Port (Puerto de base de datos): indique el puerto predeterminado que se utilizará al conectar a instancias del clúster de base de datos Aurora MySQL. El valor predeterminado es `3306`.

> **Note**
>
> Es posible que se encuentre detrás de un firewall de una compañía que no permite el acceso a los puertos predeterminados, como el puerto predeterminado de MySQL, el 3306. En este caso, proporcione un valor de puerto permitido por el firewall corporativo. Recuerde el valor del puerto cuando se conecte más adelante al clúster de base de datos de Aurora MySQL.

- Encryption (Cifrado): elija Enable Encryption (Habilitar cifrado) para que el nuevo clúster de base de datos de Aurora MySQL se cifre en reposo. Si elige Enable encryption (Habilitar cifrado), debe elegir una clave de KMS como valor de AWS KMS key.

  Si la instantánea de base de datos no está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo.

  Si la instantánea de base de datos está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo con la clave de cifrado especificada. Puede especificar la clave de cifrado utilizada por la instantánea de base de datos o una clave distinta. No puede crear un clúster de base de datos sin cifrar a partir de una instantánea de base de datos cifrada.

- Auto Minor Version Upgrade (Actualización automática a versiones secundarias): este ajuste no se aplica a los clústeres de base de datos Aurora MySQL.

  Para obtener más información acerca de las actualizaciones de motor de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

4. Elija Migrate (Migrar) para migrar la instantánea de base de datos.
5. Elija Instances y, a continuación, seleccione el icono de flecha para mostrar la información detallada del clúster de base de datos y monitorizar el progreso de la migración. En la página de detalles, encontrará el punto de enlace del clúster que se utiliza para conectar a la instancia principal del clúster de base de datos. Para obtener más información acerca de la conexión a un clúster de base de datos de Aurora MySQL, consulte Conexión a un clúster de base de datos Amazon Aurora.

AWS CLI

Puede crear un clúster de base de datos de Aurora desde una instantánea de base de datos de una instancia de base de datos de RDS for MySQL con el comando `restore-db-cluster-from-snapshot` y los siguientes parámetros:

- `--db-cluster-identifier`: nombre del clúster de base de datos que se creará.
- `--engine aurora-mysql`: para un clúster de base de datos compatible con MySQL 5.7 o con MySQL 8.0.
- `--kms-key-id`: el AWS KMS key para cifrar, si lo desea, el clúster de base de datos en función de si la instantánea de base de datos está cifrada o no.
  - Si la instantánea de base de datos no está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo. De lo contrario, el clúster no estará cifrado.
  - Si la instantánea de base de datos está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo con la clave de cifrado especificada. De lo contrario, el clúster de base de datos se cifrará en reposo con la clave de cifrado de la instantánea de base de datos.

    > **Note**
    >
    > No puede crear un clúster de base de datos sin cifrar a partir de una instantánea de base de datos cifrada.
- `--snapshot-identifier`: el nombre de recurso de Amazon (ARN) de la instantánea de base de datos que se va a migrar. Para obtener más información sobre los ARN de Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS).

Al migrar la instantánea de base de datos con el comando `RestoreDBClusterFromSnapshot`, este crea tanto el clúster de base de datos como la instancia principal.

En este ejemplo, va a crear un clúster de base de datos compatible con MySQL 5.7 denominado *`mydbcluster`* a partir de una instantánea de base de datos con un ARN definido en *`mydbsnapshotARN`*.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
```

```
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

En este ejemplo, va a crear un clúster de base de datos compatible con MySQL 5.7 denominado *mydbcluster* a partir de una instantánea de base de datos con un ARN definido en *mydbsnapshotARN*.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora

Aurora usa la funcionalidad de replicación de registros binarios de los motores de base de datos MySQL para crear un tipo especial de clúster de base de datos denominado réplica de lectura de Aurora para una instancia de base de datos de RDS para MySQL de origen. Las actualizaciones realizadas en la instancia de base de datos de RDS para MySQL de origen se replican de forma asíncrona en la réplica de lectura de Aurora.

Es recomendable usar esta funcionalidad para migrar desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Aurora MySQL creando una réplica de lectura de Aurora de la instancia de base de datos de RDS para MySQL de origen. Cuando el retraso de la réplica entre la instancia de base de datos de RDS para MySQL y la réplica de lectura de Aurora sea 0, podrá dirigir las aplicaciones cliente a la réplica de lectura de Aurora y detener después la replicación para convertir la réplica de lectura de Aurora en un clúster de base de datos de Aurora MySQL independiente. Esta migración puede tardar un tiempo considerable, aproximadamente varias horas por tebibyte (TiB) de datos.

Para obtener una lista de las regiones en las que está disponible Aurora, consulte Amazon Aurora en la Referencia general de AWS.

Cuando se crea una réplica de lectura de Aurora de una instancia de base de datos de RDS para MySQL, Amazon RDS crea una instantánea de base de datos de la instancia de base de datos de RDS para MySQL de origen (privada para Amazon RDS y sin cargo). Después, Amazon RDS migra los datos de la instantánea de base de datos a la réplica de lectura de Aurora. Una vez que los datos de la instantánea de base de datos se hayan migrado al nuevo clúster de base de datos de Aurora MySQL, Amazon RDS comenzará la replicación entre la instancia de base de datos de RDS para MySQL y el clúster de base de datos de Aurora MySQL. Si la instancia de base de datos de RDS para MySQL contiene tablas que usen motores de almacenamiento distintos de InnoDB o que usen el formato de filas comprimidas, puede acelerar el proceso de creación de una réplica de lectura de Aurora modificando esas tablas para que usen el motor de almacenamiento de InnoDB y el formato de filas dinámicas antes de crear la réplica de lectura de Aurora. Para obtener más información acerca del proceso de copia de una instantánea de base de datos MySQL en un clúster de base de datos de Aurora MySQL, consulte Migración de datos de una instancia de base de datos de RDS for MySQL a un clúster de base de datos de Amazon Aurora MySQL.

Solo puede tener una réplica de lectura de Aurora para una instancia de base de datos de RDS para MySQL.

> **Note**
>
> Pueden surgir problemas de replicación a causa de las diferencias de características entre Aurora MySQL y la versión del motor de base de datos de MySQL de su instancia de base de datos de RDS para MySQL, que es la replicación principal. Si se produce un error, puede encontrar ayuda en el foro de la comunidad de Amazon RDS o contáctese con AWS Support.
> No puede crear una réplica de lectura de Aurora si su instancia de base de datos de RDS para MySQL ya es el origen de una réplica de lectura entre regiones.
> No se puede migrar a la versión 3.05 o posteriores de Aurora MySQL desde algunas versiones anteriores de RDS for MySQL 8.0, como 8.0.11, 8.0.13 y 8.0.15. Le recomendamos que actualice a la versión 8.0.28 de RDS for MySQL antes de realizar la migración.

Para obtener más información sobre las réplicas de lectura de MySQL, consulte Trabajo con réplicas de lectura de instancias de base de datos MariaDB, MySQL y PostgreSQL.

Creación de una réplica de lectura de Aurora

Puede crear una réplica de lectura de Aurora para una instancia de base de datos de RDS para MySQL mediante la consola, la AWS CLI o la API de RDS.

Consola

Para crear una réplica de lectura de Aurora a partir de una instancia de base de datos de RDS para MySQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Seleccione la instancia de base de datos MySQL que desea usar como origen de una réplica de lectura de Aurora.
4. En Actions (Acciones), elija Create Aurora read replica (Crear réplica de lectura de Aurora).
5. Elija las especificaciones del clúster de base de datos que desee usar para la réplica de lectura de Aurora y que se describen en la tabla siguiente.

Opción	Descripción
DB instance class (Clase de instancia de base de datos)	Elija una clase de instancia de base de datos que defina los requisitos de procesamiento y memoria para la instancia principal del clúster de base de datos. Para obtener más información acerca de las opciones de clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora.
Multi-AZ deployment (Implementación Multi-AZ)	Elija Create Replica in Different Zone (Crear una réplica en una zona diferente) para crear una réplica en espera del nuevo clúster de base de datos en otra zona de disponibilidad de la región de AWS de destino a fin de permitir la conmutación por error. Para obtener más información acerca del uso de varias zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.

Opción	Descripción
DB Instance Identifier (Identificador de instancias de bases de datos)	Escriba un nombre para la instancia principal de su clúster de base de datos de réplica de lectura de Aurora. Este identificador se utiliza en la dirección del punto de enlace de la instancia principal del nuevo clúster de base de datos. El identificador de instancias de bases de datos tiene las siguientes limitaciones: • Debe contener de 1 a 63 caracteres alfanuméricos o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser único para todas las instancias de base de datos de cada cuenta de AWS y para cada región AWS. Como el clúster de base de datos réplica de lectura de Aurora entre regiones se crea a partir de una instantán ea de la instancia de base de datos origen, el nombre de usuario principal y la contraseña principal de la réplica de lectura de Aurora coinciden con el nombre de usuario principal y la contraseña principal de la instancia de base de datos origen.
Virtual Private Cloud (VPC) (Nube virtual privada)	Seleccione la VPC que alojará el clúster de base de datos. Seleccione Create new VPC (Crear nueva VPC) para que Aurora cree una VPC automáticamente. Para obtener más información, consulte Requisitos previos de clúster de base de datos.

Opción	Descripción
Grupo de subredes de base de datos	Seleccione el grupo de subredes de base de datos que desea utilizar para el clúster de base de datos. Seleccione Create new DB subnet group (Crear nuevo grupo de subredes de base de datos) para que Aurora cree un grupo de subredes de base de datos automáticamente. Para obtener más información, consulte Requisitos previos de clúster de base de datos.
Public accessibility (Accesibilidad pública)	Seleccione `Yes` para asignar al clúster de base de datos una dirección IP pública; de lo contrario, seleccione `No`. Las instancias del clúster de base de datos pueden ser una combinación de instancias de base de datos públicas y privadas. Para obtener más información acerca del modo de ocultar instancias al acceso público, consulte Cómo ocultar un clúster de base de datos en una VPC desde Internet..
Availability zone (Zona de disponibilidad)	Determine si desea especificar una zona de disponibilidad concreta. Para obtener más información acerca de las zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.
Grupo de seguridad de VPC (firewall)	Seleccione Create new VPC security group (Crear nuevo grupo de seguridad de VPC) para que Aurora cree un grupo de seguridad de VPC automáticamente. Seleccione Select existing VPC security groups (Seleccionar grupos de seguridad de VPC existentes) para especificar uno o varios grupos de seguridad de VPC para proteger el acceso de red al clúster de base de datos. Para obtener más información, consulte Requisitos previos de clúster de base de datos.

Opción	Descripción
Database port (Puerto de base de datos)	Especifique el puerto que deben usar las aplicaciones y las utilidades para obtener acceso a la base de datos. Los clústeres de base de datos de Aurora MySQL usan el puerto 3306 de MySQL de forma predeterminada. Los firewalls de algunas compañías bloquean las conexiones a este puerto. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el nuevo clúster de base de datos.
Grupo de parámetros de base de datos	Seleccione un grupo de parámetros de base de datos para el clúster de base de datos de Aurora MySQL. Aurora tiene un grupo de parámetros de base de datos predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros de base de datos. Para obtener más información acerca de los grupos de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora.
Grupo de parámetros de clúster de base de datos	Seleccione el grupo de parámetros del clúster de base de datos para el clúster de base de datos de Aurora MySQL. Aurora cuenta con un grupo de parámetros de clúster base de datos predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros de clúster de base de datos. Para obtener más información acerca de los grupos de parámetros de clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Opción	Descripción
Cifrado	Elija Disable encryption (Deshabilitar cifrado) si no desea que se cifre su nuevo clúster de base de datos de Aurora. Elija Enable encryption (Habilitar cifrado) para que el nuevo clúster de base de datos de Aurora se cifre en reposo. Si elige Enable encryption (Habilitar cifrado), debe elegir una clave de KMS como valor de AWS KMS key. Si la instancia de base de datos MySQL no está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo. Si la instancia de base de datos MySQL está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo con la clave de cifrado especificada. Puede especificar la clave de cifrado utilizada por la instancia de base de datos MySQL o una clave distinta. No puede crear un clúster de base de datos sin cifrar a partir de una instancia de base de datos MySQL cifrada.
Prioridad	Elija una prioridad de conmutación por error para el clúster de base de datos. Si no selecciona un valor, el ajuste predeterminado es tier-1. Esta prioridad determina el orden en que se promueven las réplicas de Aurora cuando el sistema se recupera de un error en la instancia principal. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.
Backup retention period (Periodo de retención de copia de seguridad)	Seleccione el tiempo (entre 1 y 35 días) durante el que Aurora conserva los backups de la base de datos. Los backups se pueden utilizar para las restauraciones a un momento dado (PITR) de la base de datos con una precisión de segundos.

Opción	Descripción
Supervisión mejorada	Elija Enable enhanced monitoring (Habilitar monitorización mejorada) a fin de habilitar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta su clúster de base de datos. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
Monitoring Role (Rol de supervisión)	Solo está disponible si Enhanced Monitoring (Monitorización mejorada) se establece en Enable enhanced monitoring (Habilitar monitorización mejorada). Elija el rol de IAM que ha creado para permitir que Aurora se comunique con Amazon CloudWatch Logs automáticamente o elija Default (Predeterminado) para que Aurora cree un rol automáticamente denominado `rds-monitoring-role` . Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
Granularity (Grado de detalle)	Solo está disponible si Enhanced Monitoring (Monitorización mejorada) se establece en Enable enhanced monitoring (Habilitar monitorización mejorada). Defina el intervalo, en segundos, en el que se recopilan las métricas para el clúster de base de datos.
Auto minor version upgrade (Actualización automática de versiones secundarias)	Esta configuración no se aplica a los clústeres de base de datos Aurora MySQL. Para obtener más información acerca de las actualizaciones de motor de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

Opción	Descripción
Periodo de mantenimiento	Seleccione Select window (Seleccionar periodo) y especifique el intervalo de tiempo semanal durante el que se puede llevar a cabo el mantenimiento del sistema. O seleccione No preference (Sin preferencia) para que Aurora asigne un periodo aleatoriamente.

6. Elija Create read replica (Crear réplica de lectura).

AWS CLI

Para crear una réplica de lectura de Aurora a partir de una instancia de base de datos de RDS para MySQL de origen, utilice los comandos `create-db-cluster` y `create-db-instance` de AWS CLI para crear un nuevo clúster de base de datos de Aurora MySQL. Cuando llame al comando `create-db-cluster`, incluya el parámetro `--replication-source-identifier` para identificar el Nombre de recurso de Amazon (ARN) de la instancia de base de datos MySQL de origen. Para obtener más información sobre los ARN de Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS).

No especifique el nombre de usuario maestro, la contraseña maestra o el nombre de la base de datos, ya que la réplica de lectura de Aurora usa el mismo nombre de usuario maestro, la misma contraseña maestra y el mismo nombre de base de datos que la instancia de base de datos MySQL de origen.

Para Linux, macOS o:Unix

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 \
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

En:Windows

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 ^
```

```
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Si utiliza la consola para crear una réplica de lectura de Aurora, Aurora crea automáticamente la instancia principal para la réplica de lectura de Aurora del clúster de base de datos. Si usa la AWS CLI para crear una réplica de lectura de Aurora, debe crear expresamente la instancia primaria del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de base de datos.

Puede crear una instancia principal para el clúster de base de datos con el comando `create-db-instance` de la AWS CLI y los siguientes parámetros.

- `--db-cluster-identifier`

  El nombre del clúster de base de datos.
- `--db-instance-class`

  El nombre de la clase de instancia de base de datos que se va a utilizar para la instancia principal.
- `--db-instance-identifier`

  El nombre de la instancia principal.
- `--engine aurora`

En este ejemplo, va a crear una instancia principal llamada *myreadreplicainstance* para el clúster de base de datos llamado *myreadreplicacluster* con la clase de instancia de base de datos especificada en *myinstanceclass*.

Example

Para Linux, macOS o:Unix

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass \
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

En:Windows

```
aws rds create-db-instance ^
```

```
    --db-cluster-identifier myreadreplicacluster ^
    --db-instance-class myinstanceclass ^
    --db-instance-identifier myreadreplicainstance ^
    --engine aurora
```

API de RDS

Para crear una réplica de lectura de Aurora a partir de una instancia de base de datos de RDS para MySQL de origen, use los comandos `CreateDBCluster` y `CreateDBInstance` de la API de Amazon RDS para crear una instancia principal y un clúster de base de datos de Aurora nuevos. No especifique el nombre de usuario maestro, la contraseña maestra o el nombre de la base de datos, ya que la réplica de lectura de Aurora usa el mismo nombre de usuario maestro, la misma contraseña maestra y el mismo nombre de base de datos que la instancia de base de datos de RDS para MySQL de origen.

Puede crear un nuevo clúster de base de datos de Aurora para una réplica de lectura de Aurora a partir de una instancia de base de datos de RDS para MySQL de origen con el comando `CreateDBCluster` de la API de Amazon RDS y los siguientes parámetros:

- `DBClusterIdentifier`

  El nombre del clúster de base de datos que se creará.

- `DBSubnetGroupName`

  El nombre del grupo de subredes de la base de datos que desea asociar con este clúster de base de datos.

- `Engine=aurora`
- `KmsKeyId`

  La AWS KMS key para cifrar, si lo desea, el clúster de base de datos en función de si la instancia de base de datos MySQL está cifrada o no.

  - Si la instancia de base de datos MySQL no está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo. De lo contrario, el clúster de base de datos se cifrará en reposo con la clave de cifrado predeterminada para la cuenta.
  - Si la instancia de base de datos MySQL está cifrada, especifique una clave de cifrado para cifrar el clúster de base de datos en reposo con la clave de cifrado especificada. De lo contrario, el clúster de base de datos se cifrará en reposo con la clave de cifrado de la instancia de base de datos MySQL.

> **Note**
>
> No puede crear un clúster de base de datos sin cifrar a partir de una instancia de base de datos MySQL cifrada.

- `ReplicationSourceIdentifier`

  El nombre de recurso de Amazon (ARN) de la instancia de base de datos MySQL de origen. Para obtener más información sobre los ARN de Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS).

- `VpcSecurityGroupIds`

  La lista de grupos de seguridad de VPC de EC2 que se va a asociar con este clúster de base de datos.

En este ejemplo, se crea un clúster de base de datos llamado *`myreadreplicacluster`* a partir de una instancia de base de datos MySQL principal con un ARN definido en *`mysqlmasterARN`*, asociado con un grupo de subredes de base de datos llamado *`mysubnetgroup`* y un grupo de seguridad de la VPC llamado *`mysecuritygroup`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBCluster
    &DBClusterIdentifier=myreadreplicacluster
    &DBSubnetGroupName=mysubnetgroup
    &Engine=aurora
    &ReplicationSourceIdentifier=mysqlprimaryARN
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &VpcSecurityGroupIds=mysecuritygroup
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150927/us-east-1/rds/aws4_request
    &X-Amz-Date=20150927T164851Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=6a8f4bd6a98f649c75ea04a6b3929ecc75ac09739588391cd7250f5280e716db
```

Si utiliza la consola para crear una réplica de lectura de Aurora, Aurora crea automáticamente la instancia principal para la réplica de lectura de Aurora del clúster de base de datos. Si usa la AWS CLI para crear una réplica de lectura de Aurora, debe crear expresamente la instancia primaria del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de base de datos.

Puede crear una instancia principal para el clúster de base de datos con el comando `CreateDBInstance` de la API de Amazon RDS y los siguientes parámetros:

- `DBClusterIdentifier`

  El nombre del clúster de base de datos.
- `DBInstanceClass`

  El nombre de la clase de instancia de base de datos que se va a utilizar para la instancia principal.
- `DBInstanceIdentifier`

  El nombre de la instancia principal.
- `Engine=aurora`

En este ejemplo, va a crear una instancia principal llamada *`myreadreplicainstance`* para el clúster de base de datos llamado *`myreadreplicacluster`* con la clase de instancia de base de datos especificada en *`myinstanceclass`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBInstance
    &DBClusterIdentifier=myreadreplicacluster
    &DBInstanceClass=myinstanceclass
    &DBInstanceIdentifier=myreadreplicainstance
    &Engine=aurora
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140424/us-east-1/rds/aws4_request
    &X-Amz-Date=20140424T194844Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=bee4aabc750bf7dad0cd9e22b952bd6089d91e2a16592c2293e532eeaab8bc77
```

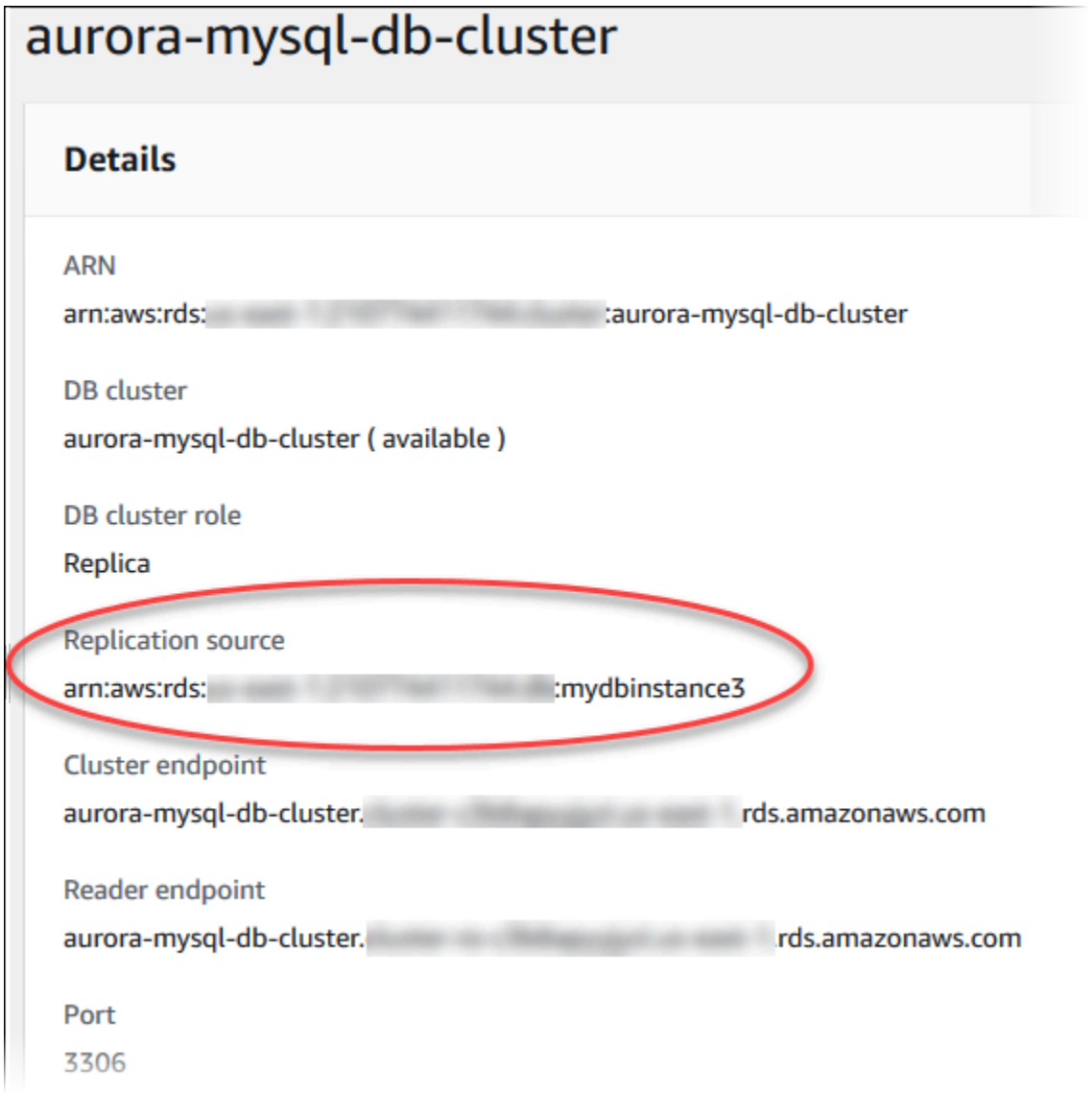
Visualización de una réplica de lectura de Aurora

Para ver las relaciones de reproducción de MySQL con Aurora MySQL de los clústeres de base de datos de Aurora MySQL, use la Consola de administración de AWS o la AWS CLI.

Consola

Para ver la instancia de base de datos MySQL principal para una réplica de lectura de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos la réplica de lectura de Aurora para mostrar sus detalles. La información de la instancia de base de datos MySQL principal está en el campo Replication source (Origen de replicación).



AWS CLI

Para ver las relaciones de replicación de MySQL con Aurora MySQL de los clústeres de base de datos de Aurora MySQL mediante AWS CLI, utilice los comandos `describe-db-clusters` y `describe-db-instances`.

Para determinar qué instancia de base de datos MySQL es la instancia principal, utilice `describe-db-clusters` y especifique el identificador de clúster de la réplica de lectura de Aurora para la opción `--db-cluster-identifier`. Consulte el elemento `ReplicationSourceIdentifier` de la salida para ver el ARN de la instancia de base de datos que es la replicación principal.

Para determinar qué clúster de base de datos es la réplica de lectura de Aurora, utilice `describe-db-instances` y especifique el identificador de la instancia de base de datos MySQL para la opción `--db-instance-identifier`. Consulte el elemento `ReadReplicaDBClusterIdentifiers` de la salida para ver el identificador del clúster de base de datos la réplica de lectura de Aurora.

Example

Para Linux, macOS o:Unix

```
aws rds describe-db-clusters \
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances \
    --db-instance-identifier mysqlprimary
```

En:Windows

```
aws rds describe-db-clusters ^
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances ^
    --db-instance-identifier mysqlprimary
```

Promoción de una réplica de lectura de Aurora

Una vez que se complete la migración, puede promocionar la réplica de lectura de Aurora a un clúster de base de datos independiente mediante la Consola de administración de AWS o la AWS CLI.

Después, puede dirigir sus aplicaciones cliente al punto de conexión para la réplica de lectura de Aurora. Para obtener más información acerca de los puntos de enlace de Aurora, consulte Conexiones de puntos de conexión de Amazon Aurora. La promoción debe completarse relativamente rápido, y podrá leer y escribir en la réplica de lectura de Aurora durante la promoción. Sin embargo, no podrá eliminar la instancia de base de datos MySQL principal ni desvincular la instancia de base de datos y la réplica de lectura de Aurora durante ese tiempo.

Antes de promocionar la réplica de lectura de Aurora, detenga la escritura de transacciones en la instancia de base de datos MySQL de origen y espere hasta que el retardo de la réplica de lectura

de Aurora llegue a 0. Puede ver el retardo de una réplica de lectura de Aurora llamando al comando `SHOW SLAVE STATUS` (Aurora MySQL, versión 2) o `SHOW REPLICA STATUS` (Aurora MySQL versión 3) en la réplica de lectura de Aurora. Verifique el valor Seconds behind master (Segundos detrás del maestro).

Puede comenzar escribiendo en la réplica de lectura de Aurora cuando las transacciones de escritura en el principal se hayan detenido y el retardo de la réplica sea 0. Si escribe en la réplica de lectura de Aurora antes de esto y modifica tablas que también se están modificando en el MySQL principal, se arriesga a interrumpir la replicación en Aurora. Si ocurre esto, tendrá que eliminar y volver a crear la réplica de lectura de Aurora.

Consola

Para promover una réplica de lectura de Aurora a un clúster de base de datos Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos para la réplica de lectura de Aurora.
4. En Actions (Acciones), seleccione Promote (Promover).
5. Elija Promote Read Replica (Promover réplica de lectura).

Después de promocionar, confirme que la promoción se ha completado mediante el siguiente procedimiento.

Para confirmar que se ha promovido la réplica de lectura Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Events.
3. En la página Events (Eventos) , compruebe que hay un evento `Promoted Read Replica cluster to a stand-alone database cluster` para el clúster promocionado.

Una vez que se haya completado la promoción, la instancia de base de datos MySQL principal y la réplica de lectura de Aurora se desvincularán y podrá eliminar sin riesgo la instancia de base de datos si lo desea.

AWS CLI

Para promocionar una réplica de lectura de Aurora a un clúster de base de datos independiente, utilice el comando `promote-read-replica-db-cluster` de la AWS CLI.

Example

Para Linux, macOS o:Unix

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

En:Windows

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

Administración de Amazon Aurora MySQL

En las siguientes secciones, se analiza la administración de un clúster de base de datos de Amazon Aurora MySQL.

Temas

- Administración del rendimiento y el escalado para Amazon Aurora MySQL
- Búsqueda de datos anteriores de un clúster de base de datos de Aurora
- Pruebas de Amazon Aurora MySQL por medio de consultas de inserción de errores
- Modificación de las tablas de Amazon Aurora con DDL rápido
- Visualización del estado del volumen para un clúster de base de datos de Aurora MySQL

Administración del rendimiento y el escalado para Amazon Aurora MySQL

Escalado de las instancias de base de datos Aurora MySQL

Puede escalar las instancias de base de datos Aurora MySQL de dos formas, mediante el escalado de instancia y mediante el escalado de lectura. Para obtener más información acerca del escalado de lectura, consulte Escalado de lectura.

Para escalar el clúster de bases de datos de Aurora MySQL, modifique la clase de instancia de base de datos para cada instancia de base de datos del clúster de bases de datos. Aurora MySQL admite varias clases de instancia de base de datos optimizadas para Aurora. No utilice las clases de instancia db.t2 o db.t3 con clústeres de Aurora que tengan más de 40 TB. Para obtener especificaciones de las clases de instancia de base de datos admitidas por Aurora MySQL, consulte Clases de instancia de base de datos de Amazon Aurora.

> **Note**
>
> Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más información sobre las clases de instancia T, consulte Utilización de clases de instancia T para el desarrollo y la prueba.

Número máximo de conexiones a una instancia de base de datos Aurora MySQL

El número máximo de conexiones permitidas a una instancia de base de datos Aurora MySQL viene determinado por el parámetro `max_connections` del grupo de parámetros de nivel de instancia para la instancia de base de datos.

En la siguiente tabla se indica el valor resultante predeterminado de `max_connections` para cada clase de instancia de base de datos disponible para Aurora MySQL. Puede aumentar el número máximo de conexiones de la instancia de base de datos Aurora MySQL escalando la instancia hasta una clase de instancia de base de datos con más memoria o definiendo un valor más grande para el parámetro `max_connections` en el grupo de parámetros de base de datos de la instancia, hasta un máximo de 16 000.

> **Tip**
>
> Si sus aplicaciones abren y cierran conexiones con frecuencia, o mantienen abierto un gran número de conexiones de larga duración, le recomendamos que utilice Amazon RDS Proxy. El RDS Proxy es un proxy de base de datos totalmente administrado y de alta disponibilidad que utiliza agrupación de conexiones para compartir conexiones de base de datos de forma segura y eficiente. Para obtener más información acerca de RDS Proxy, consulteAmazon RDS Proxy para Aurora.

Para obtener más información acerca de cómo las instancias de Aurora Serverless v2 manejan este parámetro, consulte Número máximo de conexiones para Aurora Serverless v2.

Clase de instancia	Valor predeterminado de max_connections		
db.t2.small	45		
db.t2.medium	90		
db.t3.small	45		
db.t3.medium	90		
db.t3.large	135		

Clase de instancia	Valor predeterminado de max_connections		
db.t4g.medium	90		
db.t4g.large	135		
db.r3.large	1 000		
db.r3.xlarge	2000		
db.r3.2xlarge	3 000		
db.r3.4xlarge	4000		
db.r3.8xlarge	5000		
db.r4.large	1 000		
db.r4.xlarge	2000		
db.r4.2xlarge	3 000		
db.r4.4xlarge	4000		
db.r4.8xlarge	5000		
db.r4.16xlarge	6000		
db.r5.large	1 000		
db.r5.xlarge	2000		
db.r5.2xlarge	3 000		
db.r5.4xlarge	4000		
db.r5.8xlarge	5000		

Clase de instancia	Valor predeterminado de max_connections		
db.r5.12xlarge	6000		
db.r5.16xlarge	6000		
db.r5.24xlarge	7000		
db.r6g.large	1 000		
db.r6g.xlarge	2000		
db.r6g.2xlarge	3 000		
db.r6g.4xlarge	4000		
db.r6g.8xlarge	5000		
db.r6g.12xlarge	6000		
db.r6g.16xlarge	6000		
db.r6i.large	1 000		
db.r6i.xlarge	2000		
db.r6i.2xlarge	3 000		
db.r6i.4xlarge	4000		
db.r6i.8xlarge	5000		
db.r6g.12xlarge	6000		
db.r6i.16xlarge	6000		
db.r6i.24xlarge	7000		

Clase de instancia	Valor predeterminado de max_connections		
db.r6i.32xlarge	7000		
db.r7g.large	1 000		
db.r7g.xlarge	2000		
db.r7g.2xlarge	3 000		
db.r7g.4xlarge	4000		
db.r7g.8xlarge	5000		
db.r7g.12xlarge	6000		
db.r7g.16xlarge	6000		
db.r7i.large	1 000		
db.r7i.xlarge	2000		
db.r7i.2xlarge	3 000		
db.r7i.4xlarge	4000		
db.r7i.8xlarge	5000		
db.r7i.12xlarge	6000		
db.r7i.16xlarge	6000		
db.r7i.24xlarge	7000		
db.r7i.48xlarge	8000		
db.r8g.large	1 000		

Clase de instancia	Valor predeterminado de max_connections		
db.r8g.xlarge	2000		
db.r8g.2xlarge	3 000		
db.r8g.4xlarge	4000		
db.r8g.8xlarge	5000		
db.r8g.12xlarge	6000		
db.r8g.16xlarge	6000		
db.r8g.24xlarge	7000		
db.r8g.48xlarge	8000		
db.x2g.large	2000		
db.x2g.xlarge	3 000		
db.x2g.2xlarge	4000		
db.x2g.4xlarge	5000		
db.x2g.8xlarge	6000		
db.x2g.12xlarge	7000		
db.x2g.16xlarge	7000		

Tip

El cálculo del parámetro `max_connections` utiliza la base logarítmica 2 (distinta del logaritmo natural) y el valor de `DBInstanceClassMemory` en bytes para la clase de

instancia de Aurora MySQL seleccionada. El parámetro solo acepta valores enteros y las partes decimales se truncan en los cálculos. La fórmula implementa los límites de conexión de la siguiente manera:

- Incremento de 1000 conexiones para instancias R3, R4 y R5 grandes
- Incremento de 45 conexiones para variantes de memoria de instancias T2 y T3

Ejemplo: en db.r6g.large, aunque la fórmula calcula 1069,2, el sistema implementa 1000 para mantener patrones incrementales coherentes.

Si crea un nuevo grupo de parámetros para personalizar su propio límite de conexión predeterminado, verá que el límite de conexión predeterminado se obtiene mediante una fórmula basada en el valor `DBInstanceClassMemory`. Como se muestra en la tabla anterior, la fórmula produce límites de conexión que aumentan en 1000 a medida que la memoria se duplica entre instancias R3, R4 y R5 cada vez más grandes, y en 45 para diferentes tamaños de memoria de instancias T2 y T3.

Consulte Especificación de parámetros de base de datos para obtener más información sobre cómo se calcula `DBInstanceClassMemory`.

Las instancias de base de datos de RDS para MySQL y Aurora MySQL tienen distintas cantidades de sobrecarga de memoria. Por lo tanto, el valor `max_connections` puede ser diferente para las instancias de base de datos de RDS for MySQL y Aurora MySQL que utilizan la misma clase de instancia. Los valores de la tabla sólo se aplican a instancias de base de datos de Aurora MySQL.

> **Note**
>
> Los límites de conectividad mucho más bajos para las instancias T2 y T3 se deben a que en Aurora esas clases de instancias están pensadas solo para escenarios de desarrollo y pruebas, no para cargas de trabajo de producción.

Los límites de conexión predeterminados se ajustan para los sistemas que utilizan los valores predeterminados para otros consumidores de memoria importantes, como el grupo del búfer y la caché de consultas. Si cambia esas otras configuraciones para el clúster, considere ajustar el límite de conexión para tener en cuenta el aumento o la disminución de la memoria disponible en las instancias de base de datos.

Límites de almacenamiento temporal de Aurora MySQL

Aurora MySQL almacena tablas e índices en el subsistema de almacenamiento de Aurora. Aurora MySQL utiliza almacenamiento local o temporal independiente para archivos temporales no persistentes y tablas temporales distintas de InnoDB. El almacenamiento local incluye también archivos que se utilizan para fines tales como ordenar conjuntos de datos grandes durante el procesamiento de consultas o para operaciones de creación de índices. No incluye tablas temporales de InnoDB.

Para obtener más información sobre las tablas temporales de la versión 3 de Aurora MySQL, consulte Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3. Para obtener más información sobre las tablas temporales de la versión 2, consulte Comportamiento de los espacios de tabla temporales en Aurora MySQL versión 2.

Los datos y los archivos temporales de estos volúmenes se pierden al iniciar y detener la instancia de base de datos y durante la sustitución del host.

Estos volúmenes de almacenamiento local están respaldados por Amazon Elastic Block Store (EBS) y pueden ampliarse utilizando una clase de instancia de base de datos mayor. Para obtener más información acerca del almacenamiento, consulte Almacenamiento de Amazon Aurora.

El almacenamiento local también se utiliza para importar datos de Amazon S3 mediante `LOAD DATA FROM S3` o `LOAD XML FROM S3` y para exportar datos a S3 mediante SELECT INTO OUTFILE S3. Para obtener más información acerca de la importación y la exportación a S3, consulte lo siguiente:

- Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3
- Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3

Aurora MySQL utiliza un almacenamiento permanente independiente para los registros de errores, los registros generales, los registros de consultas lentas y los registros de auditoría para la mayoría de las clases de instancias de base de datos de Aurora MySQL (sin incluir los tipos de clases de instancias de rendimiento ampliable, como db.t2, db.t3 y db.t4g). Los datos de este volumen se retienen al iniciar y detener la instancia de base de datos y durante la sustitución del host.

Este volumen de almacenamiento permanente también está respaldado por Amazon EBS y tiene un tamaño fijo según la clase de instancia de base de datos. No puede ampliarse utilizando una clase de instancia de base de datos mayor.

En la siguiente tabla se muestra la cantidad máxima de almacenamiento temporal y permanente disponible para cada clase de instancia de base de datos de Aurora MySQL. Para obtener más información sobre la compatibilidad de la clase de instancia de la base de datos con Aurora, consulte Clases de instancia de base de datos de Amazon Aurora.

Clase de instancia de base de datos	Almacenamiento temporal/ local máximo disponible (GiB)	Almacenamiento máximo adicional disponible para archivos de registro (GiB)
db.x2g.16xlarge	1 280	500
db.x2g.12xlarge	960	500
db.x2g.8xlarge	640	500
db.x2g.4xlarge	320	500
db.x2g.2xlarge	160	60
db.x2g.xlarge	80	60
db.x2g.large	40	60
db.r8g.48xlarge	3840	500
db.r8g.24xlarge	1920	500
db.r8g.16xlarge	1 280	500
db.r8g.12xlarge	960	500
db.r8g.8xlarge	640	500
db.r8g.4xlarge	320	500
db.r8g.2xlarge	160	60
db.r8g.xlarge	80	60
db.r8g.large	32	60

Clase de instancia de base de datos	Almacenamiento temporal/ local máximo disponible (GiB)	Almacenamiento máximo adicional disponible para archivos de registro (GiB)
db.r7i.48xlarge	3840	500
db.r7i.24xlarge	1920	500
db.r7i.16xlarge	1 280	500
db.r7i.12xlarge	960	500
db.r7i.8xlarge	640	500
db.r7i.4xlarge	320	500
db.r7i.2xlarge	160	60
db.r7i.xlarge	80	60
db.r7i.large	32	60
db.r7g.16xlarge	1 280	500
db.r7g.12xlarge	960	500
db.r7g.8xlarge	640	500
db.r7g.4xlarge	320	500
db.r7g.2xlarge	160	60
db.r7g.xlarge	80	60
db.r7g.large	32	60
db.r6i.32xlarge	2560	500
db.r6i.24xlarge	1920	500
db.r6i.16xlarge	1 280	500

Clase de instancia de base de datos	Almacenamiento temporal/ local máximo disponible (GiB)	Almacenamiento máximo adicional disponible para archivos de registro (GiB)
db.r6g.12xlarge	960	500
db.r6i.8xlarge	640	500
db.r6i.4xlarge	320	500
db.r6i.2xlarge	160	60
db.r6i.xlarge	80	60
db.r6i.large	32	60
db.r6g.16xlarge	1 280	500
db.r6g.12xlarge	960	500
db.r6g.8xlarge	640	500
db.r6g.4xlarge	320	500
db.r6g.2xlarge	160	60
db.r6g.xlarge	80	60
db.r6g.large	32	60
db.r5.24xlarge	1920	500
db.r5.16xlarge	1 280	500
db.r5.12xlarge	960	500
db.r5.8xlarge	640	500
db.r5.4xlarge	320	500
db.r5.2xlarge	160	60

Clase de instancia de base de datos	Almacenamiento temporal/ local máximo disponible (GiB)	Almacenamiento máximo adicional disponible para archivos de registro (GiB)
db.r5.xlarge	80	60
db.r5.large	32	60
db.r4.16xlarge	1 280	500
db.r4.8xlarge	640	500
db.r4.4xlarge	320	500
db.r4.2xlarge	160	60
db.r4.xlarge	80	60
db.r4.large	32	60
db.t4g.large	32	–
db.t4g.medium	32	–
db.t3.large	32	–
db.t3.medium	32	–
db.t3.small	32	–
db.t2.medium	32	–
db.t2.small	32	–

Important

Estos valores representan la cantidad máxima teórica de almacenamiento gratuito en cada instancia de base de datos. El almacenamiento local real disponible para usted puede ser inferior. Aurora utiliza un poco de almacenamiento local para sus procesos de administración, y la instancia de base de datos utiliza algo de almacenamiento local incluso antes de cargar

datos. Puede monitorear el almacenamiento temporal disponible para una instancia de base de datos específica con la métrica `FreeLocalStorage` de CloudWatch, descrita en Métricas de Amazon CloudWatch para Amazon Aurora. Puede verificar la cantidad de almacenamiento gratuito en este momento. También puede representar un gráfico de la cantidad de almacenamiento gratuito a lo largo del tiempo. La monitorización del almacenamiento gratuito a lo largo del tiempo lo ayuda a determinar si el valor aumenta o disminuye, o a encontrar los valores mínimos, máximos o medios.
(Esto no se aplica a .)

Búsqueda de datos anteriores de un clúster de base de datos de Aurora

Con Edición compatible con Amazon Aurora MySQL puede realizar búsquedas de datos anteriores en un clúster de base de datos en un momento específico, sin restaurar datos desde un backup.

Contenido

- Información general de búsquedas de datos anteriores
  - Ventana de búsqueda de datos anteriores
  - Tiempo de búsqueda de datos anteriores
  - Limitaciones de la búsqueda de datos anteriores
- Disponibilidad en regiones y versiones
- Consideraciones de actualización para clústeres habilitados para backtrack
- Configuración de la búsqueda de datos anteriores en un clúster de base de datos de Aurora MySQL
- Búsqueda de datos anteriores para un clúster de base de datos de Aurora MySQL
- Monitorización de la búsqueda de datos anteriores para un clúster de base de datos de Aurora MySQL
- Suscripción a un evento de búsqueda de datos anteriores con la consola
- Recuperar búsqueda de datos anteriores existentes
- Deshabilitación de la búsqueda de datos anteriores para un clúster de base de datos de Aurora MySQL

Información general de búsquedas de datos anteriores

La búsqueda de datos anteriores "rebobina" el clúster de base de datos al momento que especifique. La búsqueda de datos anteriores no es un sustituto del backup del clúster de base de datos para que pueda restaurarlo a un momento determinado. No obstante, la búsqueda de datos anteriores presenta las siguientes ventajas respecto al backup y restauración tradicional:

- Puede deshacer errores con facilidad. Si lleva a cabo una acción destructiva por error, por ejemplo DELETE sin una cláusula WHERE, puede realizar una búsqueda de datos anteriores del clúster de base de datos a un momento anterior a la acción destructiva con una interrupción de servicio mínima.
- Puede realizar una búsqueda de datos anteriores en un clúster de base de datos rápidamente. La restauración de un clúster de base de datos a un momento determinado lanza un nuevo clúster de base de datos y lo restaura a partir de datos de backup o de una instantánea de clúster de base de datos, lo que puede tardar horas. La búsqueda de datos anteriores de un clúster de base de datos no requiere un nuevo clúster de base de datos y rebobina el clúster de base de datos en cuestión de minutos.
- Puede explorar cambios de datos anteriores. Puede realizar búsqueda de datos anteriores de un clúster de base de datos repetidamente adelante y atrás en el tiempo para ayudar a determinar cuándo se produjo un cambio de datos particular. Por ejemplo, puede realizar una búsqueda de datos anteriores de un clúster de base de datos de hace tres horas y, a continuación, realizar la búsqueda de datos anteriores hacia adelante en el tiempo una hora. En este caso, el tiempo de búsqueda de datos anteriores es dos horas antes de la hora original.

> **Note**
>
> Para obtener más información acerca la restauración de un clúster de base de datos a un momento determinado, consulte Información general de copias de seguridad y restauración de un clúster de base de datos Aurora.

Ventana de búsqueda de datos anteriores

Con la búsqueda de datos anteriores, hay una ventana de búsqueda de datos anteriores de destino y una ventana de búsqueda de datos anteriores real:

- La ventana de búsqueda de datos anteriores de destino es la cantidad de tiempo que desea poder realizar búsqueda de datos anteriores en su clúster de base de datos. Cuando habilita la búsqueda de datos anteriores, especifica una ventana de búsqueda de datos anteriores de destino. Por ejemplo, podría especificar una ventana de búsqueda de datos anteriores de 24 horas si desea poder realizar la búsqueda de datos anteriores del clúster de base de datos un día.
- La ventana de búsqueda de datos anteriores real es la cantidad de tiempo real en la que puede realizar búsqueda de datos anteriores en su clúster de base de datos, que puede ser inferior a la de la ventana de búsqueda de datos anteriores de destino. La ventana de búsqueda de datos anteriores real se basa en su carga de trabajo y en el almacenamiento disponible para información de almacenamiento sobre cambios de base de datos, denominados registros de cambio.

A medida que actualiza su clúster de base de datos de Aurora con la búsqueda de datos anteriores habilitada, genera registros de cambios. Aurora mantiene los registros de cambios del periodo de búsqueda de datos anteriores de destino y paga una tarifa por hora para almacenarlos. Tanto la ventana de búsqueda de datos anteriores de destino como la carga de trabajo de su clúster de base de datos determinan el número de registros de cambio que puede almacenar. La carga de trabajo es el número de cambios que realiza en su clúster de base de datos en un período de tiempo dado. Si la carga de trabajo es pesada, almacena más registros de cambio en su ventana de búsqueda de datos anteriores de la que tendría si la carga de trabajo fuera ligera.

Puede entender la ventana de búsqueda de datos anteriores de destino como el objetivo para la cantidad de tiempo máxima que desea poder realizar la búsqueda de datos anteriores en su clúster de base de datos. En la mayoría de los casos, puede realizar una búsqueda de datos anteriores del período de tiempo máximo que haya especificado. No obstante, en algunos casos, el clúster de base de datos no puede almacenar suficientes registros de cambio para realizar la búsqueda de datos anteriores durante el período de tiempo máximo y la ventana de búsqueda de datos anteriores real es inferior a la de destino. Normalmente, la ventana de búsqueda de datos anteriores real es más pequeña que el destino cuando se tiene una carga de trabajo extremadamente pesada en el clúster de base de datos. Cuando la ventana de búsqueda de datos anteriores real es inferior al destino, le enviamos una notificación.

Cuando la búsqueda de datos anteriores está habilitada para un clúster de base de datos y elimina una tabla almacenada en el clúster de base de datos, Aurora mantiene dicha tabla en los registros de cambio de búsqueda de datos anteriores. Lo hace para que pueda volver a un momento anterior a la eliminación de la tabla. Si no tiene espacio suficiente en su ventana de búsqueda de datos anteriores para almacenar la tabla, la tabla podría acabar por eliminarse de los registros de cambios de búsqueda de datos anteriores.

Tiempo de búsqueda de datos anteriores

Aurora siempre realiza la búsqueda de datos anteriores en un momento que sea coherente para el clúster de base de datos. Al hacerlo así, se elimina la posibilidad de transacciones sin confirmar cuando se ha completado la búsqueda de datos anteriores. Cuando se especifica una hora para una búsqueda de datos anteriores, Aurora elige automáticamente la hora coherente más próxima posible. Este enfoque significa que la búsqueda de datos anteriores completada podría no concordar exactamente con la hora que especifique, pero puede determinar la hora exacta de una búsqueda de datos anteriores con el comando describe-db-cluster-backtracks de la CLI de AWS. Para obtener más información, consulte Recuperar búsqueda de datos anteriores existentes.

Limitaciones de la búsqueda de datos anteriores

Las limitaciones siguientes son aplicables a la búsqueda de datos anteriores:

- La búsqueda de datos anteriores solo está disponible para clústeres de base de datos que se crearon con la característica de búsqueda de datos anteriores habilitada. Tampoco puede modificar un clúster de base de datos para habilitar la característica de búsqueda de datos anteriores. Puede habilitar la característica de búsqueda de datos anteriores cuando cree un clúster de base de datos nuevo o restaure una instantánea de un clúster de base de datos.
- El límite para una ventana de búsqueda de datos anteriores es de 72 horas.
- La búsqueda de datos anteriores afecta a todo el clúster de base de datos. Por ejemplo, puede realizar la búsqueda de datos anteriores selectivamente en una única tabla o una única actualización de datos.
- No puede crear réplicas de lectura entre regiones desde un clúster habilitado para búsqueda de datos anteriores, pero sí puede habilitar la replicación de registros binarios (binlog) en el clúster. Además, si intenta realizar una búsqueda de datos anteriores en un clúster de base de datos para el que está habilitado el registro binario, suele producirse un error, a menos que haya elegido forzar la búsqueda de datos anteriores. Cualquier intento de forzar una búsqueda de datos anteriores interrumpirá las réplicas de lectura descendentes e interferirá con otras operaciones, como las implementaciones azul/verde.
- No puede realizar una búsqueda de datos anteriores de un clon de base de datos a una hora anterior a la que se creó dicho clon de base de datos. Sin embargo, puede utilizar la base de datos original para realizar una búsqueda de datos anteriores a un momento anterior a la creación del clon. Para obtener más información acerca de la clonación de la base de datos, consulte Clonación de un volumen de clúster de base de datos de Amazon Aurora.

- La búsqueda de datos anteriores provoca una breve interrupción de la instancia de base de datos. Debe detener o pausar las aplicaciones antes de iniciar una operación de búsqueda de datos anteriores para asegurarse de que no haya ninguna solicitud nueva de lectura o escritura. Durante la operación de búsqueda de datos anteriores, Aurora pone en pausa la base de datos, cierra las conexiones abiertas y borra las lecturas y escrituras sin confirmar. A continuación, espera a que se complete la operación de búsqueda de datos anteriores.
- No puede restaurar una instantánea entre regiones de un clúster habilitado para una búsqueda de datos anteriores en una región de AWS que no admita una búsqueda de datos anteriores.
- Si realiza una actualización local de un clúster que tiene habilitado el retroceso desde la versión 2 de Aurora MySQL a la versión 3, no podrá realizar un retroceso a un punto en el tiempo anterior a la actualización.

Disponibilidad en regiones y versiones

Backtrack no está disponible para Aurora PostgreSQL.

A continuación se presentan los motores compatibles y la disponibilidad de Backtrack con Aurora MySQL.

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Este de EE. UU. (Norte de Virginia)	Todas las versiones	Todas las versiones
Este de EE. UU. (Ohio)	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Norte de California)	Todas las versiones	Todas las versiones
Oeste de EE. UU. (Oregón)	Todas las versiones	Todas las versiones
Africa (Cape Town)	–	–

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Asia-Pacífico (Hong Kong)	–	–
Asia-Pacífico (Yakarta)	–	–
Asia-Pacífico (Malasia)	–	–
Asia-Pacífico (Melbourne)	–	–
Asia-Pacífico (Mumbai)	Todas las versiones	Todas las versiones
Asia-Pacífico (Nueva Zelanda)	–	–
Asia-Pacífico (Osaka)	Todas las versiones	Versión 2.07.3 y posterior
Asia-Pacífico (Seúl)	Todas las versiones	Todas las versiones
Asia-Pacífico (Singapur)	Todas las versiones	Todas las versiones
Asia-Pacífico (Sídney)	Todas las versiones	Todas las versiones
Asia-Pacífico (Taipéi)	–	–
Asia-Pacífico (Tailandia)	–	–
Asia-Pacífico (Tokio)	Todas las versiones	Todas las versiones

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
Canadá (centro)	Todas las versiones	Todas las versiones
Oeste de Canadá (Calgary)	–	–
China (Pekín)	–	–
China (Ningxia)	–	–
Europa (Fráncfort)	Todas las versiones	Todas las versiones
Europa (Irlanda)	Todas las versiones	Todas las versiones
Europa (Londres)	Todas las versiones	Todas las versiones
Europa (Milán)	–	–
Europa (París)	Todas las versiones	Todas las versiones
Europa (España)	–	–
Europa (Estocolmo)	–	–
Europa (Zúrich)	–	–
Israel (Tel Aviv)	–	–
México (central)	–	–
Medio Oriente (Baréin)	–	–
Medio Oriente (EAU)	–	–

Región	Aurora MySQL versión 3	Aurora MySQL versión 2
América del Sur (São Paulo)	–	–
AWS GovCloud (Este de EE. UU.)	–	–
AWS GovCloud (Oeste de EE. UU.)	–	–

Consideraciones de actualización para clústeres habilitados para backtrack

Puede actualizar un clúster de base de datos que tiene habilitado el retroceso desde la versión 2 de Aurora MySQL a la versión 3, ya que todas las versiones secundarias de la versión 3 de Aurora MySQL son compatibles con la función de retroceso.

Configuración de la búsqueda de datos anteriores en un clúster de base de datos de Aurora MySQL

Para utilizar la característica de búsqueda de datos anteriores, debe habilitar la búsqueda de datos anteriores y especificar una ventana de búsqueda de datos anteriores de destino. De lo contrario, se deshabilita la búsqueda de datos anteriores.

Para el periodo de búsqueda de datos anteriores de destino, especifique la cantidad de tiempo que desea poder rebobinar la base de datos mediante la búsqueda de datos anteriores. Aurora intenta retener suficientes registros de cambio para admitir ese periodo.

Consola

Puede utilizar la consola para configurar la búsqueda de datos anteriores cuando se crea un nuevo clúster de base de datos. También puede modificar un clúster de base de datos para cambiar la ventana de retroceso de un clúster habilitado para backtrack. Si desactiva el seguimiento de retroceso completo para un clúster al establecer la ventana de retroceso en 0, no podrá volver a habilitar el retroceso para ese clúster.

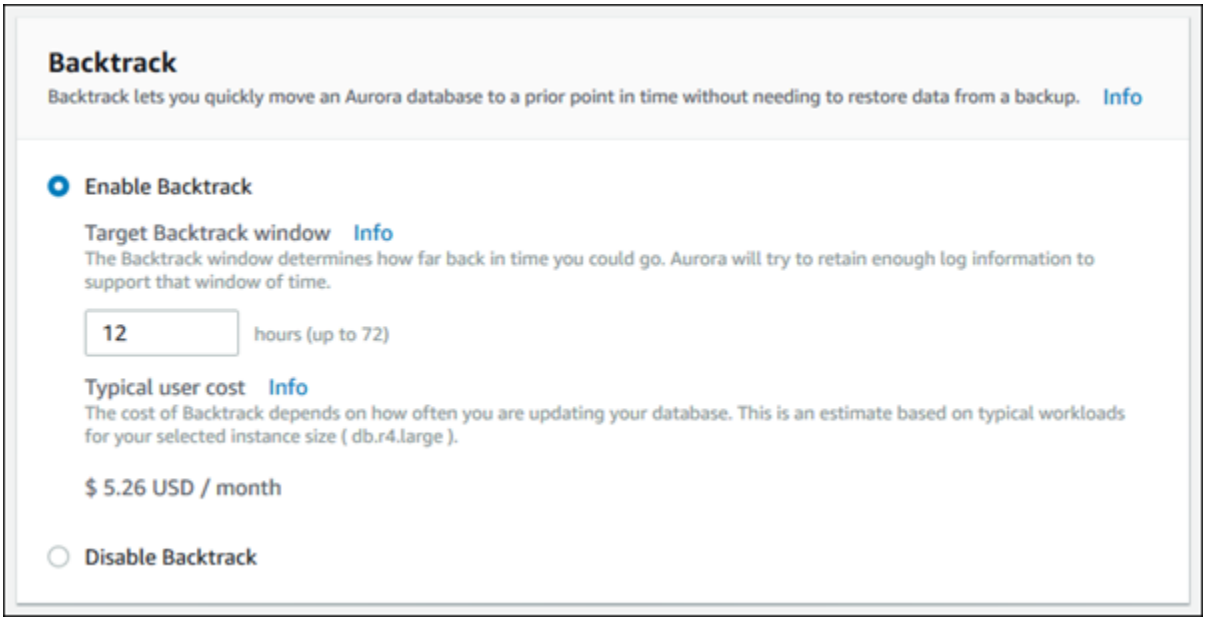
Temas

- Configuración de la búsqueda de datos anteriores con la consola al crear un clúster de base de datos
- Configuración de la búsqueda de datos anteriores con la consola al modificar un clúster de base de datos

Configuración de la búsqueda de datos anteriores con la consola al crear un clúster de base de datos

Cuando se crea un nuevo clúster de base de datos Aurora MySQL, la búsqueda de datos anteriores está configurada cuando elige Enable Backtrack (Habilitar búsqueda de datos anteriores) y especifica un valor de Target Backtrack window (Ventana de búsqueda de datos anteriores de destino) que sea mayor que cero en la sección Backtrack (Búsqueda de datos anteriores).

Para crear un clúster de base de datos, siga las instrucciones en Creación de un clúster de base de datos de Amazon Aurora. La imagen siguiente muestra la sección Backtrack (Búsqueda de datos anteriores).



Cuando se crea un nuevo clúster de base de datos, Aurora no tiene ningún dato para la carga de trabajo del clúster de base de datos. Por tanto, no puede estimar un costo específicamente para el nuevo clúster de base de datos. En lugar de ello, la consola presenta un costo de usuario típico para la ventana de búsqueda de datos anteriores de destino basado en una carga de trabajo típica. El costo típico tiene como objetivo proporcionar una referencia general para el costo de la característica de búsqueda de datos anteriores.

> **Important**
>
> El costo real podría no coincidir con el costo típico, ya que el costo real se basa en la carga de trabajo de su clúster de base de datos.

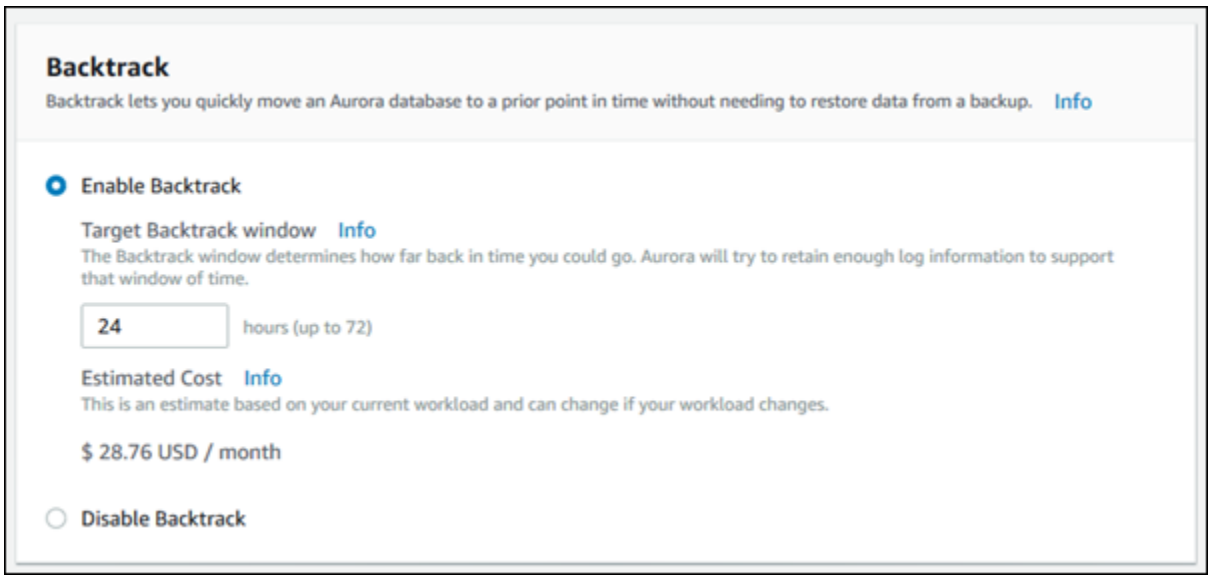
Configuración de la búsqueda de datos anteriores con la consola al modificar un clúster de base de datos

Puede modificar la búsqueda de datos anteriores de un clúster de base de datos utilizando la consola.

> **Note**
>
> Actualmente, puede modificar el backtracking solo para un clúster de base de datos que tenga habilitada la característica Backtrack. La sección Backtrack no aparece para un clúster de base de datos creado con la característica Backtrack deshabilitada o si la característica Backtrack se ha deshabilitado para el clúster de base de datos.

Para modificar la búsqueda de datos anteriores de un clúster de base de datos utilizando la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Databases (Bases de datos).
3. Elija el clúster que desea modificar y elija Modify (Modificar).
4. Para la Target Backtrack window (Ventana de búsqueda de datos anteriores de destino), modifique la cantidad de tiempo que desea poder realizar la búsqueda de datos anteriores. El límite son 72 horas.



La consola muestra el costo estimado para la cantidad de tiempo que ha especificado en función de la carga de trabajo pasada del clúster de base de datos:

- Si la búsqueda de datos anteriores se deshabilitó en el clúster de base de datos, la estimación de costo se basa en la métrica `VolumeWriteIOPS` para el clúster de base de datos en Amazon CloudWatch.
- Si la búsqueda de datos anteriores se habilitó anteriormente en el clúster de base de datos, la estimación de costo se basa en la métrica `BacktrackChangeRecordsCreationRate` para el clúster de base de datos en Amazon CloudWatch.

5. Elija Continue.
6. Para Scheduling of Modifications (Programación de modificaciones), elija una de las siguientes:
   - Apply during the next scheduled maintenance window (Aplicar durante la próxima ventana de mantenimiento programada): espere para aplicar la modificación de Target Backtrack window (Ventana de búsqueda de datos anteriores de destinos) hasta la próxima ventana de mantenimiento.
   - Apply immediately (Aplicar inmediatamente): aplique la modificación de Target Backtrack window (Ventana de búsqueda de datos anteriores de destino) lo antes posible.
7. Elija Modify clúster (Modificar clúster).

AWS CLI

Cuando se crea un nuevo clúster de base de datos de Aurora MySQL con el comando create-db-clúster de la CLI de AWS, la búsqueda de datos anteriores se configura cuando se especifica un valor `--backtrack-window` mayor que cero. El valor `--backtrack-window` especifica la ventana de búsqueda de datos anteriores de destino. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

También puede especificar el valor `--backtrack-window` con los siguientes comandos de la CLI de:AWS

- modify-db-clúster
- restore-db-clúster-from-s3
- restore-db-clúster-from-snapshot
- restore-db-clúster-to-point-in-time

El siguiente procedimiento describe cómo modificar la ventana de búsqueda de datos anteriores de destino para un clúster de base de datos utilizando la AWS CLI.

Para modificar la ventana de búsqueda de datos anteriores de destino para un clúster de base de datos utilizando la AWS CLI

- Llame al comando modify-db-clúster de la CLI de AWS y suministre los siguientes valores:
  - `--db-cluster-identifier`: el nombre del clúster de base de datos.
  - `--backtrack-window`: el número máximo de segundos que desea poder realizar una búsqueda de datos anteriores en el clúster de base de datos.

  En el siguiente ejemplo, se establece la ventana de búsqueda de datos anteriores de destino para `sample-cluster` en un día (86 400 segundos).

  Para Linux, macOS o:Unix

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 86400
```

En:Windows

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 86400
```

> **Note**
>
> Actualmente, puede habilitar la búsqueda de datos anteriores solo para un clúster de base de datos que se creó con la característica de búsqueda de datos anteriores habilitada.

API de RDS

Cuando se crea un nuevo clúster de base de datos de Aurora MySQL utilizando la operación CreateDBclúster de la API de Amazon RDS, la búsqueda de datos anteriores se configura cuando se especifica un valor de `BacktrackWindow` mayor que cero. El valor `BacktrackWindow` especifica la ventana de búsqueda de datos anteriores de destino para el clúster de base de datos especificado en el valor `DBClusterIdentifier`. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

También puede especificar el valor `BacktrackWindow` utilizando las siguientes operaciones de la API:

- ModifyDBclúster
- RestoreDBclústerFromS3
- RestoreDBclústerFromSnapshot
- RestoreDBclústerToPointInTime

> **Note**
>
> Actualmente, puede habilitar la búsqueda de datos anteriores solo para un clúster de base de datos que se creó con la característica de búsqueda de datos anteriores habilitada.

Búsqueda de datos anteriores para un clúster de base de datos de Aurora MySQL

Puede realizar una búsqueda de datos anteriores de un clúster de base de datos a una marca temporal de búsqueda de datos anteriores especificada. Si la marca temporal de búsqueda de datos anteriores no es anterior al tiempo de búsqueda de datos anteriores más temprano posible y no se encuentra en el futuro, se realiza la búsqueda de datos anteriores del clúster de base de datos a dicha marca temporal.

En caso contrario, se suele producir un error. Además, si intenta realizar una búsqueda de datos anteriores en un clúster de base de datos para el que está habilitado el registro binario, suele producirse un error normalmente, a menos que haya elegido forzar la búsqueda de datos anteriores. El forzado de una búsqueda de datos anteriores puede interferir con otras operaciones que utilicen el registro binario.

Important

La búsqueda de datos anteriores no genera entradas de log binario para los cambios que realiza. Si tiene habilitado el registro binario para el clúster de base de datos, la búsqueda de datos anteriores podría no ser compatible con la implementación de log binario.

Note

Para los clones de base de datos, no puede realizar una búsqueda de datos anteriores del clúster de base de datos antes de la fecha y hora en la que se creó el clon. Para obtener más información acerca de la clonación de la base de datos, consulte Clonación de un volumen de clúster de base de datos de Amazon Aurora.

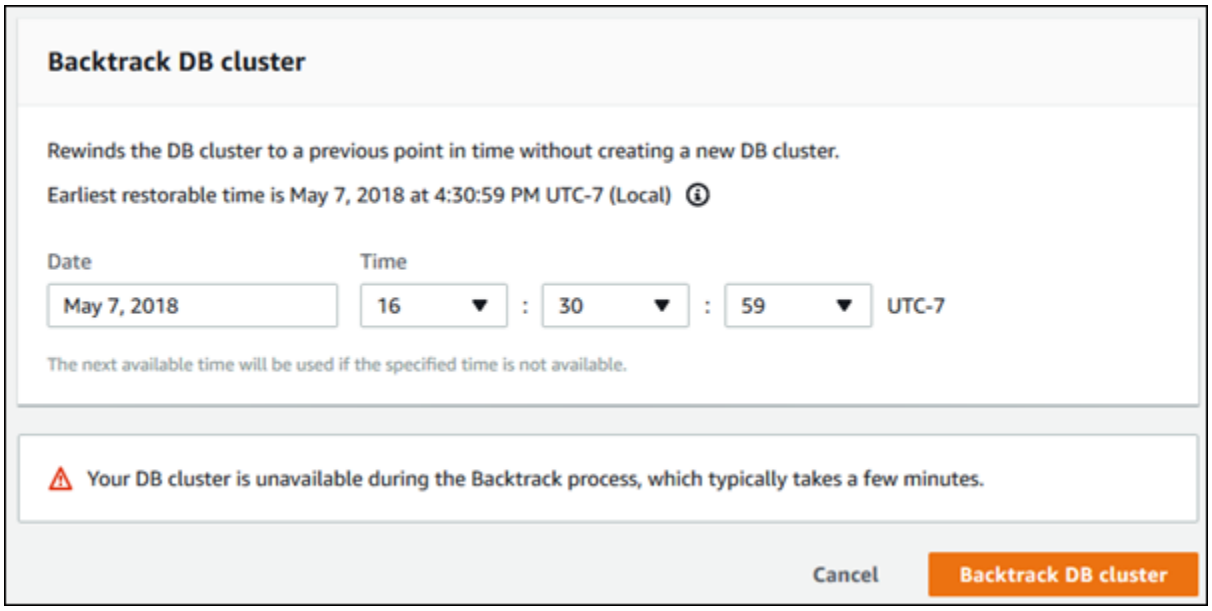
Consola

El siguiente procedimiento describe cómo realizar una operación de búsqueda de datos anteriores para un clúster de base de datos utilizando la consola.

Para realizar una operación de búsqueda de datos anteriores utilizando la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, elija Instances (Instancias).
3. Elija la instancia principal del clúster de la base de datos en la que desea realizar la búsqueda de datos anteriores.
4. En Actions (Acciones), elija (Clúster de base de datos de búsqueda de datos anteriores).
5. En la página Backtrack DB clúster (Realizar búsqueda de datos anteriores de clúster de base de datos), especifique la marca temporal de búsqueda de datos anteriores en la que realizar la búsqueda de datos anteriores en el clúster de base de datos.



6. Elija Backtrack DB clúster (Realizar búsqueda de datos anteriores en clúster de base de datos).

AWS CLI

El siguiente procedimiento describe cómo realizar una búsqueda de datos anteriores en un clúster de base de datos usando la AWS CLI.

Para realizar una búsqueda de datos anteriores de un clúster de base de datos utilizando la AWS CLI

- Llame al comando backtrack-db-clúster de la CLI de AWS y suministre los siguientes valores:
  - `--db-cluster-identifier`: el nombre del clúster de base de datos.
  - `--backtrack-to`: la marca temporal de búsqueda de datos anteriores para realizar la búsqueda de datos anteriores en el clúster de base de datos, especificada en formato ISO 8601.

El siguiente ejemplo realiza una búsqueda de datos anteriores en el clúster de base de datos `sample-cluster` el 19 de marzo de 2018 a las 10:00 h.

Para Linux, macOS o:Unix

```
aws rds backtrack-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-to 2018-03-19T10:00:00+00:00
```

En:Windows

```
aws rds backtrack-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-to 2018-03-19T10:00:00+00:00
```

API de RDS

Para realizar una búsqueda de datos anteriores en un clúster de base de datos utilizando la API de Amazon RDS, utilice la acción BacktrackDBclúster. Esta acción realiza una búsqueda de datos anteriores en el clúster de base de datos especificado en el valor `DBClusterIdentifier` a la hora especificada.

Monitorización de la búsqueda de datos anteriores para un clúster de base de datos de Aurora MySQL

Puede ver información de búsqueda de datos anteriores y monitorear métricas de búsqueda de datos anteriores para un clúster de base de datos.

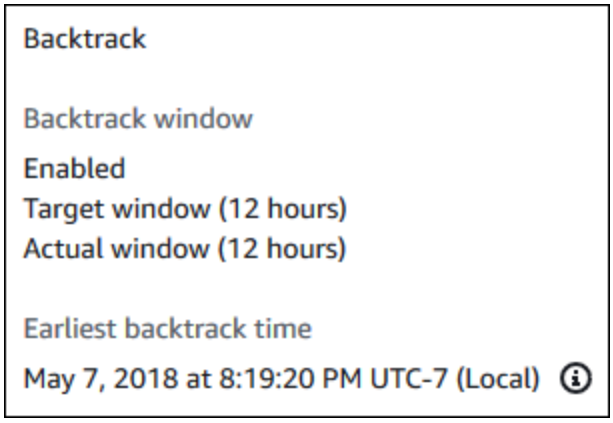
Consola

Para ver información de búsqueda de datos anteriores y monitorear métricas de búsqueda de datos anteriores utilizando la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Databases (Bases de datos).

3. Elija el nombre del clúster de búsqueda de datos anteriores para abrir información sobre el mismo.

   La información de búsqueda de datos anteriores se encuentra en la sección Backtrack (Búsqueda de datos anteriores).



   Cuando la búsqueda de datos anteriores está habilitada, está disponible la información siguiente:

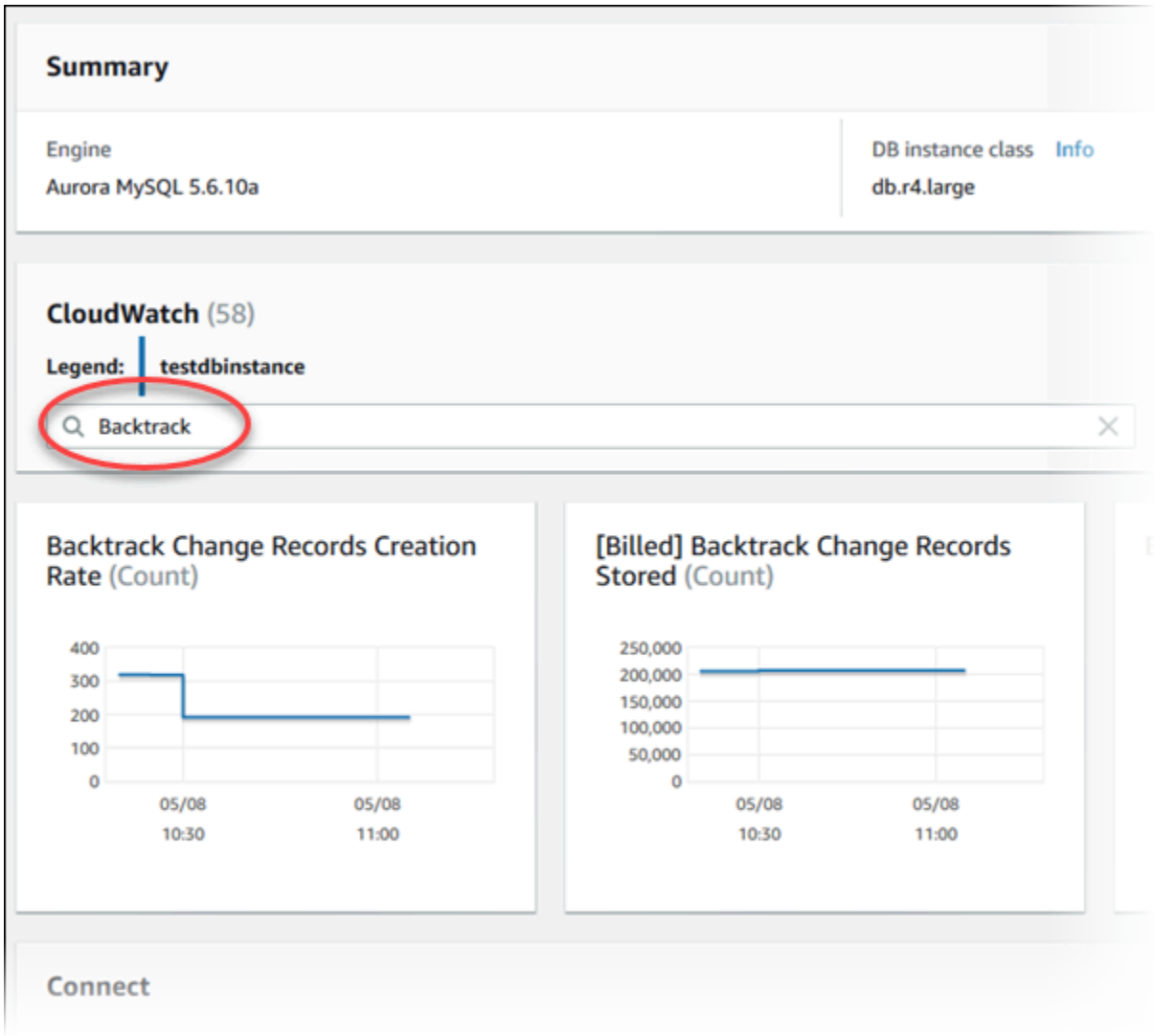
   - Target window (Ventana de destino): la cantidad de tiempo actual especificada para la ventana de búsqueda de datos anteriores de destino. El objetivo es la cantidad de tiempo máxima que puede realizar la búsqueda de datos anteriores si hay suficiente almacenamiento.
   - Actual window (Ventana real): la cantidad de tiempo real que puede realizar la búsqueda de datos anteriores, que puede ser inferior a la ventana de búsqueda de datos anteriores de destino. La ventana de búsqueda de datos anteriores real se basa en su carga de trabajo y en el almacenamiento disponible para mantener los registros de cambio de búsqueda de datos anteriores.
   - Earliest backtrack time (Tiempo de búsqueda de datos anteriores más temprano): el tiempo de búsqueda de datos anteriores más temprano posible para el clúster de base de datos. No puede realizar una búsqueda de datos anteriores de un clúster de base de datos de un momento anterior al tiempo mostrado.

4. Realice lo siguiente para ver las métricas de búsqueda de datos anteriores del clúster de base de datos:

   a. En el panel de navegación, elija Instances (Instancias).

   b. Elija el nombre de la instancia principal del clúster de base de datos para mostrar sus detalles.

c. En la sección CloudWatch, escriba **`Backtrack`** en el recuadro CloudWatch para mostrar solo las métricas de búsqueda de datos anteriores.



Se muestran las siguientes métricas:

- Backtrack Change Records Creation Rate (Count) [Tasa de creación de registros de cambio de búsqueda de datos anteriores (Recuento)]: esta métrica muestra el número de registros de cambio de búsqueda de datos anteriores creados durante cinco minutos para su clúster de base de datos. Puede utilizar esta métrica para estimar el costo de búsqueda de datos anteriores de su ventana de búsqueda de datos anteriores de destino.
- [Billed] Backtrack Change Records Stored (Count) ([Facturado] Registros de cambio de búsqueda de datos anteriores almacenados [Recuento]): esta métrica muestra el número real de registros de cambio de búsqueda de datos anteriores utilizados por su clúster de base de datos.

- Backtrack Window Actual (Minutes) [Ventana de búsqueda de datos anteriores real (Minutos)]: esta métrica muestra si hay una diferencia entre la ventana de búsqueda de datos anteriores de destino y la ventana de búsqueda de datos anteriores real. Por ejemplo, si su ventana de búsqueda de datos anteriores de destino es de 2 horas (120 minutos) y esta métrica muestra que la ventana de búsqueda de datos anteriores real es de 100 minutos, entonces la ventana de búsqueda de datos anteriores real es inferior al destino.
- Backtrack Window Alert (Count) [Alerta de ventana de búsqueda de datos anteriores (Recuento)]: esta métrica muestra con qué frecuencia la ventana de búsqueda de datos anteriores real es inferior a la ventana de búsqueda de datos anteriores de destino para un período de tiempo dado.

> **Note**
>
> Las métricas siguientes podrían llevar un retardo detrás de la hora actual:
>
> - Tasa de creación de registros de cambio de búsqueda de datos anteriores (Recuento)
> - [Billed] Backtrack Change Records Stored (Count) ([Facturado] Registros de cambio de búsqueda de datos anteriores almacenados [Recuento])

AWS CLI

El siguiente procedimiento describe cómo ver la información de búsqueda de datos anteriores para un clúster de base de datos utilizando la AWS CLI.

Para ver la información de búsqueda de datos anteriores de un clúster de base de datos utilizando AWS CLI

- Llame al comando describe-db-clústers de la CLI de AWS y suministre los siguientes valores:
  - `--db-cluster-identifier`: el nombre del clúster de base de datos.

  El ejemplo siguiente enumera información de búsqueda de datos anteriores para `sample-cluster`.

Para Linux, macOS o:Unix

```
aws rds describe-db-clusters \
    --db-cluster-identifier sample-cluster
```

En:Windows

```
aws rds describe-db-clusters ^
    --db-cluster-identifier sample-cluster
```

API de RDS

Para ver la información de búsqueda de datos anteriores para un clúster de base de datos utilizando la API de Amazon RDS, utilice la operación DescribeDBclústers. Esta acción devuelve la información de búsqueda de datos anteriores para el clúster de base de datos especificado en el valor `DBClusterIdentifier`.

Suscripción a un evento de búsqueda de datos anteriores con la consola

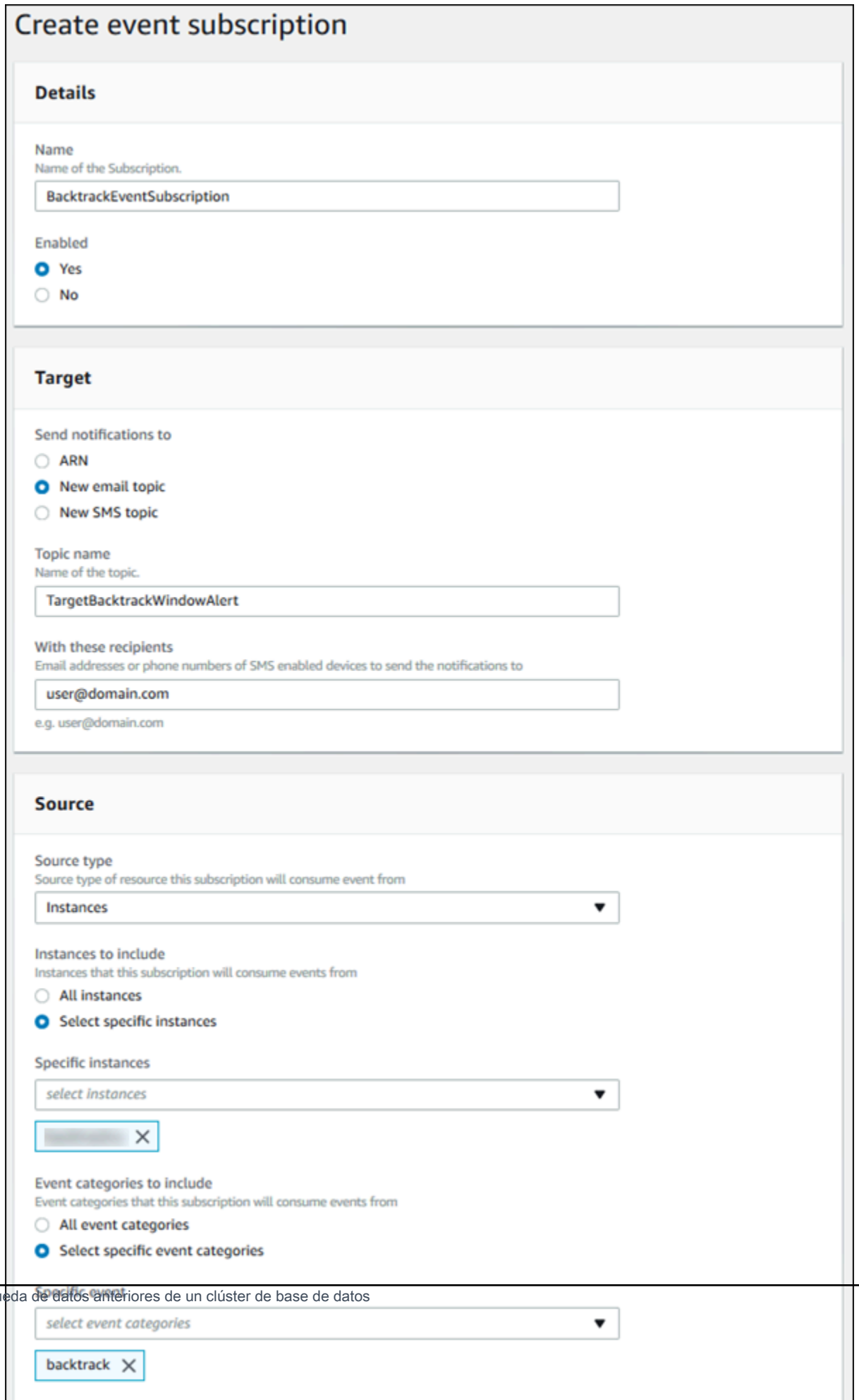
El siguiente procedimiento describe cómo suscribirse a un evento de búsqueda de datos anteriores utilizando la consola. El evento le envía una notificación de correo electrónico o texto cuando su ventana de búsqueda de datos anteriores real es inferior a su ventana de búsqueda de datos anteriores de destino.

Para ver información de búsqueda de datos anteriores mediante la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Event subscriptions (Suscripciones de evento).
3. Seleccione Create event subscription (Crear suscripción de evento).
4. En el cuadro Name (Nombre), escriba un nombre para la suscripción de evento y asegúrese de que se haya seleccionado Yes (Sí) en Enabled (Habilitado).
5. En la sección Target (Destino), elija New email topic (Nuevo tema de correo electrónico).

6. Para Topic name (Nombre de tema), escriba un nombre para el tema y para With these recipients (Con estos destinatarios), introduzca las direcciones de correo electrónico o los números de teléfono para recibir las notificaciones.
7. En la sección Source (Origen), elija Instances (Instancias) para Source type (Tipo de origen).
8. Para Instances to include (Instancias que incluir), elija Select specific instances (Seleccionar instancias específicas) y elija su instancia de base de datos.
9. Para Event categories to include (Categorías de evento que incluir), elija Select specific event categories (Seleccionar categorías de evento específicas) y elija backtrack (búsqueda de datos anteriores).

   La página debería tener un aspecto similar a la página siguiente.


Create event subscription
Details
Name
Name of the Subscription.
BacktrackEventSubscription
Enabled
Yes
No
Target
Send notifications to
ARN
New email topic
New SMS topic
Topic name
Name of the topic.
TargetBacktrackWindowAlert
With these recipients
Email addresses or phone numbers of SMS enabled devices to send the notifications to
user@domain.com
e.g. user@domain.com
Source
Source type
Source type of resource this subscription will consume event from
Instances
Instances to include
Instances that this subscription will consume events from
All instances
Select specific instances
Specific instances
select instances
Event categories to include
Event categories that this subscription will consume events from
All event categories
Select specific event categories
select event categories
backtrack

10. Seleccione Create (Crear).

Recuperar búsqueda de datos anteriores existentes

Puede recuperar información de búsquedas de datos anteriores existentes para un clúster de base de datos. Esta información incluye el identificador único de la búsqueda de datos anteriores, la fecha y la hora para realizar la búsqueda de datos anteriores de inicio y final, la fecha y la hora a la que se solicitó la búsqueda de datos anteriores y el estado actual de la búsqueda de datos anteriores.

> **Note**
>
> Actualmente no se pueden recuperar las búsquedas de datos anteriores existentes utilizando la consola.

AWS CLI

El siguiente procedimiento describe cómo recuperar las búsquedas de datos anteriores existentes para un clúster de base de datos utilizando la AWS CLI.

Para recuperar búsquedas de datos anteriores existentes utilizando la AWS CLI

- Llame al comando describe-db-cluster-backtracks de la CLI de AWS y suministre los siguientes valores:
  - `--db-cluster-identifier`: el nombre del clúster de base de datos.

  El ejemplo siguiente recuperar búsquedas de datos anteriores existentes para `sample-cluster`.

  Para Linux, macOS o Unix:

```
aws rds describe-db-cluster-backtracks \
    --db-cluster-identifier sample-cluster
```

  Para Windows:

```
aws rds describe-db-cluster-backtracks ^
    --db-cluster-identifier sample-cluster
```

API de RDS

Para recuperar información sobre las búsquedas de datos anteriores para un clúster de base de datos utilizando la API de Amazon RDS, utilice la operación DescribeDBClusterBacktracks. Esta acción devuelve la información acerca de las búsquedas de datos anteriores para el clúster de base de datos especificado en el valor `DBClusterIdentifier`.

Deshabilitación de la búsqueda de datos anteriores para un clúster de base de datos de Aurora MySQL

Puede deshabilitar la característica de búsqueda de datos anteriores para un clúster de base de datos.

Consola

Puede deshabilitar la búsqueda de datos anteriores de un clúster de base de datos utilizando la consola. Después de desactivar el seguimiento de retroceso completo para un clúster, no podrá volver a habilitarlo para ese clúster.

Para deshabilitar la característica de búsqueda de datos anteriores de un clúster de base de datos utilizando la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Databases (Bases de datos).
3. Elija el clúster que desea modificar y elija Modify (Modificar).
4. En la sección Backtrack (Búsqueda de datos anteriores), seleccione Disable Backtrack (Deshabilitar búsqueda de datos anteriores).
5. Elija Continue.
6. Para Scheduling of Modifications (Programación de modificaciones), elija una de las siguientes:

   - Apply during the next scheduled maintenance window (Aplicar durante la próxima ventana de mantenimiento programada): espere para aplicar la modificación hasta la próxima ventana de mantenimiento.
   - Apply immediately (Aplicar inmediatamente): aplique la modificación lo antes posible.

7. Elija Modify clúster (Modificar clúster).

AWS CLI

Puede deshabilitar la característica de búsqueda de datos anteriores de un clúster de base de datos utilizando la AWS CLI estableciendo la ventana de búsqueda de datos anteriores de destino en `0` (cero). Después de desactivar el seguimiento de retroceso completo para un clúster, no podrá volver a habilitarlo para ese clúster.

Para modificar la ventana de búsqueda de datos anteriores de destino para un clúster de base de datos utilizando la AWS CLI

- Llame al comando modify-db-clúster de la CLI de AWS y suministre los siguientes valores:
  - `--db-cluster-identifier`: el nombre del clúster de base de datos.
  - `--backtrack-window` – especifique `0` para desactivar el retroceso.

  El ejemplo siguiente deshabilita la característica de búsqueda de datos anteriores para el `sample-cluster` estableciendo `--backtrack-window` en `0`.

  Para Linux, macOS o:Unix

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 0
```

  En:Windows

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 0
```

API de RDS

Para deshabilitar la característica de búsqueda de datos anteriores de un clúster de base de datos utilizando la API de Amazon RDS, utilice la operación ModifyDBclúster. Establezca el

valor `BacktrackWindow` en `0` (cero) y especifique el clúster de base de datos en el valor `DBClusterIdentifier`. Después de desactivar el seguimiento de retroceso completo para un clúster, no podrá volver a habilitarlo para ese clúster.

Pruebas de Amazon Aurora MySQL por medio de consultas de inserción de errores

Para probar la tolerancia a errores de su clúster de base de datos de Aurora MySQL, utilice las consultas de inserción de errores. Las consultas de inyección de errores se emiten como comandos SQL a una instancia Amazon Aurora. Permiten programar una simulación de uno de los siguientes eventos:

- Un bloqueo de una instancia de base de datos de escritor o lector
- Un error de una réplica de Aurora
- Un error de disco
- Congestión del disco

Cuando una consulta de inserción de errores especifica un bloqueo, fuerza un bloqueo de la instancia de base de datos de Aurora MySQL. Las otras consultas de inserción de errores producen simulaciones de eventos de error, pero no hacen que el evento ocurra. Cuando se envía una consulta de inserción de errores, se especifica también la cantidad de tiempo que debe durar la simulación del evento de error.

Puede enviar una consulta de inserción de errores a una de las instancias de réplica de Aurora conectándose al punto de enlace de la réplica de Aurora. Para obtener más información, consulte Conexiones de puntos de conexión de Amazon Aurora.

La ejecución de consultas de inyección de errores requiere todos los privilegios del usuario maestro. Para obtener más información, consulte Privilegios de la cuenta de usuario maestro.

Prueba de un bloqueo de instancia

Puede forzar el bloqueo de una instancia de Amazon Aurora con la consulta de inserción de errores `ALTER SYSTEM CRASH`.

En esta consulta de inserción de errores no se produce una conmutación por error. Si quiere probar una conmutación por error, puede elegir la acción de instancia Failover (Conmutación por error) para

el clúster de base de datos en la consola de RDS o usar el comando failover-db-clúster de AWS CLI o la operación FailoverDBclúster de la API de RDS.

Sintaxis

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Opciones

La consulta de inserción de errores toma uno de los siguientes tipos de bloqueos:

- **`INSTANCE`**: se simula un bloqueo de la base de datos compatible con MySQL para la instancia de Amazon Aurora.
- **`DISPATCHER`**: se simula un bloqueo del distribuidor de la instancia de escritor del clúster de base de datos de Aurora. El distribuidor escribe actualizaciones en el volumen de clúster de un clúster de base de datos Amazon Aurora.
- **`NODE`**: se simula un bloqueo de la base de datos compatible con MySQL y del distribuidor para la instancia de Amazon Aurora. En esta simulación de inserción de errores también se elimina la caché.

El tipo de bloqueo predeterminado es `INSTANCE`.

Prueba de un error de una réplica de Aurora

Puede simular el error de una réplica de Aurora con la consulta de inserción de errores `ALTER SYSTEM SIMULATE READ REPLICA FAILURE`.

Un error de réplica de Aurora bloquea todas las solicitudes de la instancia del escritor realizadas a una réplica de Aurora o a todas las réplicas de Aurora del clúster de base de datos durante un intervalo de tiempo especificado. Cuando se complete el intervalo de tiempo, las réplicas de Aurora afectadas se sincronizarán automáticamente con la instancia de escritor.

Sintaxis

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE
    [ TO ALL | TO "replica name" ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opciones

Esta consulta de inserción de errores toma los siguientes parámetros:

- **`percentage_of_failure`**: el porcentaje de solicitudes que se deben bloquear durante el evento de error. Puede ser un valor doble entre 0 y 100. Si se especifica 0, no se bloquea ninguna solicitud. Si se especifica 100, se bloquean todas las solicitudes.
- Tipo de falla: el tipo de error que se va a simular. Especifique `TO ALL` para simular errores para todas las réplicas de Aurora del clúster de base de datos. Especifique `TO` y el nombre de una réplica de Aurora para simular un error de la réplica de Aurora única. El tipo de error predeterminado es `TO ALL`.
- **`quantity`**: la cantidad de tiempo que debe durar la simulación del error de la réplica de Aurora. El intervalo es una cantidad seguida por una unidad de tiempo. La simulación durará esa cantidad de la unidad especificada. Por ejemplo, `20 MINUTE` hará que la simulación se ejecute durante 20 minutos.

> **Note**
>
> Debe tener cuidado al especificar el intervalo de tiempo del evento de error de la réplica de Aurora. Si especifica un intervalo de tiempo demasiado largo y la instancia de escritor escribe una gran cantidad de datos durante el evento de error, su clúster de base de datos de Aurora es posible que asuma que la réplica de Aurora se ha bloqueado y reemplazarla.

Prueba de un error de disco

Puede simular un error de disco para un clúster de base de datos Aurora con la consulta de inserción de errores `ALTER SYSTEM SIMULATE DISK FAILURE`.

Durante la simulación de un error de disco, el clúster de base de datos de Aurora marca de forma aleatoria los segmentos de disco como defectuosos. Las solicitudes que lleguen a esos segmentos se bloquearán mientras dure la simulación.

Sintaxis

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opciones

Esta consulta de inserción de errores toma los siguientes parámetros:

- **`percentage_of_failure`**: el porcentaje del disco que se debe marcar como defectuoso durante el evento de error. Puede ser un valor doble entre 0 y 100. Si se especifica 0, ninguna parte del disco se marca como defectuosa. Si se especifica 100, todo el disco se marca como defectuoso.
- **`DISK index`**: un bloque lógico de datos concreto para el que se debe simular el evento de error. Si se sobrepasa el rango de bloques de datos lógicos disponibles, aparece un error que indica el valor máximo del índice que se puede especificar. Para obtener más información, consulte Visualización del estado del volumen para un clúster de base de datos de Aurora MySQL.
- **`NODE index`**: un nodo de almacenamiento concreto para el que se debe simular el evento de error. Si se sobrepasa el rango de nodos de almacenamiento disponibles, aparece un error que indica el valor máximo del índice que se puede especificar. Para obtener más información, consulte Visualización del estado del volumen para un clúster de base de datos de Aurora MySQL.
- **`quantity`**: la cantidad de tiempo que debe durar la simulación del error de disco. El intervalo es una cantidad seguida por una unidad de tiempo. La simulación durará esa cantidad de la unidad especificada. Por ejemplo, `20 MINUTE` hará que la simulación se ejecute durante 20 minutos.

Prueba de congestión del disco

Puede simular un error de disco para un clúster de base de datos Aurora con la consulta de inserción de errores `ALTER SYSTEM SIMULATE DISK CONGESTION`.

Durante la simulación de congestión del disco, el clúster de base de datos de Aurora marca de forma aleatoria los segmentos de disco como congestionados. Las solicitudes que lleguen a esos segmentos se retrasarán entre el mínimo especificado y el tiempo de demora máximo mientras dure la simulación.

Sintaxis

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION
    BETWEEN minimum AND maximum MILLISECONDS
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opciones

Esta consulta de inserción de errores toma los siguientes parámetros:

- **`percentage_of_failure`**: el porcentaje del disco que se debe marcar como congestionado durante el evento de error. Puede ser un valor doble entre 0 y 100. Si se especifica 0, ninguna parte del disco se marca como congestionada. Si se especifica 100, todo el disco se marca como congestionado.
- **`DISK index`** o:**`NODE index`** un disco o nodo concreto para el que se debe simular el evento de error. Si se sobrepasa el rango de índices para el disco o el nodo, aparece un error que indica el valor máximo del índice que se puede especificar.
- **`minimum`** y:**`maximum`** la cantidad mínima y máxima de demora de la congestión en milisegundos. Los segmentos de disco marcados como congestionados se retrasarán una cantidad de tiempo aleatoria del rango comprendido entre la cantidad mínima y máxima de milisegundos mientras dure la simulación.
- **`quantity`**: la cantidad de tiempo durante la que se debe simular la congestión del disco. El intervalo es una cantidad seguida por una unidad de tiempo. La simulación durará esa cantidad de la unidad de tiempo especificada. Por ejemplo, `20 MINUTE` hará que la simulación se ejecute durante 20 minutos.

Modificación de las tablas de Amazon Aurora con DDL rápido

Amazon Aurora incluye optimizaciones para ejecutar una operación `ALTER TABLE` en su lugar, casi instantáneamente. La operación se completa sin que sea necesario copiar la tabla y sin que haya un impacto perceptible en otras instrucciones DML. Como la operación no consume almacenamiento temporal para una copia de la tabla, las instrucciones DDL resultan prácticas incluso para tablas grandes en clases de instancia pequeñas.

Aurora MySQL versión 3 es compatible con la característica de MySQL 8.0 llamada DDL instantáneo. La versión 2 de Aurora MySQL utiliza una implementación diferente denominada DDL rápido.

Temas

- DDL instantáneo (Aurora MySQL versión 3)
- DDL rápida (Aurora MySQL versión 2)

DDL instantáneo (Aurora MySQL versión 3)

La optimización realizada por Aurora MySQL versión 3 para mejorar la eficiencia de algunas operaciones de DDL se denomina DDL instantáneo.

Aurora MySQL versión 3 es compatible con el DDL instantáneo de la comunidad MySQL 8.0. Realice una operación DDL instantánea mediante la cláusula `ALGORITHM=INSTANT` con la instrucción `ALTER TABLE`. Para obtener información sobre la sintaxis y el uso de DDL instantáneo, consulte ALTER TABLE (Modificar tabla) y Online DDL Operations (Operaciones DDL en línea) en la documentación de MySQL.

En los siguientes ejemplos se muestra la característica DDL instantánea. Las instrucciones `ALTER TABLE` agregan columnas y cambian los valores predeterminados de columnas. Los ejemplos incluyen columnas regulares y virtuales, así como tablas regulares y particionadas. En cada paso, puede ver los resultados, para ello, emita las instrucciones `SHOW CREATE TABLE` y `DESCRIBE`.

```
mysql> CREATE TABLE t1 (a INT, b INT, KEY(b)) PARTITION BY KEY(b) PARTITIONS 6;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t1 RENAME TO t2, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b SET DEFAULT 100, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b DROP DEFAULT, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN c ENUM('a', 'b', 'c'), ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 MODIFY COLUMN c ENUM('a', 'b', 'c', 'd', 'e'), ALGORITHM =
 INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN (d INT GENERATED ALWAYS AS (a + 1) VIRTUAL), ALGORITHM
 = INSTANT;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t2 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3 (a INT, b INT) PARTITION BY LIST(a)(
```

```
    ->   PARTITION mypart1 VALUES IN (1,3,5),
    ->   PARTITION MyPart2 VALUES IN (2,4,6)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE t3 ALTER COLUMN a SET DEFAULT 20, ALTER COLUMN b SET DEFAULT 200,
 ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t4 (a INT, b INT) PARTITION BY RANGE(a)
    ->   (PARTITION p0 VALUES LESS THAN(100), PARTITION p1 VALUES LESS THAN(1000),
    ->   PARTITION p2 VALUES LESS THAN MAXVALUE);
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE t4 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

/* Sub-partitioning example */
mysql> CREATE TABLE ts (id INT, purchased DATE, a INT, b INT)
    ->   PARTITION BY RANGE( YEAR(purchased) )
    ->     SUBPARTITION BY HASH( TO_DAYS(purchased) )
    ->     SUBPARTITIONS 2 (
    ->       PARTITION p0 VALUES LESS THAN (1990),
    ->       PARTITION p1 VALUES LESS THAN (2000),
    ->       PARTITION p2 VALUES LESS THAN MAXVALUE
    ->    );
Query OK, 0 rows affected (0.10 sec)

mysql> ALTER TABLE ts ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

DDL rápida (Aurora MySQL versión 2)

El DDL rápido en Aurora MySQL es una optimización diseñada para mejorar el rendimiento de determinados cambios de esquema, como agregar o eliminar columnas, al reducir el tiempo de inactividad y el uso de recursos. Permite que estas operaciones se realicen de manera más eficiente en comparación con los métodos DDL tradicionales.

> **Important**
>
> Actualmente, debe habilitar el modo lab de Aurora para usar DDL rápida. Para obtener información sobre cómo habilitar el modo lab, consulte Modo lab de Amazon Aurora MySQL. La optimización de DDL rápida se ha ingresado en el modo lab en la versión 2 de Aurora MySQL con el fin de mejorar la eficiencia de determinadas operaciones de DDL. En la versión 3 de Aurora MySQL, se ha interrumpido el modo lab y la DDL rápida se ha sustituido por la característica de MySQL 8.0 DDL instantáneo.

En MySQL, muchas operaciones de lenguaje de definición de datos (DDL) tienen un impacto considerable en el desempeño.

Por ejemplo, supongamos que usa una operación `ALTER TABLE` para agregar una columna a una tabla. En función del algoritmo especificado para la operación, esta puede incluir los siguientes pasos:

- Crear una copia completa de la tabla
- Crear una tabla temporal para procesar las operaciones simultáneas de lenguaje de manipulación de datos (DML)
- Reconstruir todos los índices de la tabla
- Aplicar bloqueos de tabla mientras se aplican los cambios simultáneos de DML
- Ralentizar el rendimiento DML concurrente

Este impacto en el rendimiento puede resultar especialmente difícil en entornos con tablas grandes o volúmenes altos de transacciones. DDL rápida ayuda a mitigar estos desafíos al optimizar los cambios de esquema, lo que permite operaciones más rápidas y que requieren menos recursos.

Limitaciones rápidas de DDL

Actualmente, el DDL rápido tiene las siguientes limitaciones:

- El DDL rápido solo admite la adición de columnas que puedan contener valores nulos, sin valores predeterminados, al final de una tabla existente.
- El DDL rápido no funciona para tablas particionadas.
- El DDL rápido no funciona para las tablas de InnoDB que usan el formato de fila REDUNDANT.
- El DDL rápido no funciona para las tablas con índices de búsqueda de texto completo.

- Si el tamaño máximo posible de registro para la operación DDL es demasiado grande, no se utiliza el DDL rápido. Un tamaño de registro es demasiado grande si es superior a la mitad del tamaño de la página. El tamaño máximo de un registro se computa sumando los tamaños máximos de todas las columnas. Para columnas de tamaño variable, según estándares de InnoDB, los bytes externos no se incluyen para computación.

Sintaxis DDL rápida

```
ALTER TABLE tbl_name ADD COLUMN col_name column_definition
```

Esta declaración puede usar las siguientes opciones:

- **`tbl_name`** — el nombre de la tabla que se va a modificar.
- **`col_name`** — el nombre de la columna que se va a añadir.
- **`col_definition`** — la definición de la columna que se va a añadir.

> **Note**
>
> Debe especificar una definición de columna que pueda tener valores nulos sin un valor predeterminado. De lo contrario, no se usa el DDL rápido.

Ejemplos rápidos de DDL

Los siguientes ejemplos demuestran la aceleración de las operaciones con el DDL rápido. El primer ejemplo SQL ejecuta instrucciones `ALTER TABLE` en una tabla grande sin utilizar el DDL rápido. Esta operación lleva mucho tiempo. Un ejemplo de CLI muestra cómo habilitar el DDL rápido para el clúster. Luego, otro ejemplo SQL ejecuta las mismas instrucciones `ALTER TABLE` en una tabla idéntica. Con el DDL rápido habilitado, la operación es muy rápida.

En este ejemplo se utiliza la tabla `ORDERS` del punto de referencia TPC-H, que contiene 150 millones de filas. Este clúster utiliza intencionadamente una clase de instancia relativamente pequeña para demostrar cuánto tiempo pueden tardar las instrucciones `ALTER TABLE` cuando no se puede usar el DDL rápido. En el ejemplo se crea un clon de la tabla original que contiene datos idénticos. Al comprobar la configuración de `aurora_lab_mode`, se confirma que el clúster no puede usar el DDL rápido, ya que el modo lab no está habilitado. A continuación, las instrucciones `ALTER TABLE ADD COLUMN` tardan mucho tiempo en agregar nuevas columnas al final de la tabla.

```
mysql> create table orders_regular_ddl like orders;
Query OK, 0 rows affected (0.06 sec)

mysql> insert into orders_regular_ddl select * from orders;
Query OK, 150000000 rows affected (1 hour 1 min 25.46 sec)

mysql> select @@aurora_lab_mode;
+------------------+
| @@aurora_lab_mode |
+------------------+
|                0 |
+------------------+

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (40 min 31.41 sec)

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (40 min 44.45 sec)
```

Este ejemplo realiza la misma preparación de una tabla grande que el ejemplo anterior. Sin embargo, no puede simplemente habilitar el modo de laboratorio dentro de una sesión SQL interactiva. Esa configuración debe estar habilitada en un grupo de parámetros personalizado. Para ello, es necesario salir de la `mysql` sesión y ejecutar algunos AWS comandos de la CLI o utilizar el Consola de administración de AWS.

```
mysql> create table orders_fast_ddl like orders;
Query OK, 0 rows affected (0.02 sec)

mysql> insert into orders_fast_ddl select * from orders;
Query OK, 150000000 rows affected (58 min 3.25 sec)

mysql> set aurora_lab_mode=1;
ERROR 1238 (HY000): Variable 'aurora_lab_mode' is a read only variable
```

Habilitar el modo de laboratorio para el clúster requiere algo de trabajo con un grupo de parámetros. En este ejemplo de la CLI de AWS, se utiliza un grupo de parámetros de clúster a fin de asegurarse de que todas las instancias de base de datos del clúster utilicen el mismo valor para la configuración del modo de laboratorio.

```
$ aws rds create-db-cluster-parameter-group \
  --db-parameter-group-family aurora5.7 \
```

```
    --db-cluster-parameter-group-name lab-mode-enabled-57 --description 'TBD'
$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].[ParameterName,ParameterValue]' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 0
$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --parameters ParameterName=aurora_lab_mode,ParameterValue=1,ApplyMethod=pending-
reboot
{
    "DBClusterParameterGroupName": "lab-mode-enabled-57"
}

# Assign the custom parameter group to the cluster that's going to use Fast DDL.
$ aws rds modify-db-cluster --db-cluster-identifier tpch100g \
  --db-cluster-parameter-group-name lab-mode-enabled-57
{
  "DBClusterIdentifier": "tpch100g",
  "DBClusterParameterGroup": "lab-mode-enabled-57",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.2",
  "Status": "available"
}

# Reboot the primary instance for the cluster tpch100g:
$ aws rds reboot-db-instance --db-instance-identifier instance-2020-12-22-5208
{
  "DBInstanceIdentifier": "instance-2020-12-22-5208",
  "DBInstanceStatus": "rebooting"
}

$ aws rds describe-db-clusters --db-cluster-identifier tpch100g \
  --query '*[].[DBClusterParameterGroup]' --output text
lab-mode-enabled-57

$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 1
```

En el ejemplo siguiente se muestran los pasos restantes después de que surta efecto el cambio de grupo de parámetros. Prueba la configuración de `aurora_lab_mode` para asegurarse de que el clúster puede usar el DDL rápido. Luego ejecuta las instrucciones `ALTER TABLE` para agregar columnas al final de otra tabla grande. Esta vez, las instrucciones terminan muy rápidamente.

```
mysql> select @@aurora_lab_mode;
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 1 |
+-------------------+

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (1.51 sec)

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (0.40 sec)
```

Visualización del estado del volumen para un clúster de base de datos de Aurora MySQL

En Amazon Aurora, un volumen de clúster de bases de datos se compone de un conjunto de bloques lógicos. Cada uno de esos bloques representa 10 gigabytes de almacenamiento asignado. Estos bloques se denominan grupos de protección.

Los datos de cada grupo de protección se replican en seis dispositivos de almacenamiento físicos denominados nodos de almacenamiento. Estos nodos de almacenamiento se distribuyen entre tres zonas de disponibilidad (AZ) en la región de AWS en la que reside el clúster de base de datos. A su vez, cada nodo de almacenamiento contiene uno o varios bloques lógicos de datos para el volumen del clúster de bases de datos. Para obtener más información acerca de los grupos de protección y los nodos de almacenamiento, consulte Introducing the Aurora Storage Engine en el Blog de base de datos de AWS.

Puede simular el error de un nodo de almacenamiento completo o de un único bloque lógico de datos de un nodo de almacenamiento. Para ello, use la instrucción de inserción de errores `ALTER SYSTEM SIMULATE DISK FAILURE`. Para la instrucción, especifique el valor del índice de un bloque lógico de datos o nodo de almacenamiento concreto. Sin embargo, si especifica un valor de índice mayor que el número de bloques lógicos de datos o los nodos de almacenamiento utilizados por el volumen de clúster de base de datos, la instrucción devuelve un error. Para obtener más

información acerca de las consultas de inserción de errores, vea Pruebas de Amazon Aurora MySQL por medio de consultas de inserción de errores.

Puede evitar ese error usando la instrucción `SHOW VOLUME STATUS`. La instrucción devuelve dos variables de estado de servidor, `Disks` y `Nodes`. Estas variables representan el número total de bloques lógicos de datos y nodos de almacenamiento, respectivamente, para el volumen de clúster de base de datos.

Sintaxis

```
SHOW VOLUME STATUS
```

Ejemplo

El ejemplo siguiente muestra un resultado típico de SHOW VOLUME STATUS.

```
mysql> SHOW VOLUME STATUS;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Disks         | 96    |
| Nodes         | 74    |
+---------------+-------+
```

Ajuste de Aurora MySQL

Los eventos de espera y los estados de subprocesos son importantes herramientas de ajuste para Aurora MySQL. Si puede averiguar por qué las sesiones esperan por recursos y qué están haciendo, podrá reducir mejor los cuellos de botella. Puede utilizar la información de esta sección para encontrar las posibles causas y acciones correctivas.

Amazon DevOps Guru para RDS puede determinar de forma proactiva si las bases de datos Aurora MySQL están experimentando condiciones problemáticas que podrían causar problemas mayores en el futuro. Amazon DevOps Guru para RDS publica una explicación y recomendaciones sobre las acciones correctivas en una información proactiva. Esta sección contiene información sobre los problemas comunes.

> **Important**
>
> Los eventos de espera y los estados de subprocesos de esta sección son específicos de Aurora MySQL. Utilice la información de esta sección para ajustar solo Amazon Aurora, no Amazon RDS for MySQL.
> Algunos eventos de espera en esta sección no tienen análogos en las versiones de código abierto de estos motores de base de datos. Otros eventos de espera tienen los mismos nombres que los eventos en los motores de código abierto, pero se comportan de forma diferente. Por ejemplo, el almacenamiento de Amazon Aurora funciona de forma diferente al almacenamiento de código abierto, por lo que los eventos de espera relacionados con el almacenamiento indican condiciones de recursos diferentes.

Temas

- Conceptos esenciales para el ajuste de Aurora MySQL
- Ajuste de Aurora MySQL con eventos de espera
- Ajustar Aurora MySQL con estados de subprocesos
- Ajuste de Aurora MySQL con información proactiva de Amazon DevOps Guru

Conceptos esenciales para el ajuste de Aurora MySQL

Antes de ajustar la base de datos Aurora MySQL, asegúrese de saber qué son los eventos de espera y por qué se producen. Revise también la arquitectura de disco y memoria básica de Aurora MySQL

cuando utilice el motor de almacenamiento InnoDB. Para obtener un diagrama de arquitectura útil, consulte el Manual de referencia de MySQL.

Temas

- Eventos de espera de Aurora MySQL
- Estados del subproceso Aurora MySQL
- Memoria de Aurora MySQL
- Procesos de Aurora MySQL

Eventos de espera de Aurora MySQL

Un evento de espera indica un recurso por el cual una sesión está en espera. Por ejemplo, el evento de espera `io/socket/sql/client_connection` indica que un subproceso está controlando una nueva conexión. Los recursos típicos por los que espera una sesión son los siguientes:

- Acceso de subproceso único a un búfer, por ejemplo, cuando una sesión intenta modificar un búfer
- Una fila bloqueada actualmente por otra sesión
- Lectura de un archivo de datos
- Escritura de un archivo de registro

Por ejemplo, para satisfacer una consulta, la sesión podría hacer un escaneo de tabla completo. Si los datos ya no están en la memoria, la sesión espera a que se complete la E/S del disco. Cuando los búferes se leen en la memoria, es posible que la sesión tenga que esperar porque otras sesiones tienen acceso a los mismos búferes. La base de datos registra las esperas mediante un evento de espera predefinido. Estos eventos se agrupan en categorías.

Un evento de espera no muestra por sí solo un problema de rendimiento. Por ejemplo, si los datos solicitados no están en memoria, es necesario leer los datos del disco. Si una sesión bloquea una fila para una actualización, otra sesión espera a que se desbloquee la fila para poder actualizarla. Una confirmación requiere un tiempo de espera para que se complete la escritura en un archivo de registro. Las esperas forman parte del funcionamiento normal de una base de datos.

Un gran número de eventos de espera suele mostrar un problema de rendimiento. En estos casos, se pueden utilizar los datos de los eventos de espera para determinar en qué se gastan las sesiones. Por ejemplo, si un informe que normalmente se ejecuta en minutos ahora se ejecuta durante horas, puede identificar los eventos de espera que más contribuyen al tiempo total de espera. Si puede

determinar las causas de los principales eventos de espera, a veces puede hacer cambios que mejoren el rendimiento. Por ejemplo, si la sesión se encuentra a la espera de una fila que ha sido bloqueada por otra sesión, puede terminar la sesión de bloqueo.

Estados del subproceso Aurora MySQL

Un estado de subproceso general es un valor `State` asociado al procesamiento general de consultas. Por ejemplo, el estado del subproceso `sending data` indica que un subproceso está leyendo y filtrando filas de una consulta para determinar el conjunto de resultados correcto.

Puede utilizar estados de subprocesos para ajustar Aurora MySQL de forma similar a la forma en que utiliza los eventos de espera. Por ejemplo, apariciones frecuentes de `sending data` suelen indicar que una consulta no utiliza un índice. Para obtener más información acerca de los estados de los subprocesos, consulte Estados generales de subprocesos en el Manual de referencia de MySQL.

Al utilizar Información sobre rendimiento, se cumple una de las condiciones siguientes:

- Performance Schema activado: Aurora MySQL muestra los eventos de espera en lugar del estado de subprocesos.
- Performance Schema desactivado: Aurora MySQL muestra los estados de subprocesos.

Recomendamos configurar Performance Schema para la administración automática. Performance Schema proporciona información adicional y mejores herramientas para investigar posibles problemas de rendimiento. Para obtener más información, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

Memoria de Aurora MySQL

En Aurora MySQL, las áreas de memoria más importantes son el grupo de búferes y el búfer de registro.

Temas

- Grupo de búferes

Grupo de búferes

El grupo de búferes es el área de memoria compartida en la que Aurora MySQL almacena en caché los datos de tabla e índice. Las consultas pueden acceder a los datos de uso frecuente directamente desde la memoria sin leer desde el disco.

El grupo de búferes está estructurado como una lista vinculada de páginas. Una página puede contener varias filas. Aurora MySQL utiliza un algoritmo menos usado recientemente (LRU) para determinar cuáles son las páginas del grupo más antiguas.

Para obtener más información, consulte Buffer Pool (Grupo de búferes) en el Manual de referencia de MySQL.

Procesos de Aurora MySQL

Aurora MySQL utiliza un modelo de procesos muy diferente del de Aurora PostgreSQL.

Temas

- Servidor MySQL (mysqld)
- Subprocesos
- Grupo de subprocesos

Servidor MySQL (mysqld)

El servidor MySQL es un proceso del sistema operativo único denominado mysqld. El servidor MySQL no genera procesos adicionales. Por lo tanto, las bases de datos Aurora MySQL utilizan mysqld para realizar la mayor parte de su trabajo.

Cuando se inicia el servidor MySQL, escucha las conexiones de red de los clientes de MySQL. Cuando un cliente se conecta a la base de datos, mysqld abre un subproceso.

Subprocesos

Los subprocesos del administrador de conexiones asocian cada conexión de cliente con un subproceso dedicado. Este subproceso administra la autenticación, ejecuta instrucciones y devuelve los resultados al cliente. El administrador de conexiones crea nuevos subprocesos cuando es necesario.

La caché de subprocesos es el conjunto de subprocesos disponibles. Cuando finaliza una conexión, MySQL devuelve el subproceso a la caché de subprocesos si esta no está llena. La variable del sistema `thread_cache_size` determina el tamaño de la caché de subprocesos.

Grupo de subprocesos

El grupo de subprocesos consta de varios grupos de subprocesos. Cada grupo administra un conjunto de conexiones de clientes. Cuando un cliente se conecta a la base de datos, el grupo

de subprocesos asigna las conexiones a los grupos de subprocesos de forma rotativa. El grupo de subprocesos separa las conexiones y los subprocesos. No existe una relación fija entre las conexiones y los subprocesos que ejecutan las instrucciones que se reciben de esas conexiones.

Ajuste de Aurora MySQL con eventos de espera

La siguiente tabla resume los eventos de espera de Aurora MySQL que suelen indicar problemas de rendimiento. Los siguientes eventos de espera son un subconjunto de la lista de Eventos de espera de Aurora MySQL.

Evento de espera	Descripción
cpu	Este evento ocurre cuando un subproceso está activo en la CPU o espera por la CPU.
io/aurora_redo_log_flush	Este evento se produce cuando una sesión escribe datos persistentes en el almacenam iento de Aurora.
io/aurora_respond_to_client	Este evento se produce cuando un subproces o está a la espera de devolver un conjunto de resultados a un cliente.
io/redo_log_flush	Este evento se produce cuando una sesión escribe datos persistentes en el almacenam iento de Aurora.
io/socket/sql/client_connection	Este evento se produce cuando un subproces o está controlando una nueva conexión.
io/table/sql/handler	Este evento se produce cuando el trabajo se ha delegado en un motor de almacenamiento.
synch/cond/innodb/row_lock_wait	Este evento se produce cuando una sesión ha bloqueado una fila para una actualización y otra sesión intenta actualizar la misma fila.

Evento de espera	Descripción
synch/cond/innodb/row_lock_wait_cond	Este evento se produce cuando una sesión ha bloqueado una fila para una actualización y otra sesión intenta actualizar la misma fila.
synch/cond/sql/MDL_context::COND_wait_status	Este evento se produce cuando hay subprocesos a la espera en un bloqueo de metadatos de tabla.
synch/mutex/innodb/aurora_lock_thread_slot_futex	Este evento se produce cuando una sesión ha bloqueado una fila para una actualización y otra sesión intenta actualizar la misma fila.
synch/mutex/innodb/buf_pool_mutex	Este evento se produce cuando un subproceso ha adquirido un bloqueo en el grupo de búferes de InnoDB para acceder a una página de memoria.
synch/mutex/innodb/fil_system_mutex	Este evento se produce cuando una sesión está a la espera para acceder a la memoria caché de memoria del espacio de tabla.
synch/mutex/innodb/trx_sys_mutex	Este evento se produce cuando hay una elevada actividad de la base de datos con un gran número de transacciones.
synch/sxlock/innodb/hash_table_locks	Este evento se produce cuando se deben leer desde un archivo páginas que no se encuentran en el grupo de búferes.
synch/mutex/innodb/temp_pool_manager_mutex	Este evento se produce cuando una sesión está esperando para adquirir un mutex para administrar el grupo de espacios de tablas temporales de sesión.

cpu

El evento de espera de `cpu` ocurre cuando un subproceso se encuentra activo en la CPU o en espera de la misma.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL, versiones 2 y 3

Contexto

Una conexión puede ejecutar trabajos en esta CPU para cada vCPU. En determinadas situaciones, el número de conexiones activas que están listas para ejecutarse es mayor que el número de vCPU. Este desequilibrio provoca conexiones a la espera de recursos de CPU. Si el número de conexiones activas permanece es constantemente superior al número de vCPUs, la instancia experimenta contención de CPU. La contención hace que se produzca el evento de espera `cpu`.

> **Note**
>
> La métrica de Información sobre rendimiento para la CPU es `DBLoadCPU`. El valor de `DBLoadCPU` puede diferir del valor de la métrica `CPUUtilization` de CloudWatch. Esta última métrica se recopila del hipervisor para una instancia de base de datos.

Las métricas del sistema operativo de Información sobre rendimiento proporcionan información detallada sobre la utilización de la CPU. Por ejemplo, puede mostrar las siguientes métricas:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`

- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Información sobre rendimiento informa del uso de la CPU por parte del motor de base de datos como `os.cpuUtilization.nice.avg`.

Causas probables del aumento de las esperas

Cuando este evento se produce más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas suelen ser las siguientes:

- Consultas analíticas
- Transacciones altamente concurrentes
- Transacciones de larga duración
- Aumento repentino del número de conexiones, conocido como tormenta de inicios de sesión
- Aumento del cambio de contexto

Acciones

Si el evento de espera de `cpu` domina la actividad de la base de datos, no indica necesariamente un problema de rendimiento. Responda a este evento solo cuando el rendimiento se deteriore.

Dependiendo de la causa del aumento de utilización de la CPU, considere la posibilidad de adoptar las estrategias siguientes:

- Aumente la capacidad de CPU del host. Por lo general, este enfoque solo proporciona un alivio provisional.
- Identifique las principales consultas de posible optimización.
- Redirija algunas de las cargas de trabajo de solo lectura a nodos lectores, si procede.

Temas

- Identificar las sesiones o consultas que están causando el problema
- Analizar y optimizar la elevada carga de trabajo de la CPU

Identificar las sesiones o consultas que están causando el problema

Para encontrar sesiones y consultas, consulte la tabla SQL principal en Información sobre rendimiento para obtener información sobre las instrucciones SQL que tienen la mayor carga de CPU. Para obtener más información, consulte Análisis de métricas mediante el panel de Información sobre rendimiento.

Normalmente, una o dos instrucciones SQL consumen la mayoría de los ciclos de CPU. Concentre sus esfuerzos en estas instrucciones. Supongamos que su instancia de base de datos tiene 2 vCPU con una carga media de base de datos de 3,1 sesiones activas (AAS) en el estado de la CPU. En este caso, la instancia está vinculada a la CPU. Consideremos la posibilidad de aplicar las estrategias siguientes:

- Actualizar a una clase de instancia mayor con más vCPU.
- Ajustar las consultas para reducir la carga de la CPU.

En este ejemplo, las principales consultas SQL tienen una carga de base de datos de 1,5 AAS, toda en el estado de la CPU. Otra instrucción SQL tiene una carga de 0,1 en el estado de la CPU. En este ejemplo, si detuvo la instrucción SQL de menor carga, no se reduce significativamente la carga de la base de datos. Sin embargo, si optimiza las dos consultas de alta carga para que sean el doble de eficientes, eliminará el cuello de botella de la CPU. Si reduce la carga de CPU de 1,5 AAS en un 50 por ciento, el AAS de cada instrucción se reduce a 0,75. La carga total de la base de datos en la CPU es ahora de 1,6 AAS. Este valor está por debajo del máximo de 2.0 de la vCPU.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights. Consulte también el artículo de AWS Support How can I troubleshoot and resolve high CPU utilization on my Amazon RDS for MySQL instances? (¿Cómo solucionar y resolver el problema del uso elevado de la CPU en mis instancias de Amazon RDS for MySQL?).

Analizar y optimizar la elevada carga de trabajo de la CPU

Después de identificar la consulta o las consultas que aumentan el uso de la CPU, puede optimizarlas o finalizar la conexión. En el siguiente ejemplo se muestra cómo finalizar una conexión.

```
CALL mysql.rds_kill(processID);
```

Para obtener más información, consulte mysql.rds_kill.

Si finaliza una sesión, la acción podría desencadenar una larga restauración.

Seguir las directrices para optimizar las consultas

Para optimizar las consultas, tenga en cuenta las directrices siguientes:

- Ejecute la instrucción `EXPLAIN`.

  Este comando muestra los pasos individuales necesarios para la ejecución de una consulta. Para obtener más información, consulte Optimizing Queries with EXPLAIN en la documentación de MySQL.
- Ejecute la instrucción `SHOW PROFILE`.

  Utilice esta instrucción para revisar los detalles del perfil que pueden proporcionar información sobre el uso de recursos para las instrucciones que se ejecutan durante la sesión actual. Para obtener más información, consulte SHOW PROFILE Statement en la documentación de MySQL.
- Ejecute la instrucción `ANALYZE TABLE`.

  Utilice esta instrucción para actualizar las estadísticas de índice de las tablas a las que accede la consulta de alto consumo de recursos de CPU. Al analizar la instrucción, podrá ayudar al optimizador a elegir un plan de ejecución adecuado. Para obtener más información, consulte ANALYZE TABLE Statement en la documentación de MySQL.

Siga las directrices para mejorar el uso de la CPU

Para mejorar el uso de la CPU en una instancia de base de datos, siga las directrices siguientes:

- Asegúrese de que todas las consultas utilicen índices adecuados.
- Averigüe si puede utilizar consultas paralelas de Aurora. Puede utilizar esta técnica para reducir el uso de la CPU en el nodo director con la reducción del procesamiento de la función, el filtrado de filas y la proyección de columnas para la cláusula `WHERE`.
- Averigüe si el número de ejecuciones SQL por segundo cumple los umbrales esperados.
- Averigüe si el mantenimiento del índice o la creación de nuevos índices ocupan los ciclos de CPU que necesita su carga de trabajo de producción. Programe actividades de mantenimiento fuera de los horarios de actividad pico.
- Averigüe si puede utilizar particiones para ayudar a reducir el conjunto de datos de consulta. Para obtener más información, consulte la entrada de blog How to plan and optimize Amazon Aurora with MySQL compatibility for consolidated workloads.

Verificar si hay tormentas de conexión

Si la métrica `DBLoadCPU` no es muy alta, pero la métrica `CPUUtilization` es alta, la causa del exceso de uso de recursos de la CPU se encuentra fuera del motor de base de datos. Un ejemplo clásico de ello son las tormentas de conexión.

Verifique si se cumplen las condiciones siguientes:

- Hay un aumento en la métrica `CPUUtilization` de Información sobre rendimiento y la métrica `DatabaseConnections` de Amazon CloudWatch.
- El número de subprocesos de la CPU es mayor que el número de vCPU.

Si se cumplen las condiciones anteriores, considere la posibilidad de reducir el número de conexiones de la base de datos. Por ejemplo, puede utilizar un grupo de conexiones como proxy RDS. Para obtener información sobre prácticas recomendadas para la administración y el escalado eficaz de conexiones, consulte el documento técnico Amazon Aurora MySQL DBA Handbook for Connection Management.

io/aurora_redo_log_flush

El evento `io/aurora_redo_log_flush` se produce cuando una sesión escribe datos persistentes en el almacenamiento de Amazon Aurora.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 2

Contexto

El evento `io/aurora_redo_log_flush` es para una operación de entrada/salida de escritura (E/S) en Aurora MySQL.

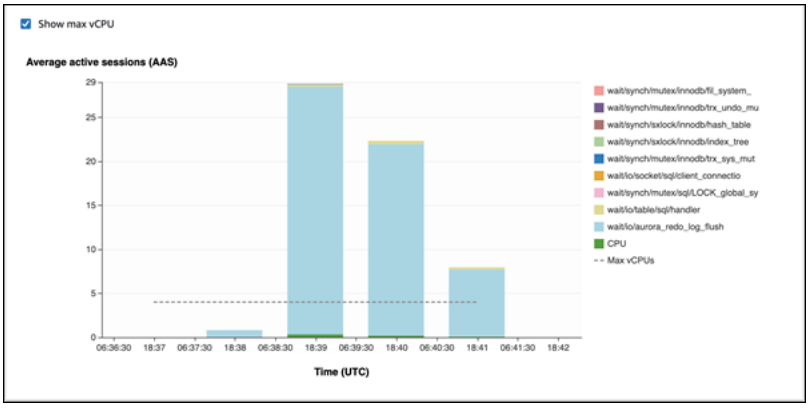
> **Note**
>
> En la versión 3 de Aurora MySQL, este evento de espera se denomina io/redo_log_flush.

Causas probables del aumento del tiempo de espera

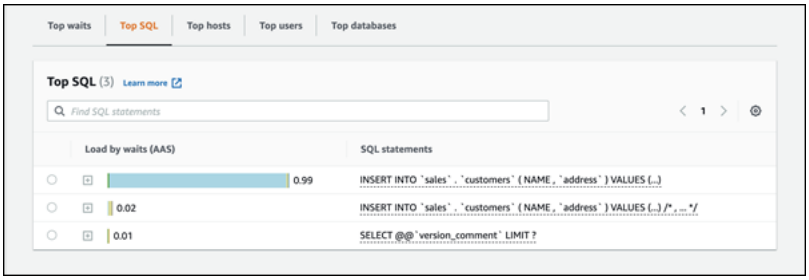
Para la persistencia de datos, las confirmaciones requieren una escritura duradera en un almacenamiento estable. Si la base de datos está realizando demasiadas confirmaciones, se produce un evento de espera en la operación de E/S de escritura, el evento de espera `io/aurora_redo_log_flush`.

En los ejemplos siguientes, se insertan 50 000 registros en un clúster de base de datos de Aurora MySQL mediante la clase de instancia de base de datos db.r5.xlarge:

- En el primer ejemplo, cada sesión inserta 10 000 registros fila por fila. De forma predeterminada, si un comando de lenguaje de manipulación de datos (DML) no se encuentra dentro de una transacción, Aurora MySQL utiliza confirmaciones implícitas. La opción Autocommit (Confirmar automáticamente) está activada. Esto significa que para cada inserción de fila hay una confirmación. Información sobre rendimiento indica que las conexiones pasan la mayor parte del tiempo esperando en el evento de espera `io/aurora_redo_log_flush`.

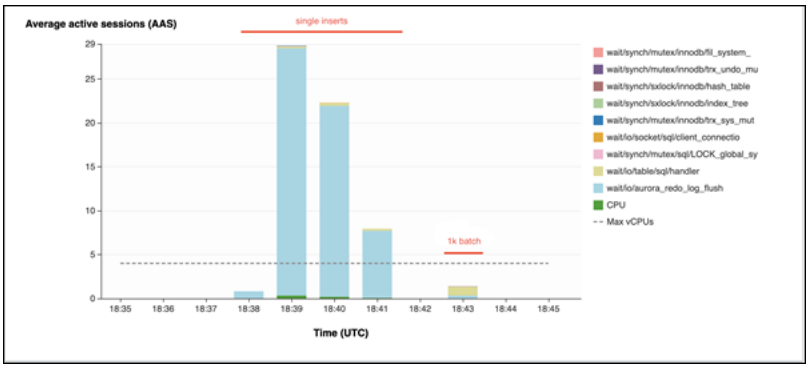


  Esto se debe al uso de las instrucciones de inserción sencillas.



Los 50 000 registros tardan 3,5 minutos en insertarse.

- En el segundo ejemplo, las inserciones se realizan en lotes de 1000 lotes, es decir, cada conexión realiza 10 confirmaciones en lugar de 10 000. Información sobre rendimiento indica que las conexiones no pasan la mayor parte del tiempo en el evento de espera `io/aurora_redo_log_flush`.



Los 50 000 registros tardan 4 segundos en insertarse.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas problemáticas
- Agrupar sus operaciones de escritura
- Desactivar la confirmación automática
- Utilizar transacciones
- Utilizar lotes

Identificar las sesiones y consultas problemáticas

Si su instancia de base de datos tiene un cuello de botella, la primera tarea que debe realizar es buscar las sesiones y consultas que lo provocan. Para ver una entrada de blog útil sobre AWS Database, consulte Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Para identificar sesiones y consultas que provocan un cuello de botella

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información de rendimiento.
3. Seleccione la instancia de base de datos.
4. En Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

   Las consultas de la parte superior de la lista son las que provocan la mayor carga de la base de datos.

Agrupar sus operaciones de escritura

Los ejemplos siguientes desencadenan el evento de espera `io/aurora_redo_log_flush`. (La opción Autocommit [Confirmar automáticamente] está activada).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Para reducir el tiempo de espera en el evento de espera `io/aurora_redo_log_flush`, agrupe sus operaciones de escritura de forma lógica en una única confirmación para reducir las llamadas persistentes al almacenamiento.

Desactivar la confirmación automática

Desactive la confirmación automática antes de realizar grandes cambios que no están dentro de una transacción, tal como se muestra en el ejemplo siguiente.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Utilizar transacciones

Puede utilizar transacciones, tal como se muestra en el ejemplo siguiente.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizar lotes

También puede realizar cambios en lotes, tal como se muestra en el siguiente ejemplo. Sin embargo, el uso de lotes demasiado grandes puede provocar problemas de rendimiento, sobre todo en réplicas de lectura o cuando se realiza una recuperación a un momento dado (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/aurora_respond_to_client

El evento `io/aurora_respond_to_client` se produce cuando un subproceso está a la espera de devolver un conjunto de resultados a un cliente.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 2

Contexto

El evento `io/aurora_respond_to_client` indica que un subproceso está a la espera de devolver un conjunto de resultados a un cliente.

El procesamiento de consultas se ha completado y los resultados se devuelven al cliente de la aplicación. Sin embargo, dado que no hay suficiente ancho de banda de red en el clúster de base de datos, un subproceso está a la espera para devolver el conjunto de resultados.

Causas probables del aumento de las esperas

Cuando el evento `io/aurora_respond_to_client` aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

Instancia de base de datos insuficiente para la carga de trabajo

La clase de instancia de base de datos que utiliza el clúster de base de datos no tiene el ancho de banda de red necesario para procesar la carga de trabajo de forma eficiente.

Grandes conjuntos de resultados

Se ha producido un aumento en el tamaño del conjunto de resultados que se devuelve porque la consulta devuelve un mayor número de filas. El conjunto de resultados de gran tamaño consume más ancho de banda de red.

Aumento de la carga en el cliente

Puede haber presión de la CPU, presión de memoria o saturación en el cliente. Un aumento de la carga en el cliente retrasa la recepción de los datos del clúster de la base de datos de Aurora MySQL.

Aumento de la latencia de la red

Puede haber un aumento de la latencia de la red entre el clúster de la base de datos de Aurora MySQL y el cliente. Una mayor latencia de la red aumenta el tiempo necesario para que el cliente reciba los datos.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas que provocan los eventos
- Escalar la clase de instancia de base de datos
- Verificar la carga de trabajo en busca de resultados inesperados
- Distribuir la carga de trabajo con instancias de lector
- Utilizar el modificador SQL_BUFFER_RESULT

Identificar las sesiones y consultas que provocan los eventos

Puede utilizar Información sobre rendimiento para mostrar las consultas que bloqueó el evento de espera `io/aurora_respond_to_client`. Normalmente, las bases de datos con una carga de moderada a significativa tienen eventos de espera. Los eventos de espera pueden ser aceptables si el rendimiento es óptimo. Si el rendimiento no es óptimo, examine dónde pasa más tiempo la base de datos. Preste atención a los eventos de espera que contribuyen a la carga más alta y averigüe si puede optimizar la base de datos y la aplicación para reducirlos.

Para buscar consultas SQL responsables de cargas elevadas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información sobre rendimiento.
3. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

   El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog de AWS Database Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Escalar la clase de instancia de base de datos

Verifique el aumento del valor de las métricas de Amazon CloudWatch relacionadas con el rendimiento de la red, como, por ejemplo, `NetworkReceiveThroughput` y `NetworkTransmitThroughput`. Si se está llegando al límite de ancho de banda de red de la clase de instancia de base de datos, puede escalar la clase de instancia de base de datos que utiliza el clúster de base de datos modificando el clúster de base de datos. Una clase de instancia de base de datos con mayor ancho de banda de red devuelve datos a los clientes de forma más eficiente.

Para obtener más información acerca del monitoreo de métricas de Amazon CloudWatch, consulte Consulta de métricas en la consola de Amazon RDS. Para obtener información acerca de las clases

de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora. Para obtener más información acerca de la modificación de un clúster de bases de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Verificar la carga de trabajo en busca de resultados inesperados

Verifique la carga de trabajo en el clúster de base de datos y asegúrese de que no produzca resultados inesperados. Por ejemplo, puede haber consultas que devuelvan un mayor número de filas que el esperado. En este caso, puede utilizar las métricas de contador de Información sobre rendimiento, como, por ejemplo, `Innodb_rows_read`. Para obtener más información, consulte Métricas de contador de Información de rendimiento.

Distribuir la carga de trabajo con instancias de lector

Puede distribuir la carga de trabajo de solo lectura con réplicas de Aurora. Puede escalar horizontalmente con la incorporación de más réplicas de Aurora. Si lo hace, se pueden incrementar los límites de limitación controlada del ancho de banda de la red. Para obtener más información, consulte Clústeres de base de datos de Amazon Aurora.

Utilizar el modificador SQL_BUFFER_RESULT

Puede agregar el modificador `SQL_BUFFER_RESULT` a las instrucciones `SELECT` para forzar el resultado a una tabla temporal antes de que se devuelva al cliente. Este modificador puede ayudar con problemas de rendimiento cuando los bloqueos InnoDB no se liberan porque las consultas se encuentran en estado de espera `io/aurora_respond_to_client`. Para obtener más información, consulte SELECT Statement en la documentación de MySQL.

io/redo_log_flush

El evento `io/redo_log_flush` se produce cuando una sesión escribe datos persistentes en el almacenamiento de Amazon Aurora.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 3

Contexto

El evento `io/redo_log_flush` es para una operación de entrada/salida de escritura (E/S) en Aurora MySQL.

> **Note**
>
> En la versión 2 de Aurora MySQL, este evento de espera se denomina io/aurora_redo_log_flush.

Causas probables del aumento del tiempo de espera

Para la persistencia de datos, las confirmaciones requieren una escritura duradera en un almacenamiento estable. Si la base de datos está realizando demasiadas confirmaciones, se produce un evento de espera en la operación de E/S de escritura, el evento de espera `io/redo_log_flush`.

Para ver ejemplos del comportamiento de este evento de espera, consulte io/aurora_redo_log_flush.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas problemáticas
- Agrupar sus operaciones de escritura
- Desactivar la confirmación automática
- Utilizar transacciones
- Utilizar lotes

Identificar las sesiones y consultas problemáticas

Si su instancia de base de datos tiene un cuello de botella, la primera tarea que debe realizar es buscar las sesiones y consultas que lo provocan. Para ver una entrada de blog útil sobre AWS Database, consulte Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Para identificar sesiones y consultas que provocan un cuello de botella

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información de rendimiento.
3. Seleccione la instancia de base de datos.
4. En Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

   Las consultas de la parte superior de la lista son las que provocan la mayor carga de la base de datos.

Agrupar sus operaciones de escritura

Los ejemplos siguientes desencadenan el evento de espera `io/redo_log_flush`. (La opción Autocommit [Confirmar automáticamente] está activada).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Para reducir el tiempo de espera en el evento de espera `io/redo_log_flush`, agrupe sus operaciones de escritura de forma lógica en una única confirmación para reducir las llamadas persistentes al almacenamiento.

Desactivar la confirmación automática

Desactive la confirmación automática antes de realizar grandes cambios que no están dentro de una transacción, tal como se muestra en el ejemplo siguiente.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Utilizar transacciones

Puede utilizar transacciones, tal como se muestra en el ejemplo siguiente.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizar lotes

También puede realizar cambios en lotes, tal como se muestra en el siguiente ejemplo. Sin embargo, el uso de lotes demasiado grandes puede provocar problemas de rendimiento, sobre todo en réplicas de lectura o cuando se realiza una recuperación a un momento dado (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/socket/sql/client_connection

El evento `io/socket/sql/client_connection` se produce cuando un subproceso está en proceso de controlar una nueva conexión.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL, versiones 2 y 3

Contexto

El evento `io/socket/sql/client_connection` indica que mysqld está ocupado con la creación de subprocesos para controlar las nuevas conexiones de clientes entrantes. En este escenario, el procesamiento del responder a las nuevas solicitudes de conexión de clientes se ralentiza mientras las conexiones esperan a que se asigne el subproceso. Para obtener más información, consulte Servidor MySQL (mysqld).

Causas probables del aumento de las esperas

Cuando este evento aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

- Se produce un aumento repentino de las nuevas conexiones de usuario desde la aplicación a la instancia de Amazon RDS.
- La instancia de base de datos no puede procesar nuevas conexiones porque la red, la CPU o la memoria tienen una limitación controlada.

Acciones

Si `io/socket/sql/client_connection` domina la actividad de la base de datos, no indica necesariamente un problema de rendimiento. En una base de datos que no está inactiva, siempre hay un evento de espera activo. Actúe solo cuando el rendimiento se vea reducido. Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas problemáticas
- Seguir las prácticas recomendadas de administración de conexiones
- Escalar verticalmente la instancia si se están limitando de forma controlada los recursos
- Verificar los principales hosts y usuarios
- Consultar las tablas performance_schema
- Verificar los estados de los subprocesos de sus consultas
- Auditar las solicitudes y consultas
- Agrupar las conexiones de base de datos

Identificar las sesiones y consultas problemáticas

Si su instancia de base de datos tiene un cuello de botella, la primera tarea que debe realizar es buscar las sesiones y consultas que lo provocan. Para ver una entrada de blog útil, consulte Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Para identificar sesiones y consultas que provocan un cuello de botella

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, seleccione Información de rendimiento.
3. Seleccione la instancia de base de datos.
4. En Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

   Las consultas de la parte superior de la lista son las que provocan la mayor carga de la base de datos.

Seguir las prácticas recomendadas de administración de conexiones

Para administrar sus conexiones, tenga en cuenta las siguientes estrategias:

- Utilice la agrupación de conexiones.

  Puede aumentar gradualmente el número de conexiones según sea necesario. Para obtener más información, consulte el documento técnico Amazon Aurora MySQL Database Administrator's Handbook.
- Utilice un nodo lector para redistribuir el tráfico de solo lectura.

  Para obtener más información, consulte Réplicas de Aurora y Conexiones de puntos de conexión de Amazon Aurora.

Escalar verticalmente la instancia si se están limitando de forma controlada los recursos

Busque ejemplos de limitación controlada en los siguientes recursos:

- CPU

  Verifique las métricas de Amazon CloudWatch para detectar usos elevados de la CPU.
- Network

  Verifique el aumento del valor de las métricas de CloudWatch `network receive throughput` y `network transmit throughput`. Si la instancia ha alcanzado el límite de ancho de banda de red para la clase de instancia, considere la posibilidad de escalar verticalmente la instancia de RDS a un tipo de clase de instancia superior. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.
- Memoria que se puede liberar

Verifique si hay una caída en la métrica de CloudWatch `FreeableMemory`. Considere, además, la posibilidad de activar el monitoreo mejorado. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.

Verificar los principales hosts y usuarios

Utilice Información sobre rendimiento para verificar los principales hosts y usuarios. Para obtener más información, consulte Análisis de métricas mediante el panel de Información sobre rendimiento.

Consultar las tablas performance_schema

Para obtener un recuento preciso de las conexiones actuales y totales, consulte las tablas de `performance_schema`. Con esta técnica, podrá identificar el host o usuario de origen responsable de crear un gran número de conexiones. Por ejemplo, consulte las tablas `performance_schema` como se muestra a continuación.

```
SELECT * FROM performance_schema.accounts;
SELECT * FROM performance_schema.users;
SELECT * FROM performance_schema.hosts;
```

Verificar los estados de los subprocesos de sus consultas

Si su problema de rendimiento continúa, verifique los estados de los subprocesos de sus consultas. En el cliente `mysql`, ejecute el siguiente comando.

```
show processlist;
```

Auditar las solicitudes y consultas

Para verificar la naturaleza de las solicitudes y consultas de las cuentas de usuario, utilice la auditoría avanzada de Aurora MySQL. Para obtener información sobre cómo activar la auditoría, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.

Agrupar las conexiones de base de datos

Considere la posibilidad de utilizar Amazon RDS Proxy para la administración de conexiones. Con el proxy de RDS puede permitir a las aplicaciones agrupar y compartir conexiones de base de datos para mejorar su capacidad de escala. RDS Proxy hace que las aplicaciones sean más resistentes a los errores de base de datos al conectarse automáticamente a una instancia de base de datos en

espera mientras se preservan las conexiones de las aplicaciones. Para obtener más información, consulte Amazon RDS Proxy para Aurora.

io/table/sql/handler

El evento `io/table/sql/handler` se produce cuando el trabajo se ha delegado en un motor de almacenamiento.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL, versiones 2 y 3

Contexto

El evento `io/table` indica una espera para acceder a una tabla. Este evento se produce independientemente de si los datos se almacenan en caché en el grupo de búferes o si se accede en el disco. El evento `io/table/sql/handler` indica un aumento de la actividad de la carga de trabajo.

Un controlador es una rutina especializada en un determinado tipo de datos o centrada en determinadas tareas especiales. Por ejemplo, un controlador de eventos recibe y procesa eventos y señales del sistema operativo o de una interfaz de usuario. Un controlador de memoria realiza tareas relacionadas con la memoria. Un controlador de entrada de archivos es una función que recibe una entrada de archivos y realiza tareas especiales en los datos, en función del contexto.

Vistas como, por ejemplo, `performance_schema.events_waits_current`, a menudo muestran `io/table/sql/handler` cuando la espera real es un evento de espera anidado como un bloqueo. Cuando la espera real no es `io/table/sql/handler`, Información sobre rendimiento informa del evento de espera anidado. Cuando Información sobre rendimiento informa `io/table/sql/handler`, representa el procesamiento de la solicitud de E/S por parte de InnoDB y no un evento

de espera anidado oculto. Para obtener más información, consulte Performance Schema Atom and Molecule Events en MySQL Reference Manual.

El evento `io/table/sql/handler` a menudo aparece en los principales eventos de espera con esperas de E/S como `io/aurora_redo_log_flush`.

Causas probables del aumento del tiempo de espera

En Información sobre rendimiento, los picos repentinos en el evento `io/table/sql/handler` indican un aumento de la actividad de la carga de trabajo. El aumento de la actividad significa un aumento de E/S.

Información sobre rendimiento filtra los ID de eventos de anidamiento y no informa de ninguna espera de `io/table/sql/handler` cuando el evento anidado subyacente sea una espera de bloqueo. Por ejemplo, si el evento de causa raíz es synch/mutex/innodb/aurora_lock_thread_slot_futex, Información sobre rendimiento muestra esta espera en los eventos de espera principales y no `io/table/sql/handler`.

En vistas como, por ejemplo, `performance_schema.events_waits_current`, las esperas de `io/table/sql/handler` a menudo aparecen cuando la espera real es un evento de espera anidado como un bloqueo. Cuando la espera real difiere de `io/table/sql/handler`, Información sobre rendimiento busca la espera anidada e informa de la espera real en lugar de `io/table/sql/handler`. Cuando Información sobre rendimiento informa de `io/table/sql/handler`, la espera real es `io/table/sql/handler` y no un evento de espera anidado oculto. Para obtener más información, consulte Performance Schema Atom and Molecule Events en MySQL 5.7 Reference Manual.

Acciones

Si este evento de espera domina la actividad de la base de datos, no indica necesariamente un problema de rendimiento. Cuando la base de datos está activa, siempre hay un evento de espera activo. Solo debe actuar cuando el rendimiento se vea reducido.

Recomendamos diferentes acciones en función de los demás eventos de espera que vea.

Temas

- Identificar las sesiones y consultas que provocan los eventos
- Verificar la correlación con las métricas de contador de Información sobre rendimiento
- Verificar otros eventos de espera correlacionados

Identificar las sesiones y consultas que provocan los eventos

Normalmente, las bases de datos con una carga de moderada a significativa tienen eventos de espera. Los eventos de espera pueden ser aceptables si el rendimiento es óptimo. Si el rendimiento no es óptimo, examine dónde pasa más tiempo la base de datos. Preste atención a los eventos de espera que contribuyen a la carga más alta y averigüe si puede optimizar la base de datos y la aplicación para reducirlos.

Para buscar consultas SQL responsables de cargas elevadas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información sobre rendimiento.
3. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

   El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Verificar la correlación con las métricas de contador de Información sobre rendimiento

Verifique si hay métricas de contador de Información sobre rendimiento como `Innodb_rows_changed`. Si las métricas de contador están correlacionadas con `io/table/sql/handler`, siga los pasos que se indican a continuación:

1. En Información sobre rendimiento, busque las instrucciones SQL responsables del evento de espera principal `io/table/sql/handler`. Si es posible, optimice esta instrucción para que devuelva menos filas.
2. Recupere las tablas principales de las vistas `schema_table_statistics` y `x $schema_table_statistics`. Estas vistas muestran la cantidad de tiempo que empleó

la tabla. Para obtener más información, consulte The schema_table_statistics and x $schema_table_statistics Views en el MySQL Reference Manual.

De forma predeterminada, las filas se ordenan por tiempo de espera total descendente. Las tablas con más contención se muestran en primer lugar. La salida indica si el tiempo se dedica a lecturas, escrituras, recuperaciones, inserciones, actualizaciones o eliminaciones.

```
mysql> select * from sys.schema_table_statistics limit 1\G

*************************** 1. row ***************************
     table_schema: read_only_db
       table_name: sbtest41
    total_latency: 54.11 m
     rows_fetched: 6001557
    fetch_latency: 39.14 m
    rows_inserted: 14833
   insert_latency: 5.78 m
     rows_updated: 30470
   update_latency: 5.39 m
     rows_deleted: 14833
   delete_latency: 3.81 m
 io_read_requests: NULL
          io_read: NULL
  io_read_latency: NULL
io_write_requests: NULL
         io_write: NULL
 io_write_latency: NULL
 io_misc_requests: NULL
  io_misc_latency: NULL
1 row in set (0.11 sec)
```

Verificar otros eventos de espera correlacionados

Si `synch/sxlock/innodb/btr_search_latch` y `io/table/sql/handler` contribuyen más a la anomalía de carga de la base de datos, verifique si la variable `innodb_adaptive_hash_index` está activada. Si lo está, considere la posibilidad de aumentar el valor del parámetro `innodb_adaptive_hash_index_parts`.

Si el índice hash adaptativo está desactivado, considere la posibilidad de activarlo. Para obtener más información acerca del índice hash adaptativo, consulte los siguientes recursos:

- Artículo Is Adaptive Hash Index in InnoDB right for my workload? en el sitio web de Percona.
- Adaptive Hash Index en el MySQL Reference Manual.
- Artículo Contention in MySQL InnoDB: Useful Info From the Semaphores Section en el sitio web de Percona.

> **Note**
>
> El índice hash adaptativo no se admite en instancias de base de datos de lector de Aurora. En algunos casos, el rendimiento puede ser deficiente en una instancia lectora cuando `synch/sxlock/innodb/btr_search_latch` y `io/table/sql/handler` son dominantes. Si es así, considere la posibilidad de redirigir la carga de trabajo temporalmente a la instancia de base de datos del escritor y active el índice hash adaptativo.

synch/cond/innodb/row_lock_wait

El evento `synch/cond/innodb/row_lock_wait` se produce cuando una sesión ha bloqueado una fila para una actualización y otra sesión intenta actualizar la misma fila. Para obtener información, consulte InnoDB Locking en la documentación de MySQL.

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 3

Causas probables del aumento del tiempo de espera

Las múltiples instrucciones de lenguaje de manipulación de datos (DML) tienen acceso a la misma fila o filas simultáneamente.

Acciones

Recomendamos diferentes acciones en función de los demás eventos de espera que vea.

Temas

- Buscar y responder a las instrucciones SQL responsables de este evento de espera

- Buscar y responder a la sesión de bloqueo

Buscar y responder a las instrucciones SQL responsables de este evento de espera

Utilice Información sobre rendimiento para identificar las instrucciones SQL responsables de este evento de espera. Consideremos la posibilidad de aplicar las estrategias siguientes:

- Si los bloqueos de fila son un problema persistente, considere la posibilidad de reescribir la aplicación para utilizar un bloqueo optimista.
- Utilice instrucciones de varias filas.
- Distribuya la carga de trabajo en distintos objetos de base de datos. Puede hacerlo mediante una partición.
- Verifique el valor del parámetro `innodb_lock_wait_timeout`. Controla cuánto tiempo esperan las transacciones antes de generar un error de tiempo de espera.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Buscar y responder a la sesión de bloqueo

Determine si la sesión de bloqueo está inactiva o activa. Además, averigüe si la sesión procede de una aplicación o de un usuario activo.

Para identificar la sesión que mantiene el candado, puede ejecutar `SHOW ENGINE INNODB STATUS`. A continuación se muestra un resultado de ejemplo.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 1688153, ACTIVE 82 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 4244, OS thread handle 70369524330224, query id 4020834 172.31.14.179
 reinvent executing
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 24 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 4 n bits 72 index GEN_CLUST_INDEX of table test.t1 trx
 id 1688153 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

O bien, puede utilizar la siguiente consulta para extraer detalles sobre los bloqueos actuales.

```
mysql> SELECT p1.id waiting_thread,
    p1.user waiting_user,
    p1.host waiting_host,
    it1.trx_query waiting_query,
    ilw.requesting_engine_transaction_id waiting_transaction,
    ilw.blocking_engine_lock_id blocking_lock,
    il.lock_mode blocking_mode,
    il.lock_type blocking_type,
    ilw.blocking_engine_transaction_id blocking_transaction,
    CASE it.trx_state
        WHEN 'LOCK WAIT'
        THEN it.trx_state
        ELSE p.state end blocker_state,
    concat(il.object_schema,'.', il.object_name) as locked_table,
    it.trx_mysql_thread_id blocker_thread,
    p.user blocker_user,
    p.host blocker_host
FROM performance_schema.data_lock_waits ilw
JOIN performance_schema.data_locks il
ON ilw.blocking_engine_lock_id = il.engine_lock_id
AND ilw.blocking_engine_transaction_id = il.engine_transaction_id
JOIN information_schema.innodb_trx it
ON ilw.blocking_engine_transaction_id = it.trx_id join information_schema.processlist p
ON it.trx_mysql_thread_id = p.id join information_schema.innodb_trx it1
ON ilw.requesting_engine_transaction_id = it1.trx_id join
 information_schema.processlist p1
ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
waiting_thread: 4244
waiting_user: reinvent
waiting_host: 123.456.789.012:18158
waiting_query: select id1 from test.t1 where id1=1 for update
waiting_transaction: 1688153
blocking_lock: 70369562074216:11:4:2:70369549808672
blocking_mode: X
blocking_type: RECORD
blocking_transaction: 1688142
blocker_state: User sleep
locked_table: test.t1
blocker_thread: 4243
blocker_user: reinvent
```

```
blocker_host: 123.456.789.012:18156
1 row in set (0.00 sec)
```

Cuando identifique la sesión, puede seguir una de las opciones siguientes:

- Ponerse en contacto con el propietario o el usuario de la aplicación.
- Si la sesión de bloqueo está inactiva, considere la posibilidad de finalizar la sesión de bloqueo. Esta acción podría desencadenar una larga restauración. Para aprender a finalizar una sesión, consulte Finalización de una sesión o una consulta.

Para obtener más información acerca de cómo identificar las transacciones de bloqueo, consulte Using InnoDB Transaction and Locking Information en la documentación de MySQL.

synch/cond/innodb/row_lock_wait_cond

El evento `synch/cond/innodb/row_lock_wait_cond` se produce cuando una sesión ha bloqueado una fila para una actualización y otra sesión intenta actualizar la misma fila. Para obtener información, consulte InnoDB Locking en la documentación de MySQL.

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 2

Causas probables del aumento del tiempo de espera

Las múltiples instrucciones de lenguaje de manipulación de datos (DML) tienen acceso a la misma fila o filas simultáneamente.

Acciones

Recomendamos diferentes acciones en función de los demás eventos de espera que vea.

Temas

- Buscar y responder a las instrucciones SQL responsables de este evento de espera
- Buscar y responder a la sesión de bloqueo

Buscar y responder a las instrucciones SQL responsables de este evento de espera

Utilice Información sobre rendimiento para identificar las instrucciones SQL responsables de este evento de espera. Consideremos la posibilidad de aplicar las estrategias siguientes:

- Si los bloqueos de fila son un problema persistente, considere la posibilidad de reescribir la aplicación para utilizar un bloqueo optimista.
- Utilice instrucciones de varias filas.
- Distribuya la carga de trabajo en distintos objetos de base de datos. Puede hacerlo mediante una partición.
- Verifique el valor del parámetro `innodb_lock_wait_timeout`. Controla cuánto tiempo esperan las transacciones antes de generar un error de tiempo de espera.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Buscar y responder a la sesión de bloqueo

Determine si la sesión de bloqueo está inactiva o activa. Además, averigüe si la sesión procede de una aplicación o de un usuario activo.

Para identificar la sesión que mantiene el candado, puede ejecutar `SHOW ENGINE INNODB STATUS`. A continuación se muestra un resultado de ejemplo.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 2771110, ACTIVE 112 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 24, OS thread handle 70369573642160, query id 13271336 172.31.14.179
 reinvent Sending data
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 43 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 3 n bits 0 index GEN_CLUST_INDEX of table test.t1 trx
 id 2771110 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

O bien, puede utilizar la siguiente consulta para extraer detalles sobre los bloqueos actuales.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
```

```
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Cuando identifique la sesión, puede seguir una de las opciones siguientes:

- Ponerse en contacto con el propietario o el usuario de la aplicación.
- Si la sesión de bloqueo está inactiva, considere la posibilidad de finalizar la sesión de bloqueo. Esta acción podría desencadenar una larga restauración. Para aprender a finalizar una sesión, consulte Finalización de una sesión o una consulta.

Para obtener más información acerca de cómo identificar las transacciones de bloqueo, consulte Using InnoDB Transaction and Locking Information en la documentación de MySQL.

synch/cond/sql/MDL_context::COND_wait_status

El evento `synch/cond/sql/MDL_context::COND_wait_status` se produce cuando hay subprocesos a la espera en un bloqueo de metadatos de tabla.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL, versiones 2 y 3

Contexto

El evento `synch/cond/sql/MDL_context::COND_wait_status` indica que hay subprocesos a la espera en un bloqueo de metadatos de tabla. En determinados casos, una sesión mantiene un bloqueo de metadatos en una tabla mientras otra sesión intenta adquirir el mismo bloqueo en

la misma tabla. En tal caso, la segunda sesión espera en el evento de espera `synch/cond/sql/MDL_context::COND_wait_status`.

MySQL utiliza el bloqueo de metadatos para administrar el acceso simultáneo a los objetos de base de datos y garantizar la coherencia de los datos. El bloqueo de metadatos se aplica a tablas, esquemas, eventos programados, espacios de tabla y bloqueos de usuario adquiridos con la función `get_lock` y programas almacenados. Los programas almacenados incluyen procedimientos, funciones y desencadenadores. Para obtener más información, consulte Metadata locking en la documentación de MySQL.

La lista de procesos de MySQL muestra esta sesión en el estado `waiting for metadata lock`. En Información sobre rendimiento, si `Performance_schema` está activado, el evento `synch/cond/sql/MDL_context::COND_wait_status` aparece.

El tiempo de espera predeterminado de una consulta a la espera de un bloqueo de metadatos se basa en el valor del parámetro `lock_wait_timeout`, que por defecto es 31 536 000 segundos (365 días).

Para obtener más información sobre los distintos bloqueos de InnoDB y los tipos de bloqueos que pueden provocar conflictos, consulte InnoDB Locking en la documentación de MySQL.

Causas probables del aumento de las esperas

Cuando el evento `synch/cond/sql/MDL_context::COND_wait_status` aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

Transacciones de larga duración

Una o varias transacciones están modificando una gran cantidad de datos y mantienen bloqueos en las tablas durante mucho tiempo.

Transacciones inactivas

Una o más transacciones permanecen abiertas durante mucho tiempo, sin confirmarse o revertirse.

Instrucciones DDL en tablas de gran tamaño

Una o varias instrucciones de definición de datos (DDL), como, por ejemplo, los comandos `ALTER TABLE`, se ejecutaron en tablas de gran tamaño.

Bloqueos de tabla explícitos

Hay bloqueos explícitos en tablas que no se están lanzando a tiempo. Por ejemplo, una aplicación puede ejecutar instrucciones `LOCK TABLE` de manera incorrecta.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera y de la versión del clúster de Aurora MySQL DB.

Temas

- Identificar las sesiones y consultas que provocan los eventos
- Verificar eventos pasados
- Ejecutar consultas en Aurora MySQL versión 2
- Responder a la sesión de bloqueo

Identificar las sesiones y consultas que provocan los eventos

Puede utilizar Información sobre rendimiento para mostrar las consultas que bloqueó el evento de espera `synch/cond/sql/MDL_context::COND_wait_status`. Sin embargo, para identificar la sesión de bloqueo, consulte las tablas de metadatos desde `performance_schema` y `information_schema` en el clúster de base de datos.

Normalmente, las bases de datos con una carga de moderada a significativa tienen eventos de espera. Los eventos de espera pueden ser aceptables si el rendimiento es óptimo. Si el rendimiento no es óptimo, examine dónde pasa más tiempo la base de datos. Preste atención a los eventos de espera que contribuyen a la carga más alta y averigüe si puede optimizar la base de datos y la aplicación para reducirlos.

Para buscar consultas SQL responsables de cargas elevadas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información sobre rendimiento.
3. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).

5. En la parte inferior de la página, elija Top SQL (SQL principal).

   El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog de AWS Database Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Verificar eventos pasados

Puede obtener información sobre este evento de espera para verificar si hay ocurrencias anteriores del mismo. Para ello, complete las acciones siguientes:

- Verifique el lenguaje de manipulación de datos (DML) y el rendimiento y la latencia de DDL para ver si se han producido cambios en la carga de trabajo.

  Puede utilizar Información sobre rendimiento para buscar las consultas que están a la espera en este evento en el momento del problema. Además, puede ver el resumen de las consultas que se ejecutan cerca del momento del problema.
- Si los registros de auditoría o los registros generales están activados para el clúster de base de datos, puede verificar si se ejecutan todas las consultas en los objetos (schema.table) involucrados en la transacción en espera. También puede verificar si se han completado las consultas que se ejecutaron antes de la transacción.

La información disponible para solucionar problemas de eventos pasados es limitada. La realización de estas verificaciones no muestra qué objeto está a la espera de información. Sin embargo, puede identificar tablas con cargas pesadas en el momento en que se produjo el evento y el conjunto de filas operadas con frecuencia que provocan conflictos en el momento del problema. A continuación, podrá utilizar esta información para reproducir el problema en un entorno de prueba y proporcionar información sobre su causa.

Ejecutar consultas en Aurora MySQL versión 2

En Aurora MySQL versión 2, puede identificar la sesión bloqueada directamente con la consulta de tablas `performance_schema` o vistas de esquema `sys`. Un ejemplo puede ilustrar cómo consultar tablas para identificar consultas y sesiones de bloqueo.

En el siguiente resultado de la lista de procesos, el ID de conexión 89 está a la espera en un bloqueo de metadatos y ejecuta el comando `TRUNCATE TABLE`. En una consulta sobre las tablas `performance_schema` o las vistas de esquema `sys`, el resultado muestra que la sesión de bloqueo es 76.

```
MySQL [(none)]> select @@version, @@aurora_version;
+-----------+------------------+
| @@version | @@aurora_version |
+-----------+------------------+
| 5.7.12    | 2.11.5           |
+-----------+------------------+
1 row in set (0.01 sec)

MySQL [(none)]> show processlist;
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+------------------------------+
| Id | User            | Host               | db        | Command | Time | State
                    | Info                         |
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+------------------------------+
|  2 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                         |
|  4 | rdsadmin        | localhost          | NULL      | Sleep   |    2 | NULL
                    | NULL                         |
|  5 | rdsadmin        | localhost          | NULL      | Sleep   |    1 | NULL
                    | NULL                         |
| 20 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                         |
| 21 | rdsadmin        | localhost          | NULL      | Sleep   |  261 | NULL
                    | NULL                         |
| 66 | auroramysql5712 | 172.31.21.51:52154 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                         |
| 67 | auroramysql5712 | 172.31.21.51:52158 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                         |
| 68 | auroramysql5712 | 172.31.21.51:52150 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                         |
| 69 | auroramysql5712 | 172.31.21.51:52162 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                         |
| 70 | auroramysql5712 | 172.31.21.51:52160 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                         |
| 71 | auroramysql5712 | 172.31.21.51:52152 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                         |
```

```
| 72 | auroramysql5712 | 172.31.21.51:52156 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 73 | auroramysql5712 | 172.31.21.51:52164 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 74 | auroramysql5712 | 172.31.21.51:52166 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 75 | auroramysql5712 | 172.31.21.51:52168 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 76 | auroramysql5712 | 172.31.21.51:52170 | NULL      | Query   |    0 | starting
                    | show processlist              |
| 88 | auroramysql5712 | 172.31.21.51:52194 | NULL      | Query   |   22 | User sleep
                    | select sleep(10000)           |
| 89 | auroramysql5712 | 172.31.21.51:52196 | NULL      | Query   |    5 | Waiting for
 table metadata lock | truncate table sbtest.sbtest1 |
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+-------------------------------+
18 rows in set (0.00 sec)
```

A continuación, una consulta sobre las tablas `performance_schema` o las vistas de esquema `sys` muestra que la sesión de bloqueo es 76.

```
MySQL [(none)]> select * from sys.schema_table_lock_waits;

+---------------+-------------+-------------------+-------------
+-------------------------------+-------------------+-----------------------
+-------------------------------+--------------------+----------------------------
+-------------------------------+--------------------+-------------
+-------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
| object_schema | object_name | waiting_thread_id | waiting_pid | waiting_account
          | waiting_lock_type | waiting_lock_duration | waiting_query
   | waiting_query_secs | waiting_query_rows_affected | waiting_query_rows_examined |
 blocking_thread_id | blocking_pid | blocking_account             | blocking_lock_type
 | blocking_lock_duration | sql_kill_blocking_query | sql_kill_blocking_connection |
+---------------+-------------+-------------------+-------------
+-------------------------------+-------------------+-----------------------
+-------------------------------+--------------------+----------------------------
+-------------------------------+--------------------+-------------
+-------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
| sbtest        | sbtest1     |               121 |          89 |
 auroramysql5712@192.0.2.0    | EXCLUSIVE         | TRANSACTION           | truncate
 table sbtest.sbtest1 |                 10 |                           0 |
```

```
                0 |                108 |           76 | auroramysql5712@192.0.2.0    |
 SHARED_READ        | TRANSACTION            | KILL QUERY 76           | KILL 76
               |
+---------------+-------------+-------------------+-------------
+------------------------------+--------------------+----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+--------------
+------------------------------+--------------------+----------------------
+-------------------------+-------------------------------+
1 row in set (0.00 sec)
```

Responder a la sesión de bloqueo

Cuando identifique la sesión, puede seguir una de las opciones siguientes:

- Ponerse en contacto con el propietario o el usuario de la aplicación.
- Si la sesión de bloqueo está inactiva, considere la posibilidad de finalizar la sesión de bloqueo. Esta acción podría desencadenar una larga restauración. Para aprender a finalizar una sesión, consulte Finalización de una sesión o una consulta.

Para obtener más información acerca de cómo identificar las transacciones de bloqueo, consulte Using InnoDB Transaction and Locking Information en la documentación de MySQL.

synch/mutex/innodb/aurora_lock_thread_slot_futex

El evento `synch/mutex/innodb/aurora_lock_thread_slot_futex` se produce cuando una sesión ha bloqueado una fila para una actualización y otra sesión intenta actualizar la misma fila. Para obtener información, consulte InnoDB Locking en MySQL Reference.

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 2

Causas probables del aumento del tiempo de espera

Las múltiples instrucciones de lenguaje de manipulación de datos (DML) tienen acceso a la misma fila o filas simultáneamente.

Acciones

Recomendamos diferentes acciones en función de los demás eventos de espera que vea.

Temas

- Buscar y responder a las instrucciones SQL responsables de este evento de espera
- Buscar y responder a la sesión de bloqueo

Buscar y responder a las instrucciones SQL responsables de este evento de espera

Utilice Información sobre rendimiento para identificar las instrucciones SQL responsables de este evento de espera. Consideremos la posibilidad de aplicar las estrategias siguientes:

- Si los bloqueos de fila son un problema persistente, considere la posibilidad de reescribir la aplicación para utilizar un bloqueo optimista.
- Utilice instrucciones de varias filas.
- Distribuya la carga de trabajo en distintos objetos de base de datos. Puede hacerlo mediante una partición.
- Verifique el valor del parámetro `innodb_lock_wait_timeout`. Controla cuánto tiempo esperan las transacciones antes de generar un error de tiempo de espera.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Buscar y responder a la sesión de bloqueo

Determine si la sesión de bloqueo está inactiva o activa. Además, averigüe si la sesión procede de una aplicación o de un usuario activo.

Para identificar la sesión que mantiene el candado, puede ejecutar `SHOW ENGINE INNODB STATUS`. A continuación se muestra un resultado de ejemplo.

```
mysql> SHOW ENGINE INNODB STATUS;

---------------------TRANSACTION 302631452, ACTIVE 2 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 376, 1 row lock(s)
```

```
MySQL thread id 80109, OS thread handle 0x2ae915060700, query id 938819 10.0.4.12
 reinvent updating
UPDATE sbtest1 SET k=k+1 WHERE id=503
------- TRX HAS BEEN WAITING 2 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 148 page no 11 n bits 30 index `PRIMARY` of table
 `sysbench2`.`sbtest1` trx id 302631452 lock_mode X locks rec but not gap waiting
Record lock, heap no 30 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
```

O bien, puede utilizar la siguiente consulta para extraer detalles sobre los bloqueos actuales.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
```

```
         waiting_user: reinvent
         waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Cuando identifique la sesión, puede seguir una de las opciones siguientes:

- Ponerse en contacto con el propietario o el usuario de la aplicación.
- Si la sesión de bloqueo está inactiva, considere la posibilidad de finalizar la sesión de bloqueo. Esta acción podría desencadenar una larga restauración. Para aprender a finalizar una sesión, consulte Finalización de una sesión o una consulta.

Para obtener más información acerca de cómo identificar las transacciones de bloqueo, consulte Using InnoDB Transaction and Locking Information en MySQL Reference Manual.

synch/mutex/innodb/buf_pool_mutex

El evento `synch/mutex/innodb/buf_pool_mutex` se produce cuando un subproceso ha adquirido un bloqueo en el grupo de búferes de InnoDB para acceder a una página de memoria.

Temas

- Versiones del motor relevantes
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor relevantes

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 2

Contexto

El mutex `buf_pool` es un único mutex que protege las estructuras de datos de control del grupo de búferes.

Para obtener más información, consulte Monitoring InnoDB Mutex Waits Using Performance Schema en la documentación de MySQL.

Causas probables del aumento de las esperas

Se trata de un evento de espera específico de la carga de trabajo. Las causas más comunes para que `synch/mutex/innodb/buf_pool_mutex` aparezca entre los eventos de espera principales son las siguientes:

- El tamaño del grupo de búferes no es lo suficientemente grande como para almacenar el conjunto de datos de trabajo.
- La carga de trabajo es más específica para determinadas páginas de una tabla específica de la base de datos, lo que genera contención en el grupo de búferes.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas que provocan los eventos
- Utilizar Información sobre rendimiento
- Crear réplicas de Aurora
- Examinar el tamaño del grupo de búferes
- Monitorear el historial de estado global

Identificar las sesiones y consultas que provocan los eventos

Normalmente, las bases de datos con una carga de moderada a significativa tienen eventos de espera. Los eventos de espera pueden ser aceptables si el rendimiento es óptimo. Si el rendimiento no es óptimo, examine dónde pasa más tiempo la base de datos. Preste atención a los eventos

de espera que contribuyen a la carga más alta y averigüe si puede optimizar la base de datos y la aplicación para reducirlos.

Para ver el gráfico SQL superior en AWS Management Console

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Performance Insights.
3. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. Debajo del cuadro Database load (Carga de base de datos), elija Top SQL (SQL principal).

   El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Utilizar Información sobre rendimiento

Este evento está relacionado con la carga de trabajo. Puede utilizar Información sobre rendimiento para hacer lo siguiente:

- Identificar cuándo comienzan los eventos de espera y verificar si se produce algún cambio en la carga de trabajo en ese momento a partir de los registros de aplicaciones u orígenes relacionados.
- Identificar las instrucciones SQL responsables de este evento de espera. Examine el plan de ejecución de las consultas para asegurarse de que las consultas estén optimizadas y utilicen los índices adecuados.

  Si las principales consultas responsables del evento de espera están relacionadas con el mismo objeto o tabla de base de datos, considere la posibilidad de crear particiones de ese objeto o tabla.

Crear réplicas de Aurora

Puede crear réplicas de Aurora para servir tráfico de solo lectura. También puede utilizar el escalado automático de Aurora para gestionar las sobretensiones en el tráfico de lectura. Asegúrese de ejecutar tareas de solo lectura programadas y copias de seguridad lógicas en las réplicas de Aurora.

Para obtener más información, consulte Amazon Aurora Auto Scaling con réplicas de Aurora.

Examinar el tamaño del grupo de búferes

Consulte la métrica `innodb_buffer_pool_wait_free` para verificar si el tamaño del grupo de búferes es suficiente para la carga de trabajo. Si el valor de esta métrica es alto y aumenta continuamente, indica que el tamaño del grupo de búferes no es suficiente para gestionar la carga de trabajo. Si `innodb_buffer_pool_size` se ha establecido correctamente, el valor de `innodb_buffer_pool_wait_free` debe ser pequeño. Para obtener más información, consulte Innodb_buffer_pool_wait_free en la documentación de MySQL.

Aumente el tamaño del grupo de búferes si la instancia de base de datos tiene suficiente memoria para búferes de sesión y tareas del sistema operativo. Si no lo hace, cambie la instancia de base de datos a una clase de instancia de base de datos de mayor tamaño para obtener memoria adicional que se pueda asignar al grupo de búferes.

> **Note**
>
> Aurora MySQL ajusta automáticamente el valor de `innodb_buffer_pool_instances` en función del `innodb_buffer_pool_size` configurado.

Monitorear el historial de estado global

Al monitorear la velocidad de cambio de las variables de estado, puede detectar problemas de bloqueo o memoria en la instancia de base de datos. Active el historial de estado global (GoSH) si aún no está activado. Para obtener más información sobre GoSH, consulte Administrar el historial de estado global.

También puede crear métricas de Amazon CloudWatch personalizadas para monitorear las variables de estado. Para obtener más información, consulte Publicación de métricas personalizadas.

synch/mutex/innodb/fil_system_mutex

El evento `synch/mutex/innodb/fil_system_mutex` se produce cuando una sesión está a la espera para acceder a la memoria caché de memoria del espacio de tabla.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL, versiones 2 y 3

Contexto

InnoDB utiliza espacios de tabla para administrar el área de almacenamiento de las tablas y archivos de registro. La caché de memoria del espacio de tabla es una estructura de memoria global que mantiene información sobre los espacios de tabla. MySQL utiliza las esperas `synch/mutex/innodb/fil_system_mutex` para controlar el acceso simultáneo a la caché de memoria del espacio de tabla.

El evento `synch/mutex/innodb/fil_system_mutex` indica que actualmente hay más de una operación que necesita recuperar y manipular información en la caché de memoria del espacio de tabla para el mismo espacio de tabla.

Causas probables del aumento de las esperas

Cuando el evento `synch/mutex/innodb/fil_system_mutex` aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, esto suele deberse a la presencia de las condiciones siguientes:

- Aumento de las operaciones simultáneas de lenguaje de manipulación de datos (DML) que actualizan o eliminan datos de la misma tabla.

- El espacio de tabla de esta tabla es muy grande y tiene muchas páginas de datos.
- El factor de relleno de estas páginas de datos es bajo.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas que provocan los eventos
- Reorganizar grandes tablas durante las horas de menor actividad

Identificar las sesiones y consultas que provocan los eventos

Normalmente, las bases de datos con una carga de moderada a significativa tienen eventos de espera. Los eventos de espera pueden ser aceptables si el rendimiento es óptimo. Si el rendimiento no es óptimo, examine dónde pasa más tiempo la base de datos. Preste atención a los eventos de espera que contribuyen a la carga más alta y averigüe si puede optimizar la base de datos y la aplicación para reducirlos.

Para buscar consultas SQL responsables de cargas elevadas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información sobre rendimiento.
3. Elija una instancia de base de datos. Se muestra el panel Información sobre rendimiento para dicha instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

   El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Otra forma de averiguar qué consultas están causando un gran número de esperas `synch/mutex/innodb/fil_system_mutex` consiste en verificar `performance_schema`, como en el ejemplo siguiente.

```
mysql> select * from performance_schema.events_waits_current where EVENT_NAME='wait/
synch/mutex/innodb/fil_system_mutex'\G
*************************** 1. row ***************************
            THREAD_ID: 19
             EVENT_ID: 195057
         END_EVENT_ID: 195057
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:6700
          TIMER_START: 1010146190118400
            TIMER_END: 1010146196524000
           TIMER_WAIT: 6405600
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 2. row ***************************
            THREAD_ID: 23
             EVENT_ID: 5480
         END_EVENT_ID: 5480
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:5906
          TIMER_START: 995269979908800
            TIMER_END: 995269980159200
           TIMER_WAIT: 250400
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
```

```
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 3. row ***************************
            THREAD_ID: 55
             EVENT_ID: 23233794
         END_EVENT_ID: NULL
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:449
          TIMER_START: 1010492125341600
            TIMER_END: 1010494304900000
           TIMER_WAIT: 2179558400
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: 23233786
   NESTING_EVENT_TYPE: WAIT
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

Reorganizar grandes tablas durante las horas de menor actividad

Reorganice las tablas de gran tamaño que identifique como origen de un gran número de eventos de espera `synch/mutex/innodb/fil_system_mutex` durante el periodo de mantenimiento fuera del horario de producción. De este modo, se garantiza que la limpieza de la asignación de los espacios de tabla internos no tenga lugar en momentos en los que el acceso rápido a la tabla es crítico. Para obtener información sobre la reorganización de tablas, consulte OPTIMIZE TABLE Statement en MySQL Reference.

synch/mutex/innodb/trx_sys_mutex

El evento `synch/mutex/innodb/trx_sys_mutex` se produce cuando hay una elevada actividad de la base de datos con un gran número de transacciones.

Temas

- Versiones del motor relevantes
- Contexto

- Causas probables del aumento de las esperas
- Acciones

Versiones del motor relevantes

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL, versiones 2 y 3

Contexto

Internamente, el motor de base de datos InnoDB utiliza el nivel de aislamiento de lectura repetible con instantáneas para proporcionar coherencia de lectura. Esto proporciona una vista puntual de la base de datos en el momento en que se creó la instantánea.

En InnoDB, todos los cambios se aplican a la base de datos tan pronto como llegan, independientemente de si están confirmados. Este enfoque significa que sin el control de simultaneidad multiversión (MVCC), todos los usuarios conectados a la base de datos ven todos los cambios y las filas más recientes. Por lo tanto, InnoDB requiere una forma de realizar un seguimiento de los cambios para comprender qué se debe revertir cuando sea necesario.

Para ello, InnoDB utiliza un sistema de transacciones (`trx_sys`) para realizar un seguimiento de las instantáneas. El sistema de transacciones realiza lo siguiente:

- Realiza un seguimiento del ID de transacción de cada fila en los registros de deshacer.
- Utiliza una estructura interna de InnoDB denominada `ReadView` que ayuda a identificar qué ID de transacción están visibles para una instantánea.

Causas probables del aumento de las esperas

Cualquier operación de base de datos que requiera un manejo coherente y controlado (creación, lectura, actualización y eliminación) de los ID de transacciones genera una llamada desde `trx_sys` al mutex.

Estas llamadas se realizan en tres funciones:

- `trx_sys_mutex_enter`: crea el mutex.
- `trx_sys_mutex_exit`: libera el mutex.

- `trx_sys_mutex_own`: comprueba si el mutex tiene propietario.

La instrumentación de InnoDB Performance Schema realiza un seguimiento de todas las llamadas de mutex `trx_sys`. El seguimiento incluye, entre otras acciones, la administración de `trx_sys` en el inicio o cierre de la base de datos, las operaciones de reversión, las limpiezas de deshacer, el acceso de lectura a filas y las cargas de grupos de búferes. La elevada actividad de la base de datos con un gran número de transacciones da como resultado la aparición frecuente de `synch/mutex/innodb/trx_sys_mutex` entre los eventos de espera principales.

Para obtener más información, consulte Monitoring InnoDB Mutex Waits Using Performance Schema en la documentación de MySQL.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identificar las sesiones y consultas que provocan los eventos
- Examinar otros eventos de espera

Identificar las sesiones y consultas que provocan los eventos

Normalmente, las bases de datos con una carga de moderada a significativa tienen eventos de espera. Los eventos de espera pueden ser aceptables si el rendimiento es óptimo. Si el rendimiento no es óptimo, examine dónde pasa más tiempo la base de datos. Observe los eventos de espera que contribuyen a la carga más alta. Descubra si puede optimizar la base de datos y la aplicación para reducir esos eventos.

Para ver el gráfico SQL principal en la Consola de administración de AWS

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Performance Insights.
3. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. Bajo el cuadro Database load (Carga de base de datos), elija Top SQL (SQL principal).

El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Examinar otros eventos de espera

Examine los demás eventos de espera asociados con el evento de espera `synch/mutex/innodb/trx_sys_mutex`. De este modo, podrá proporcionar más información acerca de la naturaleza de la carga de trabajo. Un gran número de transacciones podría reducir el rendimiento, pero la carga de trabajo también puede hacer que esto sea necesario.

Para obtener más información sobre cómo optimizar las transacciones, consulte Optimizing InnoDB Transaction Management en la documentación de MySQL.

synch/sxlock/innodb/hash_table_locks

El evento `synch/sxlock/innodb/hash_table_locks` se produce cuando se deben leer desde el almacenamiento páginas que no se encuentran en el grupo de búferes.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones:

- Aurora MySQL, versiones 2 y 3

Contexto

El evento `synch/sxlock/innodb/hash_table_locks` indica que una carga de trabajo accede con frecuencia a datos que no se almacenan en el grupo de búferes. Este evento de espera está asociado con adiciones de páginas nuevas y expulsiones de datos antiguos del grupo de búferes. Los datos almacenados en el grupo de búferes antiguos y los datos nuevos deben almacenarse en la caché, de modo que las páginas antiguas se expulsan para permitir el almacenamiento en caché de las nuevas páginas. MySQL utiliza un algoritmo de elementos menos usados recientemente (LRU) para expulsar las páginas del grupo de búferes. La carga de trabajo intenta acceder a los datos que no se han cargado en el grupo de búferes o a los datos que se han expulsado del grupo de búferes.

Este evento de espera se produce cuando la carga de trabajo debe acceder a los datos de los archivos del disco o cuando los bloques se liberan o se agregan a la lista LRU del grupo de búferes. Estas operaciones esperan para obtener un bloqueo excluido compartido (SX-lock). Este bloqueo SX se utiliza para la sincronización a través de la tabla de hash, que es una tabla en memoria diseñada para mejorar el rendimiento del acceso al grupo de búferes.

Para obtener más información, consulte Buffer Pool en la documentación de MySQL.

Causas probables del aumento de las esperas

Cuando el evento de espera `synch/sxlock/innodb/hash_table_locks` aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

Un grupo de búferes de tamaño insuficiente

: El tamaño del grupo de búferes es demasiado pequeño para mantener en memoria todas las páginas a las que se accede con frecuencia.

Carga de trabajo pesada

: La carga de trabajo está provocando expulsiones frecuentes y recargas de páginas de datos en la caché del búfer.

Errores al leer las páginas

: Hay errores al leer las páginas del grupo de búferes, lo que podría indicar daños en los datos.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Aumentar el tamaño del grupo de búferes
- Mejorar los patrones de acceso a datos
- Reducir o evitar análisis de tablas completas
- Verificar los registros de errores para detectar daños en la página

Aumentar el tamaño del grupo de búferes

Asegúrese de que el grupo de búferes tenga el tamaño adecuado para la carga de trabajo. Para ello, puede verificar la tasa de aciertos de caché del grupo de búferes. Normalmente, si el valor cae por debajo del 95 por ciento, debe considerar la posibilidad de aumentar el tamaño del grupo de búferes. Un grupo de búferes más grande puede mantener en memoria las páginas a las que se accede con frecuencia durante más tiempo. Para aumentar el tamaño del grupo de búferes, modifique el valor del parámetro `innodb_buffer_pool_size`. El valor predeterminado de este parámetro se basa en el tamaño de clase de la instancia de base de datos. Para obtener más información, consulte Best practices for Amazon Aurora MySQL database configuration.

Mejorar los patrones de acceso a datos

Verifique las consultas afectadas por esta espera y sus planes de ejecución. Considere la posibilidad de mejorar los patrones de acceso a datos. Por ejemplo, si está utilizando mysqli_result::fetch_array, puede probar a aumentar el tamaño de búsqueda de matriz.

Puede utilizar Información sobre rendimiento para mostrar consultas y sesiones que podrían estar provocando el evento de espera `synch/sxlock/innodb/hash_table_locks`.

Para buscar consultas SQL responsables de cargas elevadas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información sobre rendimiento.
3. Elija una instancia de base de datos. Se muestra el panel de Información sobre rendimiento para esa instancia de base de datos.
4. En el cuadro Database load (Carga de base de datos), elija Slice by wait (Corte por espera).
5. En la parte inferior de la página, elija Top SQL (SQL principal).

El gráfico enumera las consultas SQL responsables de la carga. Las que están en la parte superior de la lista son las más importantes. Para resolver un cuello de botella, céntrese en estas instrucciones.

Para obtener información general útil sobre la solución de problemas mediante Información sobre rendimiento, consulte la entrada de blog de AWS Database Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Reducir o evitar análisis de tablas completas

Monitoree su carga de trabajo para ver si está ejecutando análisis de tablas completas y, de ser así, reducirlos o evitarlos. Por ejemplo, puede monitorear variables de estado como, por ejemplo, `Handler_read_rnd_next`. Para obtener más información, consulte Server Status Variables en la documentación de MySQL.

Verificar los registros de errores para detectar daños en la página

Puede verificar el archivo mysql-error.log en busca de mensajes relacionados con daños que se detectaron cerca del momento en que se produjo el problema. Los mensajes con los que puede trabajar para resolver el problema se encuentran en el registro de errores. Es posible que tenga que volver a crear objetos que se han informado como dañados.

synch/mutex/innodb/temp_pool_manager_mutex

El evento de espera `synch/mutex/innodb/temp_pool_manager_mutex` se produce cuando una sesión está esperando para adquirir un mutex para administrar el grupo de espacios de tablas temporales de sesión.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de evento de espera es compatible con las siguientes versiones del motor:

- Aurora MySQL versión 3

Contexto

La versión 3.x y posterior de Aurora MySQL utiliza `temp_pool_manager_mutex` para controlar varias sesiones que acceden al grupo de espacios de tablas temporales al mismo tiempo. Aurora MySQL administra el almacenamiento a través de un volumen de clúster de Aurora para los datos persistentes y el almacenamiento local para los archivos temporales. Se necesita un espacio de tabla temporal cuando una sesión crea una tabla temporal en el volumen del clúster DE Aurora.

Cuando una sesión solicita por primera vez un espacio de tabla temporal, MySQL asigna los espacios de tablas temporales de sesión del grupo compartido. Una sesión puede contener hasta 2 espacios de tablas temporales a la vez para los siguientes tipos de tablas:

- Tablas temporales creadas por el usuario
- Tablas temporales internas generadas por el optimizador

El motor de `TempTable` predeterminado utiliza el siguiente mecanismo de desbordamiento para gestionar las tablas temporales:

- Almacena las tablas en RAM hasta el límite `temptable_max_ram`.
- Se traslada a los archivos asignados a memoria del almacenamiento local cuando la RAM está llena.
- Usa el volumen del clúster compartido cuando los archivos asignados a memoria alcanzan su límite `temptable_max_mmap`.

Cuando las tablas temporales superan los límites de RAM y almacenamiento local, MySQL las administra mediante el espacio de tablas del disco.

Cuando una sesión requiere una tabla temporal en disco, MySQL:

- Busca los espacios de tablas `INACTIVE` disponibles en el grupo para reutilizarlos.
- Crea 10 espacios de tabla nuevos si no hay espacios `INACTIVE`.

Cuando una sesión se desconecta, MySQL:

- Trunca los espacios de tablas temporales de la sesión.

- Los marca como INACTIVOS en el grupo para su reutilización.
- Mantiene el tamaño actual del grupo hasta que se reinicie el servidor.
- Vuelve al tamaño de grupo predeterminado (10 espacios de tabla) tras el reinicio.

Causas probables del aumento de las esperas

Situaciones comunes que provocan este evento de espera:

- Sesiones simultáneas que crean tablas temporales internas en el volumen del clúster.
- Sesiones simultáneas que crean tablas temporales de usuario en el volumen del clúster.
- Terminación repentina de las sesiones mediante espacios de tabla activos.
- Expansión del grupo de espacios de tabla durante cargas de trabajo de escritura intensivas.
- Consultas simultáneas mediante el acceso a `INFORMATION_SCHEMA.`

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Supervisión y optimización del uso de las tablas temporales
- Revisión de las consultas mediante INFORMATION_SCHEMA
- Aumento del parámetro innodb_sync_array_size
- Implementar la agrupación de conexiones

Supervisión y optimización del uso de las tablas temporales

Para supervisar y optimizar el uso temporal de las tablas, utilice uno de estos métodos:

- Consulte el contador de `Created_tmp_disk_tables` en la información de rendimiento para realizar un seguimiento de la creación de tablas temporales en disco en todo el clúster de Aurora.
- Ejecute este comando en la base de datos para supervisar directamente la creación de tablas temporales: `mysql> show status like '%created_tmp_disk%'`.

> **Note**
>
> El comportamiento de las tablas temporales difiere entre los nodos de lectura y escritura de Aurora MySQL. Para obtener más información, consulte Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3.

Tras identificar las consultas mediante la creación de tablas temporales, siga estos pasos de optimización:

- Use `EXPLAIN` para examinar los planes de ejecución de consultas e identificar dónde y por qué se crean las tablas temporales.
- Modifique las consultas para reducir el uso temporal de las tablas siempre que sea posible.

Si la optimización de consultas por sí sola no resuelve los problemas de rendimiento, considere la posibilidad de ajustar estos parámetros de configuración:

- `temptable_max_ram`: controla el uso máximo de RAM para las tablas temporales.
- `temptable_max_mmap`: establece el límite de almacenamiento de archivos asignados en memoria.
- `tmp_table_size`: se aplica cuando `aurora_tmptable_enable_per_table_limit` está habilitado (desactivado de forma predeterminada).

> **Important**
>
> Tenga en cuenta que determinadas condiciones de consulta siempre requerirán tablas temporales en disco, independientemente de los ajustes de configuración. Para obtener más información sobre el motor de almacenamiento de `TempTable`, consulte Use the TempTable storage engine on Amazon RDS for MySQL and Amazon Aurora MySQL.

Revisión de las consultas mediante INFORMATION_SCHEMA

Al consultar tablas de `INFORMATION_SCHEMA`, MySQL crea tablas temporales de InnoDB en el volumen del clúster. Cada sesión necesita un espacio de tablas temporal para estas tablas, lo que puede provocar problemas de rendimiento si el acceso simultáneo es elevado.

Para mejorar el desempeño:

- Use `PERFORMANCE_SCHEMA` en lugar de `INFORMATION_SCHEMA` cuando sea posible.
- Si debe utilizar `INFORMATION_SCHEMA`, reduzca la frecuencia con la que ejecuta estas consultas.

Aumento del parámetro innodb_sync_array_size

El parámetro `innodb_sync_array_size` controla el tamaño de la matriz de espera mutex/lock en MySQL. El valor predeterminado de 1 funciona para cargas de trabajo generales, pero aumentarlo puede reducir la contención de subprocesos en situaciones de alta concurrencia.

Cuando la carga de trabajo muestra un número creciente de subprocesos en espera:

- Supervise la cantidad de subprocesos de espera de la carga de trabajo.
- Establezca un `innodb_sync_array_size` igual o superior al de la vCPU de la instancia para dividir la estructura de coordinación de subprocesos y reducir la contención.

> **Note**
>
> Para determinar la cantidad de vCPU disponibles en la instancia de RDS, consulte las especificaciones de las vCPU en los tipos de instancias de Amazon RDS.

Implementar la agrupación de conexiones

MySQL asigna un espacio de tabla dedicado a cada sesión que crea una tabla temporal. Este espacio de tabla permanece activo hasta que finalice la conexión a la base de datos. Para administrar los recursos de manera más eficiente:

- Implemente la agrupación de conexiones para limitar el número de espacios de tabla temporales activos.
- Reutilice las conexiones existentes en lugar de crear nuevas para cada operación.

Ajustar Aurora MySQL con estados de subprocesos

En la tabla siguiente se resumen los estados de subprocesos generales más comunes para Aurora MySQL.

Estado general del subproceso	Descripción
???	Este estado de subproceso indica que un subproceso está procesando una instrucci ón `SELECT` que requiere el uso de una tabla temporal interna para ordenar los datos.
???	Este estado de subproceso indica que un subproceso está leyendo y filtrando filas de una consulta para determinar el conjunto de resultados correcto.

crear índice de ordenación

El estado de subproceso `creating sort index` indica que un subproceso está procesando una instrucción `SELECT` que requiere el uso de una tabla temporal interna para ordenar los datos.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de estado de subproceso es compatible con las siguientes versiones:

- Aurora MySQL versión 2, hasta la versión 2.09.2

Context

El estado `creating sort index` aparece cuando una consulta con una cláusula `ORDER BY` o `GROUP BY` no puede utilizar un índice existente para realizar la operación. En este caso, MySQL necesita realizar una operación `filesort` más costosa. Esta operación se realiza normalmente en la memoria si el conjunto de resultados no es demasiado grande. De lo contrario, requiere crear un archivo en disco.

Causas probables del aumento de las esperas

La apariencia de `creating sort index` no indica un problema. Si el rendimiento es deficiente y ve instancias frecuentes de `creating sort index`, la causa más probable es la lentitud de las consultas con los operadores `ORDER BY` o `GROUP BY`.

Acciones

La directriz general consiste en buscar consultas con las cláusulas `ORDER BY` o `GROUP BY` asociadas a los aumentos del estado `creating sort index`. A continuación, compruebe si, al agregar un índice o aumentar el tamaño del búfer de ordenación, se resuelve el problema.

Temas

- Activar Performance Schema si no está activado
- Identificar las consultas problemáticas
- Examinar los planes de explicación para el uso de filesort
- Aumentar el tamaño del búfer de ordenación

Activar Performance Schema si no está activado

Información sobre rendimiento informa de los estados de subprocesos solo si los instrumentos de Performance Schema no están activados. Cuando los instrumentos de Performance Schema están activados, Información sobre rendimiento informa de los eventos de espera en su lugar. Los instrumentos de Performance Schema proporcionan información adicional y mejores herramientas para investigar posibles problemas de rendimiento. Por lo tanto, se recomienda activar Performance Schema. Para obtener más información, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

Identificar las consultas problemáticas

Para identificar las consultas actuales que están provocando aumentos en el estado `creating sort index`, ejecute `show processlist` y compruebe si alguna de las consultas tiene `ORDER BY` o `GROUP BY`. Opcionalmente, ejecute `explain for connection N`, donde `N` es el ID de lista de procesos de la consulta con `filesort`.

Para identificar las consultas anteriores que están causando estos aumentos, active el registro de consultas lentas y busque las consultas con `ORDER BY`. Ejecute `EXPLAIN` en las consultas lentas y utilice filesort. Para obtener más información, consulte Examinar los planes de explicación para el uso de filesort.

Examinar los planes de explicación para el uso de filesort

Identifique las instrucciones con las cláusulas `ORDER BY` o `GROUP BY` que dan lugar al estado `creating sort index`.

El siguiente ejemplo muestra cómo ejecutar `explain` en una consulta. La columna `Extra` muestra que esta consulta utiliza `filesort`.

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2064548
     filtered: 100.00
        Extra: Using filesort
1 row in set, 1 warning (0.01 sec)
```

El siguiente ejemplo muestra el resultado de ejecutar `EXPLAIN` en la misma consulta después de crear un índice en la columna `c1`.

```
mysql> alter table mytable add index (c1);
```

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: index
possible_keys: NULL
          key: c1
      key_len: 1023
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: Using index
1 row in set, 1 warning (0.01 sec)
```

Para obtener información sobre el uso de índices para la optimización del orden de clasificación, consulte ORDER BY Optimization en la documentación de MySQL.

Aumentar el tamaño del búfer de ordenación

Para ver si una consulta específica ha necesitado un proceso `filesort` que creó un archivo en disco, verifique el valor de la variable `sort_merge_passes` después de ejecutar la consulta. A continuación se muestra un ejemplo.

```
mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)

--- run query
mysql> select * from mytable order by u limit 10;
--- run status again:

mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
```

```
+-------------------+-------+
1 row in set (0.01 sec)
```

Si el valor de `sort_merge_passes` es alto, considere la posibilidad de aumentar el tamaño del búfer de ordenación. Aplique el aumento en el nivel de la sesión, ya que aumentarlo globalmente puede incrementar significativamente la cantidad de RAM que utiliza MySQL. En el ejemplo siguiente se muestra cómo cambiar el tamaño del búfer de ordenación antes de ejecutar una consulta.

```
mysql> set session sort_buffer_size=10*1024*1024;
Query OK, 0 rows affected (0.00 sec)
-- run query
```

envío de datos

El estado del subproceso `sending data` indica que un subproceso está leyendo y filtrando filas de una consulta para determinar el conjunto de resultados correcto. El nombre puede dar lugar a confusión porque implica que el estado está transfiriendo datos en lugar de recopilar y preparar datos para enviarlos más adelante.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de estado de subproceso es compatible con las siguientes versiones:

- Aurora MySQL versión 2, hasta la versión 2.09.2

Context

Muchos estados de subprocesos son de corta duración. Las operaciones que se producen durante `sending data` tienden a realizar un gran número de lecturas de disco o caché. Por consiguiente, `sending data` suele ser el estado que permanece más tiempo en ejecución a lo largo de la vida útil de una consulta determinada. Este estado aparece cuando Aurora MySQL hace lo siguiente:

- Lectura y procesamiento de filas de una instrucción `SELECT`
- Realización de un gran número de lecturas desde el disco o la memoria
- Realización de una lectura completa de todos los datos de una consulta específica
- Lectura de datos desde una tabla, un índice o el trabajo de un procedimiento almacenado
- Clasificación, agrupación u ordenación de datos

Una vez que el estado `sending data` finaliza la preparación de los datos, el estado del subproceso `writing to net` indica la devolución de datos al cliente. Normalmente, `writing to net` solo se captura cuando el conjunto de resultados es muy grande o cuando la elevada latencia de red ralentiza la transferencia.

Causas probables del aumento de las esperas

La apariencia de `sending data` no indica un problema. Si el rendimiento es deficiente y ve instancias frecuentes de `sending data`, las causas más probables son las siguientes.

Temas

- Consulta ineficiente
- Configuración de servidor poco óptima

Consulta ineficiente

En la mayoría de los casos, la causa de este estado suele ser una consulta que no utiliza un índice adecuado para buscar el conjunto de resultados de una consulta específica. Por ejemplo, piense en una consulta que lee una tabla de 10 millones de registros para todos los pedidos realizados en California, donde la columna de estado no está indexada o está mal indexada. En este caso, el índice puede existir, pero el optimizador lo ignora debido a su baja cardinalidad.

Configuración de servidor poco óptima

Si aparecen varias consultas en el estado `sending data`, el servidor de base de datos podría estar mal configurado. Más específicamente, el servidor podría tener los problemas siguientes:

- El servidor de base de datos no tiene suficiente capacidad informática: E/S de disco, tipo y velocidad de disco, CPU o número de CPU.
- El servidor no tiene recursos asignados, como, por ejemplo, el grupo de búferes InnoDB para tablas InnoDB o el búfer de claves de las tablas MyIsam.

- La configuración de memoria por subproceso, como, por ejemplo, `sort_buffer`, `read_buffer`, y `join_buffer`, consumen más RAM de la necesaria, lo que provoca que el servidor físico no tenga recursos de memoria.

Acciones

La consigna general radica en buscar consultas que devuelvan un gran número de filas verificando Performance Schema. Si las consultas de registro que no utilizan índices están activadas, también puede examinar los resultados de los registros lentos.

Temas

- Activar Performance Schema si no está activado
- Examinar la configuración de memoria
- Examinar los planes de explicación para el uso de índice
- Verificar el volumen de datos devueltos
- Verificar problemas de concurrencia
- Verificar la estructura de las consultas

Activar Performance Schema si no está activado

Información sobre rendimiento informa de los estados de subprocesos solo si los instrumentos de Performance Schema no están activados. Cuando los instrumentos de Performance Schema están activados, Información sobre rendimiento informa de los eventos de espera en su lugar. Los instrumentos de Performance Schema proporcionan información adicional y mejores herramientas para investigar posibles problemas de rendimiento. Por lo tanto, se recomienda activar Performance Schema. Para obtener más información, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

Examinar la configuración de memoria

Examine la configuración de memoria de los grupos de búferes principales. Asegúrese de que estos grupos tengan el tamaño adecuado para la carga de trabajo. Si la base de datos utiliza varias instancias de grupos de búferes, asegúrese de que no estén divididas en muchos grupos de búferes pequeños. Los subprocesos solo pueden utilizar un grupo de búferes a la vez.

Asegúrese de que los ajustes de memoria utilizados para cada subproceso que se indican a continuación tengan el tamaño correcto:

- read_buffer
- read_rnd_buffer
- sort_buffer
- join_buffer
- binlog_cache

A menos que tenga un motivo específico para modificar la configuración, utilice los valores predeterminados.

Examinar los planes de explicación para el uso de índice

Para consultas en el estado del subproceso `sending data`, examine el plan para determinar si se utilizan los índices adecuados. Si una consulta no utiliza un índice útil, considere la posibilidad de agregar sugerencias como `USE INDEX` o `FORCE INDEX`. Las sugerencias pueden aumentar o disminuir considerablemente el tiempo de ejecución de una consulta, por lo que tenga cuidado antes de agregarlas.

Verificar el volumen de datos devueltos

Verifique las tablas que se están consultando y la cantidad de datos que contienen. ¿Se puede archivar alguno de estos datos? En muchos casos, la causa del tiempo de ejecución de consultas deficiente no es el plan de consultas, sino el volumen de datos que se debe procesar. Muchos desarrolladores son muy eficientes al agregar datos a una base de datos, pero rara vez consideran el ciclo de vida de los conjuntos de datos en las fases de diseño y desarrollo.

Busque consultas que tengan un buen rendimiento en bases de datos de bajo volumen pero que funcionen mal en el sistema actual. A veces, los desarrolladores que diseñan consultas específicas podrían no darse cuenta de que estas consultas están devolviendo 350 000 filas. Es posible que los desarrolladores hayan desarrollado las consultas en un entorno de menor volumen con conjuntos de datos más pequeños que los que tienen los entornos de producción.

Verificar problemas de concurrencia

Verifique si se están ejecutando varias consultas del mismo tipo al mismo tiempo. Algunas formas de consultas se ejecutan de forma eficiente cuando se ejecutan solas. Sin embargo, si se ejecutan formas similares de consulta de manera conjunta o en gran volumen, pueden provocar problemas de concurrencia. A menudo, estos problemas se producen cuando la base de datos utiliza tablas temporales para generar resultados. Un nivel de aislamiento de transacciones restrictivo también puede provocar problemas de concurrencia.

Si las tablas se leen y escriben simultáneamente, la base de datos podría estar utilizando bloqueos. Para ayudar a identificar periodos de bajo rendimiento, examine el uso de bases de datos mediante procesos por lotes a gran escala. Para ver los bloqueos y reversiones recientes, examine el resultado del comando `SHOW ENGINE INNODB STATUS`.

Verificar la estructura de las consultas

Verifique si las consultas capturadas de estos estados utilizan subconsultas. Este tipo de consulta a menudo genera un rendimiento deficiente porque la base de datos compila los resultados internamente y luego los sustituye de nuevo en la consulta para procesar datos. Este proceso es un paso adicional para la base de datos. En muchos casos, este paso puede provocar un rendimiento deficiente en condiciones de cargas altamente concurrentes.

Verifique también si sus consultas utilizan un gran número de cláusulas `ORDER BY` y `GROUP BY`. En estas operaciones, a menudo la base de datos debe formar primero todo el conjunto de datos en la memoria. A continuación, debe pedirlo o agruparlo de manera específica antes de devolverlo al cliente.

Ajuste de Aurora MySQL con información proactiva de Amazon DevOps Guru

La información proactiva de DevOps Guru detecta condiciones problemáticas conocidas en sus clústeres de bases de datos de Aurora MySQL antes de que se produzcan. DevOps Guru puede hacer lo siguiente:

- Evitar muchos problemas comunes en las bases de datos cotejando la configuración de la base de datos con la configuración habitual recomendada.
- Alertar sobre problemas críticos en su flota que, si no se comprueban, pueden provocar problemas mayores en el futuro.
- Avisarle de los problemas que acaban de descubrirse.

Cada información proactiva contiene un análisis de la causa del problema y recomendaciones para las acciones correctivas.

Temas

- La longitud de la lista de historial de InnoDB ha aumentado de forma significativa
- La base de datos está creando tablas temporales en el disco

La longitud de la lista de historial de InnoDB ha aumentado de forma significativa

A partir de *fecha*, la lista de su historial de cambios en las filas aumentó de forma significativa, hasta *longitud* en *db-instance*. Este aumento afecta al rendimiento de cierre de consultas y bases de datos.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables de este problema
- Acciones
- Métricas relevantes

Versiones del motor admitidas

Esta información es compatible con todas las versiones de Aurora MySQL.

Contexto

El sistema de transacciones InnoDB mantiene el control de simultaneidad multiversión (MVCC). Cuando se modifica una fila, la versión previa a la modificación de los datos que se están modificando se almacena como registro de deshacer en un registro de deshacer. Cada registro de deshacer tiene una referencia a su registro de rehacer anterior, lo que da lugar a una lista enlazada.

La lista del historial de InnoDB es una lista global de los registros de deshacer de las transacciones confirmadas. MySQL usa la lista del historial para purgar registros y páginas de registro cuando las transacciones ya no necesitan el historial. La longitud de la lista del historial es el número total de registros de deshacer que contienen modificaciones en la lista del historial. Cada registro contiene una o varias modificaciones. Si la longitud de la lista del historial de InnoDB aumenta demasiado, eso indica que hay un gran número de versiones de filas antiguas, y las consultas y los cierres de bases de datos se ralentizan.

Causas probables de este problema

Las causas típicas de que la lista del historial sea larga son, entre otras, las siguientes:

- Transacciones de larga duración, ya sean de lectura o escritura
- Una carga de escritura pesada

Acciones

Recomendamos diferentes acciones en función de las causas.

Temas

- No inicie ninguna operación que implique el cierre de la base de datos hasta que la lista del historial de InnoDB se reduzca.
- Identifique y finalice las transacciones de larga duración
- Utilice Información sobre rendimiento para verificar los principales hosts y usuarios.

No inicie ninguna operación que implique el cierre de la base de datos hasta que la lista del historial de InnoDB se reduzca.

Si la lista del historial de InnoDB es muy larga, los cierres de la base de datos se ralentizan, por lo que debe reducir el tamaño de la lista antes de iniciar operaciones que impliquen el cierre de la base de datos. Estas operaciones incluyen las actualizaciones de la versión principal de la base de datos.

Identifique y finalice las transacciones de larga duración

Para encontrar transacciones de larga duración, consulte `information_schema.innodb_trx`.

> **Note**
>
> Asegúrese también de buscar transacciones de larga duración en las réplicas de lectura.

Identificación y facilitación de transacciones de larga duración

1. En su cliente SQL, ejecute la siguiente consulta:

```
SELECT a.trx_id,
      a.trx_state,
      a.trx_started,
      TIMESTAMPDIFF(SECOND,a.trx_started, now()) as "Seconds Transaction Has Been
 Open",
      a.trx_rows_modified,
      b.USER,
      b.host,
      b.db,
      b.command,
```

```
      b.time,
      b.state
FROM  information_schema.innodb_trx a,
      information_schema.processlist b
WHERE a.trx_mysql_thread_id=b.id
  AND TIMESTAMPDIFF(SECOND,a.trx_started, now()) > 10
ORDER BY trx_started
```

2. Finalice cada transacción de larga duración con el procedimiento mysql.rds_kill almacenado.

Utilice Información sobre rendimiento para verificar los principales hosts y usuarios.

Optimice las transacciones para que se confirmen inmediatamente un gran número de filas modificadas.

Métricas relevantes

Las siguientes métricas están relacionadas con esta información:

- `trx_rseg_history_len`: esta métrica del contador se puede consultar en la información de rendimiento, así como en la tabla `INFORMATION_SCHEMA.INNODB_METRICS`. Para obtener más información, consulte la tabla de métricas de InnoDB en la documentación de MySQL.
- `RollbackSegmentHistoryListLength`: esta métrica de Amazon CloudWatch mide los registros undo que registran transacciones confirmadas con registros marcados para su eliminación. Estos registros están programados para ser procesados por la operación de depuración de InnoDB. La métrica `trx_rseg_history_len` tiene el mismo valor que `RollbackSegmentHistoryListLength`.
- `PurgeBoundary`: el número de transacción hasta el que se permite la depuración de InnoDB. Si esta métrica de CloudWatch no aumenta durante periodos de tiempo prolongados, es una buena indicación de que las transacciones de larga duración bloquean la depuración de InnoDB. Para investigar, compruebe las transacciones activas en el clúster de base de datos de Aurora MySQL. Esta métrica solo está disponible para Aurora MySQL versión 2.11 y posteriores, además de la versión 3.08 y posteriores.
- `PurgeFinishedPoint`: el número de transacción hasta el que se realiza la depuración de InnoDB. Esta métrica de CloudWatch puede ser de utilidad para evaluar la rapidez con la que avanza la depuración de InnoDB. Esta métrica solo está disponible para Aurora MySQL versión 2.11 y posteriores, además de la versión 3.08 y posteriores.
- `TransactionAgeMaximum`: es la antigüedad de la transacción en ejecución más antigua. Esta métrica de CloudWatch está disponible solo para la versión 3.08 y posteriores de Aurora MySQL.

- `TruncateFinishedPoint`: el número de transacción hasta el que se realiza la operación de deshacer el truncado. Esta métrica de CloudWatch solo está disponible para Aurora MySQL versión 2.11 y posteriores, además de la versión 3.08 y posteriores.

Para obtener más información sobre las métricas de CloudWatch, consulte Métricas de nivel de instancia para Amazon Aurora.

La base de datos está creando tablas temporales en el disco

El uso reciente de la tabla temporal en el disco ha aumentado de forma significativa hasta el *`porcentaje`*. La base de datos está creando alrededor de *`número`* tablas temporales por segundo. Esto podría afectar al rendimiento y aumentar las operaciones de disco en *`db-instance`*.

Temas

- Versiones del motor admitidas
- Context
- Causas probables de este problema
- Acciones
- Métricas relevantes

Versiones del motor admitidas

Esta información es compatible con todas las versiones de Aurora MySQL.

Context

A veces es necesario que el servidor MySQL cree una tabla temporal interna mientras procesa una consulta. Aurora MySQL puede contener una tabla temporal interna en la memoria, donde puede procesarla TempTable o el motor de almacenamiento MEMORY, o bien InnoDB puede almacenarla en disco. Para obtener más información, consulte Internal Temporary Table Use in MySQL (Uso de tablas temporales internas en MySQL) en MySQL Reference Manual (Manual de referencia de MySQL).

Causas probables de este problema

El aumento de las tablas temporales en disco indica el uso de consultas complejas. Si la memoria configurada no es suficiente para almacenar tablas temporales en la memoria, Aurora MySQL crea las tablas en el disco. Esto puede afectar al rendimiento y aumentar las operaciones en disco.

Acciones

Recomendamos diferentes acciones en función de las causas.

- En Aurora MySQL versión 3, le recomendamos que utilice el motor de almacenamiento TempTable.
- Optimice sus consultas para devolver menos datos. Para ello, seleccione solo las columnas necesarias.

  Si activa el esquema de rendimiento con todos los instrumentos `statement` habilitados y cronometrados, puede realizar consultas a `SYS.statements_with_temp_tables` para recuperar la lista de consultas que utilizan tablas temporales. Para obtener más información, consulte Prerequisites for Using the sys Schema (Requisitos previos para utilizar el esquema sys) en la documentación de MySQL.
- Considere la posibilidad de indexar las columnas que participan en las operaciones de clasificación y agrupación.
- Reescriba sus consultas para evitar columnas `BLOB` y `TEXT`. Estas columnas siempre usan el disco.
- Ajuste los siguientes parámetros de la base de datos: `tmp_table_size` y `max_heap_table_size`.

  El valor predeterminado para este parámetro es 16 MiB. Cuando se utiliza el motor de almacenamiento MEMORY para tablas temporales en memoria, su tamaño máximo se define mediante el valor `tmp_table_size` o `max_heap_table_size`, el que sea menor. Cuando se alcanza este tamaño máximo, MySQL convierte automáticamente la tabla temporal interna en memoria en una tabla temporal interna en disco de InnoDB. Para obtener más información, consulte Use the TempTable storage engine on Amazon RDS for MySQL and Amazon Aurora MySQL (Utilice el motor de almacenamiento TempTable en Amazon RDS para MySQL y Amazon Aurora MySQL).

  > **Note**
  >
  > Al crear tablas MEMORY de forma explícita con CREATE TABLE, solo la variable `max_heap_table_size` determina el tamaño al que puede crecer una tabla. Tampoco hay ninguna conversión a un formato en disco.

Métricas relevantes

Las siguientes métricas de Información sobre el rendimiento están relacionadas con esta información:

- Created_tmp_disk_tables
- Created_tmp_tables

Para obtener más información, consulte Created_tmp_disk_tables en la documentación de MySQL.

Consulta paralela para Amazon Aurora MySQL

En este tema, se describe la optimización del rendimiento de las consultas en paralelo de la edición compatible con Amazon Aurora MySQL. Esta característica usa una ruta de procesamiento especial para determinadas consultas de uso intensivo de datos, que aprovecha la arquitectura de almacenamiento compartido de Aurora. Consultas en paralelo funciona mejor con los clústeres de base de datos Aurora MySQL que tienen tablas con millones de filas y consultas analíticas que tardan minutos u horas en completarse.

Temas

- Información general de consultas paralelas de Aurora MySQL
- Creación de un clúster de base de datos de consultas paralelas en Aurora MySQL
- Activación y desactivación de las consultas paralelas en Aurora MySQL
- Optimización de consultas paralelas en Aurora MySQL
- Cómo comprobar qué instrucciones usan consultas paralelas para Aurora MySQL
- Monitorización de consultas paralelas para Aurora MySQL
- Constructos de SQL para consultas paralelas en Aurora MySQL

Información general de consultas paralelas de Aurora MySQL

Consultas en paralelo de Aurora MySQL es una optimización que paraleliza algunas de los cálculos y E/S del procesamiento de consultas con un uso intensivo de datos. El trabajo que se paraleliza incluye la recuperación de filas del almacenamiento, la extracción de valores de columna y la determinación de qué filas coinciden con las condiciones de la cláusula `WHERE` y de las cláusulas JOIN. Este trabajo con uso intensivo de los datos se delega (en términos de optimización de base de datos, se baja de posición) a varios nodos de la capa de almacenamiento distribuido de Aurora. Sin una consulta paralela, cada consulta transfiere todos los datos analizados a un solo nodo del clúster de Aurora MySQL (el nodo principal) y realiza ahí todos los procesamientos de consultas.

> **Tip**
>
> El motor de base de datos de PostgreSQL también tiene una característica denominada “consulta paralela”. Esa característica no está relacionada con la consulta paralela de Aurora.

Cuando está activada una característica de consulta en paralelo, el motor de Aurora MySQL determina automáticamente cuándo las consultas pueden aprovecharla, sin requerir cambios de SQL como sugerencias o atributos de tabla. En las secciones siguientes, puede encontrar una explicación cuándo se aplica una consulta en paralelo a una consulta. También podrá encontrar cómo asegurarse de que las consultas en paralelo se aplican cuando proporcionan el máximo beneficio.

> **Note**
>
> La optimización de consultas en paralelo proporciona el mayor beneficio para consultas de larga duración que tardan minutos u horas en completarse. Aurora MySQL generalmente no realiza la optimización de consultas en paralelo para consultas de bajo costo. Tampoco suele realizar la optimización de consultas paralelas si otra técnica de optimización tiene más sentido, como el almacenamiento en caché de consultas, el almacenamiento en caché de grupo de búferes o las búsquedas de índice. Consulte si observa que las consultas paralelas no se usan cuando lo esper Cómo comprobar qué instrucciones usan consultas paralelas para Aurora MySQL.

Temas

- Ventajas
- Arquitectura
- Requisitos previos
- Limitaciones
- Costos de E/S con consulta paralela

Ventajas

Con la consulta paralela, puede ejecutar consultas analíticas de uso intensivo de datos en tablas de Aurora MySQL. En muchos casos, puede obtener una mejora del rendimiento de orden de magnitud sobre la división tradicional del trabajo para el procesamiento de consultas.

Entre los beneficios de usar consultas en paralelo se incluyen los siguientes:

- Rendimiento de E/S mejorado, debido a la paralelización de solicitudes de lectura física en múltiples nodos de almacenamiento.

- Reduce el tráfico de red. Aurora no transmite páginas de datos enteras de nodos de almacenamiento al nodo principal y, después, filtra las filas y columnas innecesarias. En vez de eso, Aurora transmite tuplas compactas que contienen solo los valores de columna necesarios para el conjunto de resultados.
- Uso reducido de CPU en el nodo director debido al procesamiento de la función de insertar, filtrado de filas y proyección de columnas para la cláusula `WHERE`.
- Presión de memoria reducida en el grupo de búfer. Las páginas procesadas por la consulta paralela no se agregan al grupo de búferes. Este enfoque reduce la posibilidad de que un análisis intensivo de datos expulse los datos utilizados con frecuencia del grupo de búferes.
- Reducción potencial de la duplicación de datos en la canalización de extracción, transformación y carga (ETL), haciendo que sea práctico realizar consultas analíticas de ejecución prolongada en los datos existentes.

Arquitectura

La característica de consulta en paralelo utiliza los principios de arquitectura principales de Aurora MySQL: desacoplar el motor de base de datos del subsistema de almacenamiento y reducir el tráfico de red mediante la optimización de los protocolos de comunicación. Aurora MySQL utiliza estas técnicas para acelerar las operaciones de escritura intensiva, como el procesamiento de registros de rehacer. Las consultas en paralelo aplican los mismos principios a las operaciones de lectura.

> **Note**
>
> La arquitectura de las consultas en paralelo de Aurora MySQL es diferente de la de las características de nombre similar de otros sistemas de base de datos. La consulta en paralelo de Aurora MySQL no implica multiprocesamiento simétrico (SMP), así que no depende de la capacidad de la CPU del servidor de la base de datos. El procesamiento paralelo tiene lugar en la capa de almacenamiento, independiente del servidor de Aurora MySQL que sirve como coordinador de consultas.

De forma predeterminada, sin consulta en paralelo, el procesamiento de una consulta de Aurora implica la transmisión de datos sin procesar a un solo nodo del clúster de Aurora (el nodo principal). Aurora luego realiza todo el procesamiento adicional para esa consulta en un solo hilo en ese único nodo. Con las consultas paralelas, gran parte de este trabajo con uso intensivo de E/S y de CPU se delega a los nodos de la capa de almacenamiento. Solo las filas compactas del conjunto

de resultados se transmiten de vuelta al nodo director, con las filas ya filtradas y los valores de la columna ya extraídos y transformados. El beneficio del rendimiento viene de la reducción del tráfico de red, la reducción del uso de CPU en el nodo director y la paralelización de la E/S en los nodos de almacenamiento. La cantidad de E/S, filtrado y proyección paralelos es independiente del número de instancias de base de datos del clúster de Aurora que ejecute la consulta.

Requisitos previos

Para utilizar todas las características de la consulta paralela, se requiere un clúster de base de datos de Aurora MySQL que ejecute la versión 2.09 o una posterior. Si ya tiene un clúster que desea utilizar con consulta paralela, puede actualizarlo a una versión compatible y activar la consulta paralela después. En ese caso, asegúrese de seguir el procedimiento de actualización en Consideraciones para actualizar las consultas paralelas ya que los nombres de las opciones de configuración y los valores predeterminados son diferentes en estas versiones más recientes.

Las instancias de base de datos del clúster deben utilizar las clases de instancia `db.r*`.

Asegúrese de que la optimización de combinación hash esté activada para su clúster. Para saber cómo hacerlo, consulte Activación de una combinación hash para clústeres de consultas paralelas.

Para personalizar parámetros como `aurora_parallel_query` y `aurora_disable_hash_join`, debe tener un grupo de parámetros personalizado que utilice con el clúster. Puede especificar estos parámetros individualmente para cada instancia de base de datos mediante un grupo de parámetros de base de datos. Sin embargo, se recomienda especificarlos en un grupo de parámetros de clúster de base de datos. De esta forma, todas las instancias de base de datos del clúster heredan la misma configuración para estos parámetros.

Limitaciones

La característica de consultas en paralelo tiene las siguientes limitaciones:

- La configuración de almacenamiento del clúster de base de datos Aurora I/O-Optimized no admite consultas paralelas.
- No puede usar la consulta paralela con las clases de instancia db.t2 o db.t3. Esta limitación se aplica incluso si solicita la consulta paralela mediante la variable de sesión `aurora_pq_force`.
- La consulta paralela no se aplica a las tablas que utilizan los formatos de fila `COMPRESSED` o `REDUNDANT`. Utilice los formatos de fila `COMPACT` o `DYNAMIC` para las tablas que piensa utilizar con consultas paralelas.

- Aurora utiliza un algoritmo basado en costos para determinar si utilizar el mecanismo de consulta paralela para cada instrucción SQL. El uso de ciertas construcciones SQL en una instrucción puede evitar la consulta paralela o hacer que la consulta paralela sea poco probable para esa instrucción. Para obtener información acerca de la compatibilidad de construcciones SQL con la consulta paralela, consulte Constructos de SQL para consultas paralelas en Aurora MySQL.
- Cada instancia de base de datos Aurora solo puede ejecutar un número determinado de sesiones de consultas en paralelo a la vez. Si una consulta tiene varias partes que usan consultas en paralelo, como subconsultas, uniones u operadores `UNION`, esas fases se ejecutan en secuencia. La instrucción solo cuenta como una única sesión de consulta en paralelo cada vez. Puede monitorizar el número de sesiones activas usando las variables de estado de consultas en paralelo. Puede comprobar el límite de sesiones simultáneas para una instancia de base de datos determinada consultando la variable de estado `Aurora_pq_max_concurrent_requests`.
- La consulta en paralelo está disponible en todas las Regiones de AWS compatibles con Aurora. Para la mayoría de las Regiones de AWS, la versión de Aurora MySQL mínima requerida para utilizar la consulta en paralelo es 2.09.
- La consulta paralela está diseñada para mejorar el rendimiento de las consultas que realizan un uso intensivo de datos. No está diseñada para consultas ligeras.
- Le recomendamos que utilice nodos de lector para las instrucciones SELECT, especialmente las que realizan un uso intensivo de datos.

Costos de E/S con consulta paralela

Si su clúster de Aurora MySQL utiliza una consulta en paralelo, es posible que vea un aumento en los valores `VolumeReadIOPS`. Las consultas en paralelo no utilizan el grupo de búfer. Por lo tanto, si bien las consultas son rápidas, este procesamiento optimizado puede dar como resultado un aumento de las operaciones de lectura y los cargos asociados.

Los costos de E/S de consulta paralela de su consulta se miden en la capa de almacenamiento y serán iguales o mayores si la consulta paralela está activada. La ventaja es que mejora el rendimiento de las consultas. Hay dos razones por las que los costos de E/S pueden ser más altos con la consulta paralela:

- Aunque algunos de los datos de una tabla se encuentren en el grupo de búferes, la consulta paralela requiere que todos los datos se analicen en la capa de almacenamiento, lo que genera costos de E/S.

- La ejecución de una consulta paralela no prepara el grupo de búferes. Como resultado, las ejecuciones consecutivas de la misma consulta paralela generan el costo total de E/S.

Creación de un clúster de base de datos de consultas paralelas en Aurora MySQL

Para crear un clúster de Aurora MySQL con consultas en paralelo, agregarle nuevas instancias o realizar otras operaciones administrativas, utilice la misma Consola de administración de AWS y técnicas de AWS CLI que usaría con otros clústeres de Aurora MySQL. Puede crear un nuevo clúster para trabajar con consultas en paralelo. También puede crear un clúster de base de datos para trabajar con consultas paralelas mediante la restauración de una instantánea de un clúster de base de datos de Aurora compatible con MySQL. Si no está familiarizado con el proceso de crear un nuevo clúster de Aurora MySQL, puede encontrar información general y requisitos previos en Creación de un clúster de base de datos de Amazon Aurora.

Cuando elija una versión del motor de Aurora MySQL, le recomendamos que elija la versión más reciente disponible. En la actualidad, todas las versiones de Aurora MySQL disponibles admiten consultas en paralelo. Tiene más flexibilidad para activar y desactivar consultas paralelas o utilizar consultas paralelas con clústeres existentes si utiliza las últimas versiones.

Tanto si crea un nuevo clúster como si lo restaura de una instantánea, se usan las mismas técnicas para añadir nuevas instancias de base de datos que usaría con otros clústeres de Aurora MySQL.

Puede crear un clúster de consultas paralelas mediante la consola de Amazon RDS o la AWS CLI.

Contenido

- Creación de un clúster de consultas paralelas utilizando la consola
- Creación de un clúster de consultas paralelas utilizando la CLI

Creación de un clúster de consultas paralelas utilizando la consola

Puede crear un nuevo clúster de consultas en paralelo con la consola como se describe a continuación.

Para crear un clúster de consultas en paralelo con la Consola de administración de AWS

1. Siga el procedimiento general de la Consola de administración de AWS de Creación de un clúster de base de datos de Amazon Aurora.

2. Para Tipo de motor, elija Aurora MySQL.
3. En Configuración adicional, elija un grupo de parámetros que creó para Grupo de parámetros de clúster de base de datos. El uso de dicho grupo de parámetros personalizados es necesario para Aurora MySQL 2.09 y versiones posteriores. En el grupo de parámetros de clúster de base de datos, especifique la configuración de los parámetros `aurora_parallel_query=ON` y `aurora_disable_hash_join=OFF`. Al hacerlo, se activa la consulta paralela para el clúster y se activa la optimización de combinación hash que funciona en combinación con la consulta paralela.

Para verificar que un nuevo clúster puede usar consultas en paralelo

1. Cree un clúster utilizando la técnica anterior.
2. (Para Aurora MySQL versión 2 o 3) Compruebe que la opción de configuración de `aurora_parallel_query` es verdadera.

```
mysql> select @@aurora_parallel_query;
+-------------------------+
| @@aurora_parallel_query |
+-------------------------+
|                       1 |
+-------------------------+
```

3. (Para Aurora MySQL versión 2) Compruebe que la configuración de `aurora_disable_hash_join` está definida en falsa.

```
mysql> select @@aurora_disable_hash_join;
+----------------------------+
| @@aurora_disable_hash_join |
+----------------------------+
|                          0 |
+----------------------------+
```

4. Con algunas tablas grandes y consultas con uso intensivo de datos, compruebe los planes de consulta para confirmar que algunas de las consultas utilizan la optimización de consultas paralelas. Para ello, siga el procedimiento en Cómo comprobar qué instrucciones usan consultas paralelas para Aurora MySQL.

Creación de un clúster de consultas paralelas utilizando la CLI

Puede crear un nuevo clúster de consultas en paralelo con la CLI como se describe a continuación.

Para crear un clúster de consultas en paralelo con la AWS CLI

1. (Opcional) Compruebe qué versiones de Aurora MySQL son compatibles con clústeres de consultas paralelas. Para ello, utilice el comando `describe-db-engine-versions` y compruebe el valor del campo `SupportsParallelQuery`. Para ver un ejemplo, consulte Comprobación de la compatibilidad de la versión de Aurora MySQL para consulta paralela.
2. (Opcional) Cree un grupo de parámetros de clúster de base de datos personalizado con la configuración `aurora_parallel_query=ON` y `aurora_disable_hash_join=OFF`. Utilice comandos como los siguientes.

   ```
   aws rds create-db-cluster-parameter-group --db-parameter-group-family aurora-
   mysql8.0 --db-cluster-parameter-group-name pq-enabled-80-compatible
   aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
   enabled-80-compatible \
     --parameters
    ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
   aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
   enabled-80-compatible \
     --parameters
    ParameterName=aurora_disable_hash_join,ParameterValue=OFF,ApplyMethod=pending-
   reboot
   ```

   Si realiza este paso, especifique la opción `--db-cluster-parameter-group-name` *`my_cluster_parameter_group`* en la instrucción `create-db-cluster` posterior. Sustituya el nombre de su propio grupo de parámetros. Si omite este paso, creará el grupo de parámetros y lo asociará con el clúster más adelante, como se describe en Activación y desactivación de las consultas paralelas en Aurora MySQL.
3. Siga el procedimiento general de la AWS CLI de Creación de un clúster de base de datos de Amazon Aurora.
4. Especifique el siguiente conjunto de opciones:

   - Para la opción `--engine`, use `aurora-mysql`. Estos valores producen clústeres de consultas paralelas compatibles con MySQL 5.7 o 8.0.
   - Para la opción `--db-cluster-parameter-group-name`, especifique el nombre de un grupo de parámetros de clúster de base de datos que creó y especificó el valor del parámetro

`aurora_parallel_query=ON`. Si omite esta opción, puede crear el clúster con un grupo de parámetros predeterminado y modificarlo posteriormente para utilizar dicho grupo de parámetros personalizado.

- Para la opción `--engine-version`, utilice una versión de Aurora MySQL compatible con la consulta paralela. Utilice el procedimiento de Optimización de consultas paralelas en Aurora MySQL para obtener una lista de versiones si es necesario.

  En el siguiente ejemplo de código se muestra cómo. Sustituya su propio valor por cada una de las variables de entorno, como *`$clúster_ID`*. En este ejemplo también se especifica la opción `--manage-master-user-password` para generar la contraseña del usuario maestro y administrarla en Secrets Manager. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager. También puede utilizar la opción `--master-password` para especificar y administrar la contraseña usted mismo.

```
aws rds create-db-cluster --db-cluster-identifier $CLUSTER_ID \
  --engine aurora-mysql --engine-version 8.0.mysql_aurora.3.04.1 \
  --master-username $MASTER_USER_ID --manage-master-user-password \
  --db-cluster-parameter-group-name $CUSTOM_CLUSTER_PARAM_GROUP

aws rds create-db-instance --db-instance-identifier ${INSTANCE_ID}-1 \
  --engine same_value_as_in_create_cluster_command \
  --db-cluster-identifier $CLUSTER_ID --db-instance-class $INSTANCE_CLASS
```

5. Verificar que un clúster que ha creado o restaurado tiene la característica de consultas en paralelo disponible.

   Compruebe que la configuración de `aurora_parallel_query` existe. Si esta configuración tiene el valor 1, la consulta paralela está lista para su uso. Si esta configuración tiene el valor 0, establézcala en 1 antes de poder utilizar la consulta paralela. De cualquier manera, el clúster es capaz de realizar consultas paralelas.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Para restaurar una instantánea a un clúster de consultas en paralelo con la AWS CLI

1. Compruebe qué versiones de Aurora MySQL son compatibles con clústeres de consultas paralelas. Para ello, utilice el comando `describe-db-engine-versions` y compruebe el valor del campo `SupportsParallelQuery`. Para ver un ejemplo, consulte Comprobación de la compatibilidad de la versión de Aurora MySQL para consulta paralela. Decida qué versión usar para el clúster restaurado.
2. Localice una instantánea de clúster compatible con Aurora MySQL.
3. Siga el procedimiento general de la AWS CLI de Restauración de una instantánea de clúster de base de datos.

   ```
   aws rds restore-db-cluster-from-snapshot \
     --db-cluster-identifier mynewdbcluster \
     --snapshot-identifier mydbclustersnapshot \
     --engine aurora-mysql
   ```

4. Verificar que un clúster que ha creado o restaurado tiene la característica de consultas en paralelo disponible. Utilice el mismo procedimiento de verificación que en Creación de un clúster de consultas paralelas utilizando la CLI.

Activación y desactivación de las consultas paralelas en Aurora MySQL

Cuando las consultas en paralelo están activadas, Aurora MySQL determina si usarlas en el tiempo de ejecución para cada consulta. En el caso de uniones, operaciones UNION, subconsultas, etc., Aurora MySQL determina si usar consultas en paralelo en el tiempo de ejecución para cada bloque de consulta. Para más detalles, consulte Cómo comprobar qué instrucciones usan consultas paralelas para Aurora MySQL y Constructos de SQL para consultas paralelas en Aurora MySQL.

Puede activar y desactivar las consultas paralelas de forma dinámica en los niveles global y de sesión para una instancia de base de datos usando la opción aurora_parallel_query. Puede cambiar la configuración de `aurora_parallel_query` del grupo de clústeres de base de datos para activar o desactivar la consulta paralela de forma predeterminada.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
```

```
+------------------------+
```

Para activar o desactivar el parámetro de `aurora_parallel_query` en el nivel de sesión, utilice los métodos estándar para cambiar la opción de configuración de un cliente. Por ejemplo, puede hacerlo a través de la línea de comandos de `mysql` o dentro de una aplicación JDBC u ODBC. El comando en el cliente MySQL estándar es `set session aurora_parallel_query = {'ON'/'OFF'}`. También puede agregar el parámetro de nivel de sesión a la configuración de JDBC o en su código de aplicación para activar o desactivar las consultas en paralelo de forma dinámica.

Puede cambiar permanentemente la configuración del parámetro de `aurora_parallel_query`, ya sea para una instancia de base de datos específica o para todo el clúster. Si especifica el valor del parámetro en un grupo de parámetros de base de datos, ese valor solo se aplica a una instancia de base de datos específica del clúster. Si especifica el valor del parámetro en un grupo de parámetros de clúster de base de datos, todas las instancias de base de datos del clúster heredarán la misma configuración. Para activar o desactivar el parámetro de `aurora_parallel_query`, use las técnicas para trabajar con grupos de parámetros, como se describe en Grupos de parámetros para Amazon Aurora. Sigue estos pasos:

1. Cree un grupo de parámetros de clúster personalizado (recomendado) o un grupo de parámetros de base de datos personalizado.
2. En este grupo de parámetros, actualice `parallel_query` al valor que desee.
3. En función de si ha creado un grupo de parámetros de clúster de base de datos o un grupo de parámetros de base de datos, asocie el grupo de parámetros con el clúster de Aurora o con las instancias de base de datos específicas en las que piensa utilizar la característica de consulta paralela.

> **Tip**
>
> Dado que `aurora_parallel_query` es un parámetro dinámico, no es necesario reiniciar el clúster después de cambiar esta configuración. Sin embargo, cualquier conexión que estuviera utilizando una consulta paralela antes de cambiar la opción continuará haciéndolo hasta que se cierre la conexión o se reinicie la instancia.

Puede modificar el parámetro de consultas paralelas usando la operación de la API ModifyDBclústerParameterGroup o ModifyDBParameterGroup o la Consola de administración de AWS.

Puede activar la combinación hash para los clústeres de consultas paralelas, activar y desactivar las consultas paralelas mediante la consola de Amazon RDS o la AWS CLI y anular el optimizador de consultas paralelas.

Contenido

- Activación de una combinación hash para clústeres de consultas paralelas
- Activación y desactivación de la consulta paralela mediante la consola
- Activar y desactivar la consulta paralela mediante la CLI
- Anulación del optimizador de consultas paralelas

Activación de una combinación hash para clústeres de consultas paralelas

La consulta en paralelo se utiliza típicamente para los tipos de consultas que consumen más recursos que se benefician de la optimización de la combinación hash. Por lo tanto, es útil asegurarse de que las combinaciones hash están activadas para los clústeres en los que piensa usar consultas paralelas. Para obtener información acerca de cómo utilizar combinaciones hash de manera eficaz, consulte Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash.

Activación y desactivación de la consulta paralela mediante la consola

Puede activar o desactivar las consultas paralelas en el nivel de instancia de base de datos o el nivel de clúster de base de datos mediante el trabajo con grupos de parámetros.

Para activar o desactivar la consulta paralela de un clúster de base de datos con la Consola de administración de AWS

1. Cree un grupo de parámetros personalizados según se indica en Grupos de parámetros para Amazon Aurora.
2. Actualice aurora_parallel_query a 1 (activado) o 0 (desactivado). Para los clústeres en los que la característica de consultas paralelas está disponible, aurora_parallel_query está desactivado de forma predeterminada.
3. Si utiliza un grupo de parámetros de clúster personalizado, asócielo con el clúster de base de datos de Aurora en el que piensa utilizar la característica de consulta paralela. Si utiliza un grupo de parámetros de base de datos personalizado, asócielo con una o más instancias de base de datos en el clúster. Se recomienda utilizar un grupo de parámetros de clúster. Al hacerlo, se

asegura de que todas las instancias de base de datos del clúster tienen la misma configuración para las consultas paralelas y las características asociadas, como la combinación hash.

Activar y desactivar la consulta paralela mediante la CLI

Puede modificar el parámetro de consultas paralelas mediante el comando `modify-db-cluster-parameter-group` o `modify-db-parameter-group`. Elija el comando apropiado en función de si especifica el valor de `aurora_parallel_query` a través de un grupo de parámetros de clúster de base de datos o un grupo de parámetros de base de datos.

Para activar o desactivar las consultas paralelas para un clúster de base de datos con la CLI

- Modifique el parámetro de consultas en paralelo usando el comando `modify-db-cluster-parameter-group`. Utilice un comando como el siguiente. Sustituya el nombre apropiado por su propio grupo de parámetros personalizado. Sustituya `ON` u `OFF` por la parte de `ParameterValue` de la opción de `--parameters`.

```
$ aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "cluster_param_group_name"
}

aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters ParameterName=aurora_pq,ParameterValue=ON,ApplyMethod=pending-reboot
```

También puede activar o desactivar las consultas en paralelo en el nivel de sesión, por ejemplo, mediante la línea de comandos `mysql` o en una aplicación JDBC u ODBC. Para ello, use los métodos estándares para cambiar un ajuste de configuración de cliente. Por ejemplo, el comando en el cliente MySQL estándar es `set session aurora_parallel_query = {'ON'/'OFF'}` para Aurora MySQL.

También puede agregar el parámetro de nivel de sesión a la configuración de JDBC o en su código de aplicación para activar o desactivar las consultas en paralelo de forma dinámica.

Anulación del optimizador de consultas paralelas

Puede utilizar la variable de sesión `aurora_pq_force` para anular el optimizador de consultas paralelas y solicitar una consulta paralela para cada consulta. Le recomendamos que lo haga solo con fines de prueba. El siguiente ejemplo muestra cómo usar `aurora_pq_force` en una sesión.

```
set SESSION aurora_parallel_query = ON;
set SESSION aurora_pq_force = ON;
```

Para desactivar la anulación, haga lo siguiente:

```
set SESSION aurora_pq_force = OFF;
```

Optimización de consultas paralelas en Aurora MySQL

A fin de optimizar su clúster de base de datos para consultas paralelas, piense en qué clústeres de bases de datos mejorarían con la consulta paralela y en si deberían actualizarse para ello. A continuación, ajuste la carga de trabajo y cree objetos de esquema para la consulta paralela.

Contenido

- Planificación de un clúster de consultas paralelas
  - Comprobación de la compatibilidad de la versión de Aurora MySQL para consulta paralela
- Consideraciones para actualizar las consultas paralelas
  - Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL
  - Actualización a Aurora MySQL 2.09 y versiones posteriores
- Ajuste del rendimiento de las consultas paralelas
- Creación de objetos de esquema para aprovechar las consultas paralelas

Planificación de un clúster de consultas paralelas

La planificación de un clúster de base de datos que tiene consultas paralelas activadas requiere tomar algunas decisiones. Estas incluyen la realización de pasos de configuración (ya sea la creación o restauración de un clúster de Aurora MySQL completo) y la determinación de la amplitud de la activación de consultas paralelas en el clúster de base de datos.

Considere lo siguiente como parte de la planificación:

- Si utiliza Aurora MySQL que es compatible con MySQL 5.7, debe crear un clúster aprovisionado. A continuación, se activa la consulta paralela mediante el parámetro `aurora_parallel_query`.

  Si tiene un clúster de Aurora MySQL existente, no es necesario crear un clúster nuevo para utilizar la consulta paralela. Puede asociar el clúster o instancias de base de datos específicas en el clúster, con un grupo de parámetros que tiene activado el parámetro de `aurora_parallel_query`. Al hacerlo, puede reducir el tiempo y el esfuerzo para configurar los datos relevantes para usar con consulta paralela.
- Planifique las tablas grandes que necesite reorganizar para que pueda utilizar la consulta paralela al acceder a ellas. Es posible que pueda necesitar crear nuevas versiones de algunas tablas grandes donde la consulta paralela es útil. Por ejemplo, es posible que tenga que eliminar índices de búsqueda de texto completo. Para obtener más información, consulte Creación de objetos de esquema para aprovechar las consultas paralelas.

Comprobación de la compatibilidad de la versión de Aurora MySQL para consulta paralela

Para verificar qué versiones de Aurora MySQL son compatibles con clústeres de consultas en paralelo, utilice el comando `describe-db-engine-versions` de la AWS CLI y verifique el valor del campo `SupportsParallelQuery`. En el siguiente ejemplo de código se muestra cómo comprobar si las combinaciones están disponibles para los clústeres de consultas en paralelo en una región de AWS. Asegúrese de especificar la cadena de parámetro completa de `--query` en una sola línea.

```
aws rds describe-db-engine-versions --region us-east-1 --engine aurora-mysql \
--query '*[]|[?SupportsParallelQuery == `true`].[EngineVersion]' --output text
```

Los comandos anteriores producen un resultado similar al siguiente. Es posible que el resultado varíe en función de las versiones de Aurora MySQL disponibles en la región de AWS especificada.

```
5.7.mysql_aurora.2.11.1
5.7.mysql_aurora.2.11.2
5.7.mysql_aurora.2.11.3
5.7.mysql_aurora.2.11.4
5.7.mysql_aurora.2.11.5
5.7.mysql_aurora.2.11.6
5.7.mysql_aurora.2.12.0
5.7.mysql_aurora.2.12.1
5.7.mysql_aurora.2.12.2
5.7.mysql_aurora.2.12.3
```

```
5.7.mysql_aurora.2.12.4
8.0.mysql_aurora.3.04.0
8.0.mysql_aurora.3.04.1
8.0.mysql_aurora.3.04.2
8.0.mysql_aurora.3.04.3
8.0.mysql_aurora.3.05.2
8.0.mysql_aurora.3.06.0
8.0.mysql_aurora.3.06.1
8.0.mysql_aurora.3.07.0
8.0.mysql_aurora.3.07.1
```

Después de comenzar a utilizar la consulta paralela con un clúster, puede monitorear el rendimiento y eliminar los obstáculos al uso de consultas paralelas. Para obtener esas instrucciones, consulte Ajuste del rendimiento de las consultas paralelas.

Consideraciones para actualizar las consultas paralelas

En función de las versiones original y de destino al actualizar un clúster de consultas paralelo, es posible que encuentre mejoras en los tipos de consultas que la consulta paralela puede optimizar. Es posible que también descubra que no necesita especificar un parámetro de modo de motor especial para la consulta paralela. En las secciones siguientes se explican las consideraciones al actualizar un clúster que tiene activada la consulta paralela.

Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL

Varias instrucciones de SQL, cláusulas y tipos de datos tienen compatibilidad con consultas paralelas nuevas o mejoradas a partir de Aurora MySQL versión 3. Cuando actualice desde una versión anterior a la versión 3, verifique si las consultas adicionales pueden beneficiarse de las optimizaciones de consultas paralelas. Para obtener información sobre estas mejoras de consultas paralelas, consulte Tipos de datos de columna, Tablas particionadas, y Funciones de agregación, cláusulas GROUP BY y cláusulas HAVING.

Si va a actualizar un clúster de consultas paralelas desde Aurora MySQL 2.08 o una versión anterior, obtenga información sobre los cambios en la forma de activar la consulta paralela. Para ello lea Actualización a Aurora MySQL 2.09 y versiones posteriores.

En la versión 3 de Aurora MySQL, la optimización de combinaciones hash está activada de manera predeterminada. La opción de configuración de `aurora_disable_hash_join` de versiones anteriores no se utiliza.

Actualización a Aurora MySQL 2.09 y versiones posteriores

En Aurora MySQL 2.09 y versiones posteriores, la consulta paralela funciona para clústeres aprovisionados y no requiere el parámetro de modo de motor `parallelquery`. Por lo tanto, no es necesario crear un nuevo clúster ni restaurar desde una instantánea existente para utilizar la consulta paralela con estas versiones. Puede utilizar los procedimientos de actualización descritos en Actualización de la versión secundaria o el nivel de parche de un clúster de bases de datos Aurora MySQL para actualizar el clúster a dicha versión. Puede actualizar un clúster anterior independientemente de si era un clúster de consultas paralelas o un clúster aprovisionado. Para reducir el número de opciones en el menú Engine version (Versión del motor), puede elegir Show versions that support the parallel query feature (Mostrar versiones compatibles con la característica de consulta paralela) para filtrar las entradas de ese menú. A continuación, elija Aurora MySQL 2.09 o una versión posterior.

Después de actualizar un clúster de consultas paralelas anterior a Aurora MySQL 2.09 o una versión posterior, se activa la consulta paralela en el clúster actualizado. La consulta paralela está desactivada de forma predeterminada en estas versiones y el procedimiento para habilitarla es diferente. La optimización de combinación de hash también está desactivada de forma predeterminada y se debe activar por separado. Por lo tanto, asegúrese de activar esta configuración de nuevo después de la actualización. Para obtener instrucciones sobre cómo hacerlo, consulte Activación y desactivación de las consultas paralelas en Aurora MySQL y Activación de una combinación hash para clústeres de consultas paralelas.

En particular, se activa la consulta paralela utilizando los parámetros de configuración `aurora_parallel_query=ON` y `aurora_disable_hash_join=OFF` en lugar de `aurora_pq_supported` y `aurora_pq`. Los parámetros `aurora_pq_supported` y `aurora_pq` están obsoletos en las versiones de Aurora MySQL más recientes.

En el clúster actualizado, el atributo de `EngineMode` tiene el valor `provisioned` en lugar de `parallelquery`. Para comprobar si la consulta paralela está disponible para una versión de motor especificada, compruebe ahora el valor del campo de `SupportsParallelQuery` en la salida del comando de la `describe-db-engine-versions` AWS CLI. En versiones de Aurora MySQL anteriores, comprobó la presencia de `parallelquery` en la lista de `SupportedEngineModes`.

Después de actualizar a Aurora MySQL 2.09 o una versión posterior, puede aprovechar las siguientes características. Estas características no están disponibles para clústeres de consultas paralelas que ejecutan versiones de Aurora MySQL anteriores.

- Performance Insights. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.
- Búsqueda de datos anteriores. Para obtener más información, consulte Búsqueda de datos anteriores de un clúster de base de datos de Aurora.
- Parar y comenzar el clúster. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon Aurora.

Ajuste del rendimiento de las consultas paralelas

Para administrar el rendimiento de una carga de trabajo con consultas en paralelo, asegúrese de que dicha consulta en paralelo se usa para consultas en las que esta optimización ayuda más.

Para ello, puede hacer lo siguiente:

- Asegúrese de que las tablas más grandes son compatibles con la consulta paralela. Es posible que pueda cambiar las propiedades de la tabla o volver a crear algunas tablas para que las consultas de esas tablas puedan aprovechar la optimización de consultas paralelas. Para saber cómo hacerlo, consulte Creación de objetos de esquema para aprovechar las consultas paralelas.
- Monitorizar qué consultas usan consultas en paralelo. Para saber cómo hacerlo, consulte Monitorización de consultas paralelas para Aurora MySQL.
- Compruebe que la consulta paralela se utiliza para las consultas con mayor volumen de datos y de larga ejecución, y con el nivel adecuado de simultaneidad para su carga de trabajo. Para saber cómo hacerlo, consulte Cómo comprobar qué instrucciones usan consultas paralelas para Aurora MySQL.
- Ajuste el código SQL para activar la consulta paralela y que se aplique a las consultas que espera. Para saber cómo hacerlo, consulte Constructos de SQL para consultas paralelas en Aurora MySQL.

Creación de objetos de esquema para aprovechar las consultas paralelas

Antes de crear o modificar tablas que piensa utilizar para consultas paralelas, asegúrese de familiarizarse con los requisitos descritos en Requisitos previos y Limitaciones.

Dado que la consulta paralela requiere que las tablas usen la configuración de `ROW_FORMAT=Compact` o `ROW_FORMAT=Dynamic`, compruebe las opciones de configuración de Aurora para realizar cualquier cambio en la opción de configuración de `INNODB_FILE_FORMAT`.

Emita la instrucción `SHOW TABLE STATUS` para confirmar el formato de fila para todas las tablas en una base de datos.

Antes de cambiar el esquema para activar la consulta paralela para trabajar con más tablas, asegúrese de probar. Las pruebas deben confirmar si la consulta paralela da como resultado un incremento neto en el rendimiento de esas tablas. También asegúrese de que los requisitos del esquema de una consulta en paralelo con compatibles en los demás aspectos con sus objetivos.

Por ejemplo, antes de cambiar de `ROW_FORMAT=Compressed` a `ROW_FORMAT=Compact` o `ROW_FORMAT=Dynamic`, pruebe el rendimiento de las cargas de trabajo para las tablas originales y nuevas. Además, tenga en cuenta otros efectos potenciales como un mayor volumen de datos.

Cómo comprobar qué instrucciones usan consultas paralelas para Aurora MySQL

En el funcionamiento habitual, no es necesario realizar acciones especiales para sacar partido de las consultas en paralelo. Cuando una consulta cumple los requisitos esenciales para consultas en paralelo, el optimizador de consultas decide automáticamente si usar las consultas en paralelo para cada consulta específica.

Si ejecuta experimentos en un entorno de desarrollo o de pruebas, es posible que observe que las consultas paralelas no se utilizan porque las tablas son demasiado pequeñas en número de filas o volumen de datos general. Los datos de la tabla también podrían encontrarse completamente en el grupo de búfer, especialmente en el caso de tablas que haya creado recientemente para realizar experimentos.

Cuando monitorea o ajusta el rendimiento de clústeres, asegúrese de decidir si se utilizan las consultas paralelas en los contextos adecuados. Podría ajustar el esquema de la base de datos, la configuración, las consultas SQL o incluso la topología de clústeres y la configuración de conexión de aplicación para aprovechar esta característica.

Para comprobar si una consulta utiliza consultas paralelas, compruebe el plan de consulta (también conocido como el "plan de explicación") ejecutando la instrucción EXPLAIN. En el caso de ejemplos sobre cómo afectan las instrucciones, cláusulas y expresiones de SQL a los resultados de `EXPLAIN` para consultas en paralelo, consulte Constructos de SQL para consultas paralelas en Aurora MySQL.

En el siguiente ejemplo se muestra la diferencia entre un plan de consultas tradicional y un plan de consultas en paralelo. Este plan de explicación procede de la consulta 3 de la comparativa TPC-H. En muchas de las consultas de ejemplo de esta sección se usan las tablas del conjunto de datos de

TPC-H. Puede obtener las definiciones de tabla, consultas y el programa dbgen que genera datos de ejemplo desde el sitio web de TPC-H.

```
EXPLAIN SELECT l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) AS revenue,
  o_orderdate,
  o_shippriority
FROM customer,
  orders,
  lineitem
WHERE c_mktsegment = 'AUTOMOBILE'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < date '1995-03-13'
AND l_shipdate > date '1995-03-13'
GROUP BY l_orderkey,
  o_orderdate,
  o_shippriority
ORDER BY revenue DESC,
  o_orderdate LIMIT 10;
```

De forma predeterminada, es posible que la consulta tenga un plan como el siguiente. Si no ve la combinación hash utilizada en el plan de consulta, asegúrese de que la optimización esté activada primero.

```
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra                                              |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |  1480234 |    10.00 | Using where; Using temporary; Using filesort       |
|  1 | SIMPLE      | orders   | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 14875240 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
|  1 | SIMPLE      | lineitem | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 59270573 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
```

En la versión 3 de Aurora MySQL, puede activar la combinación hash en el nivel de sesión mediante la siguiente instrucción.

```
SET optimizer_switch='block_nested_loop=on';
```

Para la versión 2.09 y versiones posteriores de Aurora MySQL, establezca el parámetro de base de datos o el parámetro de clúster de base de datos `aurora_disable_hash_join` en `0` (desactivado). Si `aurora_disable_hash_join` está desactivado, se establece el valor de `optimizer_switch` en `hash_join=on`.

Después de activar la combinación hash, intente ejecutar la instrucción `EXPLAIN` de nuevo. Para obtener información acerca de cómo utilizar combinaciones hash de manera eficaz, consulte Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash.

Con las combinaciones hash activadas pero las consultas paralelas desactivadas, es posible que la consulta tenga un plan como el siguiente, que usa combinaciones hash, pero no consultas paralelas.

```
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
| id | select_type | table    |...| rows      | Extra
                          |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
|  1 | SIMPLE      | customer |...|   5798330 | Using where; Using index; Using
 temporary; Using filesort       |
|  1 | SIMPLE      | orders   |...| 154545408 | Using where; Using join buffer (Hash
 Join Outer table orders)   |
|  1 | SIMPLE      | lineitem |...| 606119300 | Using where; Using join buffer (Hash
 Join Outer table lineitem) |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
```

Después de que las consultas paralelas estén activadas, dos pasos de este plan de consulta pueden usar la optimización de consultas paralelas, como se muestra en la columna `Extra` de la salida de `EXPLAIN`. El procesamiento con uso intensivo de E/S y de CPU para estos pasos se baja a la capa de almacenamiento.

```
+----+...
+-------------------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                                    |
```

```
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using index; Using temporary; Using filesort
                                                      |
|  1 |...| Using where; Using join buffer (Hash Join Outer table orders); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra)   |
|  1 |...| Using where; Using join buffer (Hash Join Outer table lineitem); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Para obtener información sobre cómo interpretar las salidas de `EXPLAIN` para una consulta en paralelo y las partes de las instrucciones de SQL a las que las consultas en paralelo pueden aplicarse, consulte Constructos de SQL para consultas paralelas en Aurora MySQL.

En la siguiente salida de ejemplo se muestran los resultados de ejecutar la consulta anterior en una instancia de db.r4.2xlarge con un grupo de búfer frío. La consulta se ejecuta notablemente más rápido al usar las consultas en paralelo.

> **Note**
>
> Dado que los tiempos dependen de muchos factores del entorno, sus resultados podrían ser diferentes. Realice siempre sus propias pruebas de rendimiento para confirmar los resultados con su propio entorno, carga de trabajo, etc.

```
-- Without parallel query
+------------+-------------+-------------+----------------+
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (24 min 49.99 sec)
```

```
-- With parallel query
+------------+-------------+-------------+----------------+
```

```
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (1 min 49.91 sec)
```

Muchas de las consultas de ejemplo de esta sección usan tablas de este conjunto de datos de TPC-H, en particular la tabla PART, que tiene 20 millones de filas y la siguiente definición.

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| p_partkey     | int(11)       | NO   | PRI | NULL    |       |
| p_name        | varchar(55)   | NO   |     | NULL    |       |
| p_mfgr        | char(25)      | NO   |     | NULL    |       |
| p_brand       | char(10)      | NO   |     | NULL    |       |
| p_type        | varchar(25)   | NO   |     | NULL    |       |
| p_size        | int(11)       | NO   |     | NULL    |       |
| p_container   | char(10)      | NO   |     | NULL    |       |
| p_retailprice | decimal(15,2) | NO   |     | NULL    |       |
| p_comment     | varchar(23)   | NO   |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Experimente con su carga de trabajo para averiguar si las instrucciones de SQL individuales pueden sacar partido de las consultas en paralelo. A continuación, use las siguientes técnicas de monitoreo para ayudar a comprobar la frecuencia con la que se usa la consulta paralela en las cargas de trabajo reales a lo largo del tiempo. En el caso de cargas de trabajo reales, se aplican otros factores adicionales, como los límites de simultaneidad.

Monitorización de consultas paralelas para Aurora MySQL

Si su clúster de Aurora MySQL utiliza una consulta en paralelo, es posible que vea un aumento en los valores `VolumeReadIOPS`. Las consultas en paralelo no utilizan el grupo de búfer. Por lo tanto, si bien las consultas son rápidas, este procesamiento optimizado puede dar como resultado un aumento de las operaciones de lectura y los cargos asociados.

Además de las métricas de Amazon CloudWatch descritas en Consulta de métricas en la consola de Amazon RDS, Aurora proporciona otras variables de estado globales. Puede utilizar estas

variables de estado global para ayudar a monitorear la ejecución de consultas paralelas. Pueden darle información sobre por qué es posible que el optimizador utilice o no la consulta paralela en una situación determinada. Para acceder a estas variables, puede usar el comando `SHOW GLOBAL STATUS`. También puede encontrar estas variables enumeradas a continuación.

Una sesión de consultas en paralelo no es necesariamente un mapeo uno a uno con las consultas realizadas por la base de datos. Por ejemplo, suponga que su plan de consulta tiene dos pasos que usan la consulta paralela. En tal caso, la consulta implica dos sesiones paralelas y los contadores de intentos de solicitud y de solicitudes correctas se incrementan en dos.

Cuando experimente con las consultas en paralelo emitiendo instrucciones `EXPLAIN`, espere ver aumentos en los contadores designados como no elegidos incluso aunque las consultas no se estén ejecutando reamente. Cuando trabaje con consultas en paralelo en producción, puede comprobar si los contadores de no elegido están aumentando más rápido de lo que espera. En este punto, puede ajustar para que la consulta paralela se ejecute para las consultas que espera. Para ello, puede cambiar la configuración del clúster, la mezcla de consultas, las instancias de base de datos donde la consulta paralela está activada, etc.

Se realiza un seguimiento de estos contadores en el nivel de instancia de base de datos. Cuando se conecte a un punto de enlace distinto, podría ver métricas diferentes porque cada instancia de base de datos ejecuta su propio conjunto de consultas en paralelo. También podría ver métricas diferentes cuando el punto de enlace del lector se conecte a una instancia de base de datos distinta para cada sesión.

Nombre	Descripción
`Aurora_pq_bytes_returned`	El número de bytes de estructuras de datos de tuplas transmitido al nodo director durante las consultas en paralelo. Debe dividirse entre 16 384 para compararse con `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	El número máximo de sesiones de consultas en paralelo que se pueden ejecutar simultáneamente en esta instancia de base de datos Aurora. Este es un número fijo que depende de la clase de instancia de base de datos de AWS.

Nombre	Descripción
`Aurora_pq_pages_pushed_down`	El número de páginas de datos (cada una con un tamaño fijo de 16 KiB) en las que las consultas en paralelo evitaron una transmisión de red al nodo director.
`Aurora_pq_request_attempted`	El número de sesiones de consultas en paralelo solicitadas. Este valor podría representar más de una sesión por consulta, dependiendo de los constructos de SQL como subconsultas y uniones.
`Aurora_pq_request_executed`	El número de sesiones de consultas en paralelo ejecutadas correctamente.
`Aurora_pq_request_failed`	El número de sesiones de consultas en paralelo que devolvieron un error al cliente. En algunos casos, una solicitud de una consulta en paralelo podría producir un error, por ejemplo, debido a un problema en la capa de almacenamiento. En tales casos, la parte de la consulta que haya producido un error vuelve a intentarse usando el mecanismo de consultas no paralelas. Si la consulta reintenta da también produce un error, se devolverá un error al cliente y este contador se incrementará.
`Aurora_pq_request_in_progress`	El número de sesiones de consultas en paralelo en curso actualmente. Este número se aplica a la instancia de base de datos de Aurora concreta a la que se conecta, no a todo el clúster de base de datos de Aurora. Para ver si una instancia de base de datos está cerca de su límite de simultaneidad, compare este valor con el de `Aurora_pq_max_concurrent_requests` .

Nombre	Descripción
`Aurora_pq_request_not_chosen`	El número de veces que las consultas en paralelo no se han elegido para satisfacer una consulta. Este valor es la suma de varios otros contadores más detallados. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_below_min_rows`	El número de veces que las consultas en paralelo no se han elegido debido al número de filas de la tabla. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_column_bit`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas debido a un tipo de datos no admitido en la lista de columnas proyectadas.
`Aurora_pq_request_not_chosen_column_geometry`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene columnas con el tipo de datos `GEOMETRY`. Para obtener información acerca de las versiones de Aurora MySQL que eliminan esta limitación, consulte Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL.

Nombre	Descripción
`Aurora_pq_request_not_chosen_column_lob`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene columnas con un tipo de datos `LOB` o columnas `VARCHAR` que se almacenan externamente debido a la longitud declarada. Para obtener información acerca de las versiones de Aurora MySQL que eliminan esta limitación, consulte Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL.
`Aurora_pq_request_not_chosen_column_virtual`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla contiene una columna virtual.
`Aurora_pq_request_not_chosen_custom_charset`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene columnas con un conjunto de caracteres personalizado.
`Aurora_pq_request_not_chosen_fast_ddl`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque actualmente se altera la tabla por una instrucción `ALTER` DDL rápida.
`Aurora_pq_request_not_chosen_ few_pages_outside_buffer_pool`	El número de veces que las consultas en paralelo no se han elegido, incluso aunque menos del 95 % de los datos de tabla ya estuviera en el grupo de búfer, porque no había suficientes datos de tabla fuera de búfer para que valiera la pena realizar una consulta en paralelo.

Nombre	Descripción
`Aurora_pq_request_not_chosen_full_text_index`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene índices de texto completo.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	El número de veces que las consultas en paralelo no se han elegido debido a que un porcentaje elevado de datos de tabla (actualmente, superior al 95 %) ya estaba en el grupo de búfer. En estos casos, el optimizador determina que leer los datos del grupo de búfer es más eficiente. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_index_hint`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta incluye una sugerencia de índice.
`Aurora_pq_request_not_chosen_innodb_table_format`	El número de solicitudes de consulta en paralelo que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla utiliza un formato de fila de InnoDB no admitido. La consulta en paralelo de Aurora solo se aplica a los formatos de fila `COMPACT`, `REDUNDANT` y `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	El número de solicitudes de consultas en paralelo que usaron la ruta de procesamiento de consultas no en paralelo, debido a que la consulta se estaba iniciando en una transacción de ejecución prolongada. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.

Nombre	Descripción
`Aurora_pq_request_not_chosen_no_where_clause`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta no incluye ninguna cláusula `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta utiliza un análisis de intervalo en un índice.
`Aurora_pq_request_not_chosen_row_length_too_long`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la longitud total combinada de todas las columnas es demasiado larga.
`Aurora_pq_request_not_chosen_small_table`	El número de veces que las consultas en paralelo no se han elegido debido al tamaño general de la tabla, según lo determinado por el número de filas y la longitud promedio de las filas. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_temporary_table`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta hace referencia a tablas temporales que utilizan los tipos de tabla `MyISAM` o `memory` no admitidos.

Nombre	Descripción
`Aurora_pq_request_not_chosen_tx_isolation`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta utiliza un nivel de aislamiento de transacciones no admitido. En las instancias de base de datos del lector, la consulta paralela solo se aplica a los niveles de aislamiento `REPEATABLE READ` y `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta forma parte de una instrucción `UPDATE` o `DELETE`.
`Aurora_pq_request_not_chosen_unsupported_access`	El número de solicitudes de consultas en paralelo que usan la ruta de procesamiento de consultas no en paralelo porque la cláusula `WHERE` no cumple los criterios de consultas en paralelo. Este resultado puede producirse si la consulta no requiere un análisis de uso intensivo de datos o si la consulta es una instrucción `DELETE` o `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	El número de solicitudes de consultas en paralelo que usan la ruta de procesamiento de consultas no en paralelo porque el clúster de base de datos de Aurora MySQL no utiliza una configuración de almacenamiento de clúster de Aurora compatible. Este parámetro está disponible en la versión 3.04 y versiones posteriores de Aurora MySQL. Para obtener más información, consulte Limitaciones.

Nombre	Descripción
`Aurora_pq_request_throttled`	El número de veces que las consultas en paralelo no se han elegido debido a que el número máximo de consultas en paralelo simultáneas que ya se están ejecutando en una instancia de base de datos Aurora concreta.

Constructos de SQL para consultas paralelas en Aurora MySQL

En la siguiente sección, puede encontrar información más detallada acerca de por qué determinadas instrucciones de SQL usan o no usan consulta paralela. En esta sección también se detalla cómo interactúan las características de Aurora MySQL con la consulta paralela. Esta información le ayudará a diagnosticar los problemas de rendimiento para un clúster que use consultas paralelas o a comprender cómo se aplican las consultas paralelas a su carga de trabajo concreta.

La decisión de usar consultas en paralelo depende de muchos factores que tienen lugar en el momento en que se ejecuta la instrucción. Por tanto, la consulta en paralelo podría usarse para determinadas consultas siempre, nunca o solo en ciertas condiciones.

> **Tip**
>
> Cuando consulte estos ejemplos en HTML, puede utilizar el widget Copy (Copiar) en la esquina superior derecha de cada descripción de código para copiar el código SQL y probarlo usted mismo. El uso del widget Copy (Copiar) evita copiar los caracteres adicionales alrededor de las líneas de solicitud de `mysql>` y de continuación de `->`.

Temas

- Instrucción EXPLAIN
- Cláusula WHERE
- Lenguaje de definición de datos (DDL)
- Tipos de datos de columna
- Tablas particionadas
- Funciones de agregación, cláusulas GROUP BY y cláusulas HAVING
- Llamadas de función en la cláusula WHERE

- Cláusula LIMIT
- Operadores de comparación
- Uniones
- subconsultas
- UNION
- Vistas
- Instrucciones de lenguaje de manipulación de datos (DML)
- Transacciones y bloqueo
- Índices de árbol B
- Índices de búsqueda de texto completo (FTS)
- Virtual columns
- Mecanismos de almacenamiento en caché integrados
- Sugerencias del optimizador
- Tablas temporales MyISAM

Instrucción EXPLAIN

Como se muestra en los ejemplos de esta sección, la instrucción `EXPLAIN` indica si cada fase de una consulta es apta actualmente para las consultas en paralelo. También indica qué aspectos de una consulta se pueden bajar a la capa de almacenamiento. Los elementos más importantes del plan de consulta son los siguientes:

- Un valor distinto de `NULL` para la columna `key` indica que la consulta se puede realizar con eficacia usando búsquedas del índice y que es poco probable el uso de consultas en paralelo.
- Un valor pequeño para la columna `rows` (un valor que no sea de millones) indica que la consulta no accede a suficientes datos para que valga la pena realizar consultas paralelas. Esto significa que la consulta paralela es poco probable.
- La columna `Extra` muestra si se espera usar consultas en paralelo. Este resultado tiene el aspecto del siguiente ejemplo.

  ```
  Using parallel query (A columns, B filters, C exprs; D extra)
  ```

  El número de `columns` representa a cuántas columnas se hace referencia en el bloque de consultas.

El número de `filters` representa el número de predicados de `WHERE` que representan una simple comparación de un valor de columna con una constante. La comparación puede ser de igualdad, desigualdad o rango. Aurora puede paralelizar estos tipos de predicados con más eficacia.

El número de `exprs` representa el número de expresiones como las llamadas de función, operadores u otras expresiones que también se pueden paralelizar, aunque no con la misma eficacia que una condición de filtro.

El número `extra` representa cuántas expresiones no se pueden bajar y las realiza el nodo director.

Por ejemplo, fíjese en el siguiente resultado de `EXPLAIN`.

```
mysql> explain select p_name, p_mfgr from part
    -> where p_brand is not null
    -> and upper(p_type) is not null
    -> and round(p_retailprice) is not null;
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table |...| rows     | Extra
                                 |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part  |...| 20427936 | Using where; Using parallel query (5
 columns, 1 filters, 2 exprs; 0 extra) |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
```

La información de la columna `Extra` muestra que se extraen cinco columnas de cada fila para evaluar las condiciones de consulta y construir el conjunto de resultados. Un predicado `WHERE` implica un filtro, es decir, una columna que se prueba directamente en la cláusula `WHERE`. Dos cláusulas `WHERE` requieren la evaluación de expresiones más complicadas, en este caso implican llamadas de función. El campo `0 extra` confirma que todas las operaciones de la cláusula `WHERE` se bajan a la capa de almacenamiento como parte del procesamiento de consultas en paralelo.

En los casos en los que las consultas en paralelo no se eligen, normalmente se puede deducir el motivo de las otras columnas de la salida `EXPLAIN`. Por ejemplo, el valor de `rows` podría ser demasiado pequeño o la columna `possible_keys` podría indicar que la consulta puede usar

búsqueda de índice en lugar de un análisis de uso intensivo de datos. En el ejemplo siguiente se muestra una consulta en la que el optimizador puede estimar que la consulta solo analizará un pequeño número de filas. Lo hace en función de las características de la clave principal. En este caso, las consultas en paralelo no se requieren.

```
mysql> explain select count(*) from part where p_partkey between 1 and 100;
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows |
 Extra                    |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
|  1 | SIMPLE      | part  | range | PRIMARY       | PRIMARY | 4       | NULL |   99 |
 Using where; Using index |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
```

La salida que muestra si se usarán las consultas en paralelo tiene en cuenta todos los factores disponibles en el momento en que se ejecuta la instrucción `EXPLAIN`. El optimizador podría realizar una elección distinta cuando la consulta se ejecute realmente, si la situación ha cambiado mientras tanto. Por ejemplo, `EXPLAIN` podría notificar que una instrucción usará consultas en paralelo. Pero cuando la consulta se ejecute realmente más tarde, podría no usar consultas en paralelo en función de las condiciones de ese momento. Estas condiciones pueden incluir otras consultas paralelas que se ejecutan simultáneamente. También pueden incluir filas que se eliminan de la tabla, la creación de un nuevo índice, el paso de demasiado tiempo dentro de una transacción abierta, etc.

Cláusula WHERE

Para que una consulta use la optimización de consultas en paralelo, debe incluir una cláusula `WHERE`.

La optimización de consultas en paralelo acelera muchos tipos de expresiones usadas en la cláusula:`WHERE`

- Comparaciones simples de un valor de columna con una constante, conocidos como filtros. Estas comparaciones se benefician más de bajarse a la capa de almacenamiento. El número de expresiones de filtro de una consulta se notifica en la salida de `EXPLAIN`.
- Otros tipos de expresiones de la cláusula `WHERE` también se bajan a la capa de almacenamiento cuando es posible. El número de expresiones de ese tipo en una consulta se notifica en la salida de `EXPLAIN`. Dichas expresiones pueden ser llamadas de función, operadores `LIKE`, expresiones `CASE`, etc.

- Determinadas funciones y operadores actualmente no se bajan mediante consultas en paralelo. El número de expresiones de ese tipo en una consulta se notifica como `extra` en la salida de `EXPLAIN`. El resto de la consulta puede seguir usando consultas en paralelo.
- Aunque las expresiones de la lista de selección no se bajan, las consultas que contengan tales funciones siguen pudiendo beneficiarse del tráfico de red reducido por los resultados intermedios de las consultas en paralelo. Por ejemplo, las consultas que llaman a funciones de agregación en la lista de selección pueden beneficiarse de las consultas en paralelo, incluso aunque las funciones de agregación no se bajen.

Por ejemplo, la siguiente consulta realiza un análisis de tabla completa y procesa todos los valores de la columna `P_BRAND`. Sin embargo, no usa consultas en paralelo porque la consulta no incluye ninguna cláusula `WHERE`.

```
mysql> explain select count(*), p_brand from part group by p_brand;
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows     |
 Extra                            |
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
|  1 | SIMPLE      | part  | ALL  | NULL          | NULL | NULL    | NULL | 20427936 |
 Using temporary; Using filesort |
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
```

Por el contrario, la siguiente consulta incluye predicados de `WHERE` que filtran los resultados, de forma que se pueden aplicar las consultas en paralelo:

```
mysql> explain select count(*), p_brand from part where p_name is not null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
    -> group by p_brand;
+----+...+----------
+-------------------------------------------------------------------------------------------
+
| id |...| rows     | Extra
                                          |
+----+...+----------
+-------------------------------------------------------------------------------------------
+
```

```
|  1 |...| 20427936 | Using where; Using temporary; Using filesort; Using parallel
 query (5 columns, 1 filters, 2 exprs; 0 extra) |
+----+...+----------
+-----------------------------------------------------------------------------------------
+
```

Si el optimizador estima que el número de filas devueltas para un bloque de consulta es pequeño, las consultas en paralelo no se usan para ese bloque de consultas. En el siguiente ejemplo se muestra un caso en el que un operador de mayor que de la columna de clave principal se aplica a un millón de filas, lo que causa que se usen las consultas en paralelo. La prueba contraria de menor que se estima que se aplicará solo a unas pocas filas y no usa las consultas en paralelo.

```
mysql> explain select count(*) from part where p_partkey > 10;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+

mysql> explain select count(*) from part where p_partkey < 10;
+----+...+------+--------------------------+
| id |...| rows | Extra                    |
+----+...+------+--------------------------+
|  1 |...|    9 | Using where; Using index |
+----+...+------+--------------------------+
```

Lenguaje de definición de datos (DDL)

En la versión 2 de Aurora MySQL, las consultas en paralelo solo están disponibles para las tablas que no tienen pendientes operaciones de lenguaje de definición de datos (DDL) rápidas. En Aurora MySQL versión 3, puede utilizar una consulta paralela en una tabla al mismo tiempo que una operación DDL instantánea.

El DDL instantáneo en la versión 3 de Aurora MySQL reemplaza a la característica de DDL rápida en la versión 2 de Aurora MySQL. Para obtener información sobre DDL instantáneo, consulte DDL instantáneo (Aurora MySQL versión 3).

Tipos de datos de columna

En Aurora MySQL versión 3, la consulta paralela puede funcionar con tablas que contienen columnas con tipos de datos `TEXT`, `BLOB`, `JSON`, y `GEOMETRY`. También puede funcionar con las columnas `VARCHAR` y `CHAR` con una longitud máxima declarada superior a 768 bytes. Si la consulta hace referencia a cualquier columna que contenga tipos de objetos tan grandes, el trabajo adicional para recuperarlos agrega cierta sobrecarga al procesamiento de consultas. En ese caso, verifique si la consulta puede omitir las referencias a esas columnas. De lo contrario, ejecute puntos de referencia para confirmar si estas consultas son más rápidas con la consulta paralela activada o desactivada.

En la versión 2 de Aurora MySQL, la consulta paralela tiene estas limitaciones para los tipos de objetos grandes:

- Los tipos de datos `TEXT`, `BLOB`, `JSON` y `GEOMETRY` no son compatibles con las consultas paralelas. Una consulta que haga referencia a cualquier columna de estos tipos no puede usar consultas en paralelo.
- Las columnas de longitud variable (tipos de datos `VARCHAR` y `CHAR`) son compatibles con las consultas en paralelo hasta una longitud máxima declarada de 768 bytes. Una consulta que haga referencia a cualquier columna de los tipos declarados con una longitud máxima más larga no puede usar consultas en paralelo. En el caso de columnas que usen conjuntos de caracteres multibyte, el límite de bytes tiene en cuenta el número máximo de bytes del conjunto de caracteres. Por ejemplo, para el conjunto de caracteres `utf8mb4` (que tiene una longitud máxima de caracteres de 4 bytes), una columna `VARCHAR(192)` es compatible con una consulta en paralelo, pero una columna `VARCHAR(193)` no lo es.

Tablas particionadas

Puede utilizar tablas particionadas con consultas en paralelo en Aurora MySQL versión 3. Dado que las tablas particionadas se representan internamente como varias tablas más pequeñas, una consulta que utiliza una consulta paralela en una tabla no particionada podría no utilizar una consulta paralela en una tabla particionada idéntica. Aurora MySQL considera si cada partición es lo suficientemente grande como para calificar para la optimización de consultas paralelas, en lugar de evaluar el tamaño de toda la tabla. Verifique si la variable de estado `Aurora_pq_request_not_chosen_small_table` se incrementa si una consulta de una tabla particionada no utiliza una consulta paralela cuando se espera que lo haga.

Por ejemplo, considere una tabla particionada con `PARTITION BY HASH (`*`column`*`) PARTITIONS 2` y otra tabla particionada con `PARTITION BY HASH (`*`column`*`) PARTITIONS 10`. En la tabla

con dos particiones, las particiones son cinco veces más grandes que la tabla con diez particiones. Por lo tanto, es más probable que las consultas paralelas se utilicen para consultas en la tabla con menos particiones. En el siguiente ejemplo, la tabla `PART_BIG_PARTITIONS` tiene dos particiones y `PART_SMALL_PARTITIONS` tiene diez particiones. Con datos idénticos, es más probable que las consultas paralelas se utilicen para la tabla con menos particiones grandes.

```
mysql> explain select count(*), p_brand from part_big_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+
| id | select_type | table               | partitions | Extra
                                                                                           |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+
|  1 | SIMPLE      | part_big_partitions | p0,p1      | Using where; Using temporary;
 Using parallel query (4 columns, 1 filters, 1 exprs; 0 extra; 1 group-bys, 1 aggrs) |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+

mysql> explain select count(*), p_brand from part_small_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+----------------------+------------------------------
+------------------------------+
| id | select_type | table                | partitions                   | Extra
                   |
+----+-------------+----------------------+------------------------------
+------------------------------+
|  1 | SIMPLE      | part_small_partitions | p0,p1,p2,p3,p4,p5,p6,p7,p8,p9 | Using
 where; Using temporary |
+----+-------------+----------------------+------------------------------
+------------------------------+
```

Funciones de agregación, cláusulas GROUP BY y cláusulas HAVING

Las consultas que implican funciones de agregación suelen ser buenas candidatas para las consultas en paralelo porque implican el análisis de números grandes de filas en tablas grandes.

En Aurora MySQL 3, la consulta paralela puede optimizar las llamadas a funciones agregadas en la lista de selección y en la cláusula `HAVING`.

Antes de la versión 3 de Aurora MySQL, las llamadas de función agregadas en la lista de selección o la cláusula `HAVING` no se bajan a la capa de almacenamiento. Sin embargo, las consultas en paralelo siguen mejorando el rendimiento de tales consultas con las funciones de agregación. Para ello, primero extraen los valores de columna de las páginas de datos sin procesar en paralelo en la capa de almacenamiento. A continuación, transmiten dichos valores de vuelta al nodo director en formato compacto de tupla en lugar de como páginas de datos completas. Como siempre, la consulta requiere al menos un predicado `WHERE` para que las consultas en paralelo se activen.

En los siguientes ejemplos simples se ilustran los tipos de consultas de agregación que pueden aprovechar las consultas en paralelo. Para ello, devuelven los resultados intermedios en formato compacto al nodo director, filtran las filas que no coinciden de los resultados intermedios o ambos.

```
mysql> explain select sql_no_cache count(distinct p_brand) from part where p_mfgr =
 'Manufacturer#5';
+----+...+-----------------------------------------------------------------------------+
| id |...| Extra                                                                       |
+----+...+-----------------------------------------------------------------------------+
|  1 |...| Using where; Using parallel query (2 columns, 1 filters, 0 exprs; 0 extra) |
+----+...+-----------------------------------------------------------------------------+

mysql> explain select sql_no_cache p_mfgr from part where p_retailprice > 1000 group by
 p_mfgr having count(*) > 100;
+----+...
+-----------------------------------------------------------------------------------------------
+
| id |...| Extra
                                  |
+----+...
+-----------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (3
 columns, 0 filters, 1 exprs; 0 extra) |
```

```
+----+...
+-------------------------------------------------------------------------------------------
+
```

Llamadas de función en la cláusula WHERE

Aurora puede aplicar la optimización de consultas en paralelo a llamadas a la mayoría de las funciones integradas en la cláusula `WHERE`. La paralelización de estas llamadas de función descarga algo de trabajo de la CPU del nodo director. La evaluación de las funciones de predicado en paralelo durante la fase inicial de consulta ayuda a Aurora a minimizar la cantidad de datos que se transmiten y procesan durante las fases posteriores.

Actualmente, la paralelización no se aplica a llamadas de función de la lista de selección. Dichas funciones las evalúa el nodo director, incluso aunque aparezca una función idéntica en la cláusula `WHERE`. Los valores originales de las columnas relevantes se incluyen en las tuplas transmitidas desde los nodos de almacenamiento de vuelta al nodo director. El nodo director realiza las transformaciones, como `UPPER`, `CONCATENATE`, etc., para producir los valores finales del conjunto de resultados.

En el siguiente ejemplo, la consulta paralela paraleliza la llamada a `LOWER` porque aparece en la cláusula `WHERE`. Las consultas paralelas no afectan a las llamadas a `SUBSTR` y `UPPER` porque aparecen en la lista de selección.

```
mysql> explain select sql_no_cache distinct substr(upper(p_name),1,5) from part
    -> where lower(p_name) like '%cornflower%' or lower(p_name) like '%goldenrod%';
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
                |
+----+...
+-------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using parallel query (2 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------
+
```

La misma consideración se aplica a otras expresiones, como las expresiones `CASE` o los operadores `LIKE`. Por ejemplo, en el siguiente ejemplo se muestra que las consultas en paralelo evalúan la expresión `CASE` y los operadores `LIKE` de la cláusula `WHERE`.

```
mysql> explain select p_mfgr, p_retailprice from part
    -> where p_retailprice > case p_mfgr
    ->   when 'Manufacturer#1' then 1000
    ->   when 'Manufacturer#2' then 1200
    ->   else 950
    -> end
    -> and p_name like '%vanilla%'
    -> group by p_retailprice;
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                |
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (4
 columns, 0 filters, 2 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Cláusula LIMIT

Actualmente, las consultas en paralelo no se utilizan para ningún bloque de consultas que incluya una cláusula `LIMIT`. Aun así, las consultas en paralelo podrían usarse para fases de consulta anteriores con `GROUP BY`, `ORDER BY` o uniones.

Operadores de comparación

El optimizador estima cuántas filas analizar para evaluar los operadores de comparación y determina si usar las consultas en paralelo en función de dicha estimación.

El primer ejemplo a continuación muestra que puede realizarse una comparación de igualdad con respecto a la columna de clave principal con eficacia sin usar consultas en paralelo. El segundo ejemplo a continuación muestra que una comparación similar con respecto a una columna no indexada requiere el análisis de millones de filas y, por tanto, puede sacar partido de las consultas en paralelo.

```
mysql> explain select * from part where p_partkey = 10;
+----+...+------+-------+
| id |...| rows | Extra |
+----+...+------+-------+
|  1 |...|    1 | NULL  |
+----+...+------+-------+

mysql> explain select * from part where p_type = 'LARGE BRUSHED BRASS';
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

Se aplican las mismas consideraciones para pruebas de desigualdad y para comparaciones de rango, como de menor que, mayor que, igual a, o `BETWEEN`. El optimizador estima el número de filas que analizar y determina si vale la pena usar consultas en paralelo en función del volumen general de E/S.

Uniones

Las consultas de unión con tablas grandes suelen implicar operaciones con un uso intensivo de datos que se benefician de la optimización de consultas en paralelo. Las comparaciones de valores de columnas entre tablas múltiples (es decir, los predicados de unión en sí) actualmente no se paralelizan. Sin embargo, las consultas en paralelo pueden bajar algunos de los procesamientos internos para otras fases de unión, como la construcción del filtro de Bloom durante una unión de hash. Las consultas en paralelo se pueden aplicar a consultas de unión incluso sin una cláusula `WHERE`. Por tanto, una consulta de unión es una excepción a la regla de que se requiere una cláusula `WHERE` para usar las consultas en paralelo.

Cada fase del procesamiento de unión se evalúa para comprobar si es apto para las consultas en paralelo. Si varias fases pueden usar consultas en paralelo, estas fases se realizan en secuencia. Por tanto, cada consulta de unión cuenta como una sola sesión de consultas en paralelo en términos de límites de simultaneidad.

Por ejemplo, cuando una consulta incluye predicados `WHERE` para filtrar las filas de una de las tablas unidas, esa opción de filtrado puede usar las consultas en paralelo. Como otro ejemplo, suponga que una consulta de unión usa el mecanismo de unión de hash, por ejemplo, para unir una tabla grande con una tabla pequeña. En este caso, el análisis de tabla para producir la estructura de datos de filtro de Bloom podría usar consultas en paralelo.

> **Note**
>
> La consulta en paralelo se utiliza típicamente para los tipos de consultas que consumen más recursos que se benefician de la optimización de la combinación hash. El método para activar la optimización de uniones hash depende de la versión de Aurora MySQL. Para obtener información acerca de cada versión, consulte Activación de una combinación hash para clústeres de consultas paralelas. Para obtener información acerca de cómo utilizar combinaciones hash de manera eficaz, consulte Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash.

```
mysql> explain select count(*) from orders join customer where o_custkey = c_custkey;
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
| id |...| table    | type  | possible_keys | key         |...| rows      | Extra

              |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
|  1 |...| customer | index | PRIMARY       | c_nationkey |...|  15051972 | Using index

              |
|  1 |...| orders   | ALL   | o_custkey     | NULL        |...| 154545408 | Using join
 buffer (Hash Join Outer table orders); Using parallel query (1 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
```

En el caso de una consulta de unión que use el mecanismo de bucle anidado, el bloque de bucle anidado más exterior podría usar consultas en paralelo. El uso de las consultas en paralelo depende

de los factores habituales, como la presencia de condiciones de filtro adicionales en la cláusula `WHERE`.

```
mysql> -- Nested loop join with extra filter conditions can use parallel query.
mysql> explain select count(*) from part, partsupp where p_partkey != ps_partkey and
 p_name is not null and ps_availqty > 0;
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table    |...| rows     | Extra
                                 |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part     |...| 20427936 | Using where; Using parallel query (2
 columns, 1 filters, 0 exprs; 0 extra) |
|  1 | SIMPLE      | partsupp |...| 78164450 | Using where; Using join buffer (Block
 Nested Loop)                       |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
```

subconsultas

El bloque de consulta externa y el bloque de subconsulta interna es posible que usen cada una la consulta paralela o no. Si lo hacen es en función de las características habituales de la tabla, cláusula `WHERE`, etc., para cada bloque. Por ejemplo, la siguiente consulta usa consultas en paralelo para el bloque de subconsulta, pero no el bloque exterior.

```
mysql> explain select count(*) from part where
   --> p_partkey < (select max(p_partkey) from part where p_name like '%vanilla%');
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type |...| rows     | Extra
                       |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY     |...|     NULL | Impossible WHERE noticed after reading const tables
                       |
|  2 | SUBQUERY    |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

Actualmente, las subconsultas correlacionadas no pueden usar la optimización de consultas en paralelo.

UNION

Cada bloque de consulta de una consulta de `UNION` puede usar o no usar consultas en paralelo, en función de las características habituales de la tabla, la cláusula `WHERE`, etc., para cada parte de `UNION`

```
mysql> explain select p_partkey from part where p_name like '%choco_ate%'
    -> union select p_partkey from part where p_name like '%vanil_a%';
+----+----------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type    |...| rows     | Extra
                          |
+----+----------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY        |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
|  2 | UNION          |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
| NULL | UNION RESULT | <union1,2> |...|     NULL | Using temporary
                       |
+----+--------------+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Cada cláusula `UNION` de la consulta se ejecuta secuencialmente. Incluso aunque la consulta incluya varias fases que todas usen consultas en paralelo, solo ejecuta una sola consulta en paralelo cada vez. Por tanto, incluso una consulta multifase compleja solo cuenta como 1 para el límite de consultas en paralelo simultáneas.

Vistas

El optimizador reescribe cualquier consulta que use una vista como una consulta mayor que use las tablas subyacentes. Por tanto, las consultas en paralelo funcionan igual tanto si las referencias de tabla son vistas como si son tablas reales. Las mismas consideraciones sobre si usar consultas en

paralelo para una consulta y qué partes se deben bajar también se aplican a la consulta reescrita final.

Por ejemplo, el siguiente plan de consulta muestra una definición de consulta que normalmente no usa consultas paralelas. Cuando se consulta la vista con cláusulas `WHERE` adicionales, Aurora MySQL usa consultas en paralelo.

```
mysql> create view part_view as select * from part;
mysql> explain select count(*) from part_view where p_partkey is not null;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 0 filters, 0 exprs;
 1 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

Instrucciones de lenguaje de manipulación de datos (DML)

La instrucción `INSERT` puede usar consultas en paralelo para la fase `SELECT` de procesamiento, si la parte `SELECT` cumple las demás condiciones para las consultas en paralelo.

```
mysql> create table part_subset like part;
mysql> explain insert into part_subset select * from part where p_mfgr =
 'Manufacturer#1';
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Normalmente, después de una instrucción `INSERT`, los datos de las filas recién insertadas se encuentran en el grupo de búfer. Por tanto, una tabla podría no ser apta para las consultas en paralelo inmediatamente después de insertar un gran número de filas. Más tarde, cuando los datos se desalojen del grupo de búfer durante el funcionamiento normal, las consultas con respecto a la tabla podrían comenzar a usar las consultas en paralelo de nuevo.

La instrucción `CREATE TABLE AS SELECT` no usa consultas en paralelo, incluso aunque la porción `SELECT` de la instrucción fuera apta de otra forma para la consultas en paralelo. El aspecto de DDL de esta instrucción hace que sea incompatible con el procesamiento de consultas en paralelo. Por el contrario, en la instrucción `INSERT ... SELECT`, la porción `SELECT` puede usar consultas en paralelo.

Las consultas en paralelo nunca se usan para las instrucciones `DELETE` o `UPDATE`, independientemente del tamaño de la tabla y los predicados de la cláusula `WHERE`.

```
mysql> explain delete from part where p_name is not null;
+----+-------------+...+----------+-------------+
| id | select_type |...| rows     | Extra       |
+----+-------------+...+----------+-------------+
|  1 | SIMPLE      |...| 20427936 | Using where |
+----+-------------+...+----------+-------------+
```

Transacciones y bloqueo

Puede usar todos los niveles de aislamiento de la instancia principal de Aurora.

En las instancias de base de datos de lector de Aurora, la consulta en paralelo se aplica a las sentencias realizadas bajo el nivel de aislamiento `REPEATABLE READ`. La versión 2.09 o posteriores de Aurora MySQL también pueden usar el nivel de aislamiento `READ COMMITTED` en instancias de base de datos de lector. `REPEATABLE READ` es el nivel de aislamiento predeterminado para las instancias de base de datos de lector de Aurora. Para utilizar el nivel de aislamiento `READ COMMITTED` en instancias de base de datos de lector es necesario establecer la opción de configuración `aurora_read_replica_read_committed` en el nivel de sesión. El nivel de aislamiento de `READ COMMITTED` para las instancias del lector cumple con el comportamiento estándar de SQL. Sin embargo, el aislamiento es menos estricto en las instancias del lector que

cuando las consultas utilizan el nivel de aislamiento de `READ COMMITTED` en la instancia del escritor.

Para obtener más información acerca de los niveles de aislamiento de Aurora, especialmente las diferencias en `READ COMMITTED` entre instancias del escritor y del lector, consulte Niveles de aislamiento de Aurora MySQL.

Después de terminar una gran transacción, las estadísticas de tabla podrían quedar obsoletas. Dichas estadísticas obsoletas podrían requerir una instrucción `ANALYZE TABLE` antes de que Aurora pueda estimar con precisión el número filas. Una instrucción DML a gran escala también podría aportar una porción sustancial de los datos de tabla en el grupo de búfer. Tener estos datos en el grupo de búfer puede dar lugar a que las consultas en paralelo se elijan con menos frecuencia para esa tabla hasta que los datos se desalojen del grupo.

Cuando su sesión esté dentro de una transacción de ejecución prolongada (por ejemplo, 10 minutos), las consultas posteriores dentro de la sesión no usan consultas en paralelo. También puede agotarse el tiempo de espera durante una única consulta de ejecución prolongada. Este tipo de finalización del tiempo de espera podría ocurrir si la consulta se ejecuta durante más del intervalo máximo (actualmente, 10 minutos) antes de que empiece el procesamiento de las consultas en paralelo.

Puede reducir las posibilidades de iniciar transacciones de ejecución prolongada accidentalmente estableciendo `autocommit=1` en las sesiones de `mysql` en las que realiza consultas ad hoc (de una vez). Incluso una instrucción `SELECT` con respecto a una tabla inicia una transacción creando una vista de lectura. Una vista de lectura es un conjunto de datos uniforme para consultas posteriores que permanece hasta que la transacción se confirme. Tenga en cuenta esta restricción también al usar aplicaciones de JDBC u ODBC con Aurora porque tales aplicaciones podrían ejecutarse con el ajuste `autocommit` desactivado.

En el siguiente ejemplo se muestra cómo, con el ajuste `autocommit` desactivado, ejecutar una consulta contra una tabla crea una vista de lectura que inicia implícitamente una transacción. Las consultas que se ejecutan poco después pueden seguir usando consultas en paralelo. Sin embargo, después de una pausa de varios minutos, las consultas ya no son aptas para las consultas en paralelo. Terminar la transacción con `COMMIT` o `ROLLBACK` restaura la elegibilidad de las consultas en paralelo.

```
mysql> set autocommit=0;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
```

```
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+

mysql> select sleep(720); explain select sql_no_cache count(*) from part where
 p_retailprice > 10.0;
+------------+
| sleep(720) |
+------------+
|          0 |
+------------+
1 row in set (12 min 0.00 sec)

+----+...+---------+-------------+
| id |...| rows    | Extra       |
+----+...+---------+-------------+
|  1 |...| 2976129 | Using where |
+----+...+---------+-------------+

mysql> commit;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+
```

Para ver cuántas veces las consultas no fueron aptas para consultas en paralelo porque estaban dentro de transacciones de ejecución prolongada, consulte la variable de estado `Aurora_pq_request_not_chosen_long_trx`.

```
mysql> show global status like '%pq%trx%';
+-------------------------------------+-------+
| Variable_name                       | Value |
+-------------------------------------+-------+
| Aurora_pq_request_not_chosen_long_trx | 4     |
+-------------------------------------+-------+
```

Cualquier instrucción `SELECT` que adquiera bloqueos, como la sintaxis de `SELECT FOR UPDATE` o `SELECT LOCK IN SHARE MODE`, no puede usar consultas en paralelo.

Las consultas en paralelo pueden funcionar para una tabla que esté bloqueada por una instrucción `LOCK TABLES`.

```
mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055';
+----+...+-----------
+-------------------------------------------------------------------------------+
| id |...| rows      | Extra
           |
+----+...+-----------
+-------------------------------------------------------------------------------+
|  1 |...| 154545408 | Using where; Using parallel query (3 columns, 1 filters, 0
 exprs; 0 extra) |
+----+...+-----------
+-------------------------------------------------------------------------------+

mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055' for update;
+----+...+-----------+-------------+
| id |...| rows      | Extra       |
+----+...+-----------+-------------+
|  1 |...| 154545408 | Using where |
+----+...+-----------+-------------+
```

Índices de árbol B

Las estadísticas reunidas por la instrucción `ANALYZE TABLE` ayudan al optimizador a decidir cuándo usar las consultas en paralelo o las búsquedas de índice, según las características de los datos de cada columna. Para mantener las estadísticas actualizadas, ejecute `ANALYZE TABLE` después de las operaciones de DML que realicen cambios sustanciales en los datos de una tabla.

Si las búsquedas de índice pueden realizar una consulta de manera eficaz sin un análisis con uso intensivo de datos, Aurora podría usar búsquedas de índice. Esto evita el gasto general del procesamiento de las consultas en paralelo. También existen límites de simultaneidad sobre el número de consultas en paralelo que se pueden ejecutar a la vez en cualquier clúster de base de datos Aurora. Asegúrese de usar las prácticas recomendadas para indexar sus tablas, de forma que sus consultas más frecuentes y de mayor simultaneidad usen búsquedas de índice.

Índices de búsqueda de texto completo (FTS)

Actualmente, la consulta paralela no se usa para tablas que contengan un índice de búsqueda de texto completo, independientemente de si la consulta se refiere a dichas columnas indexadas o si usa el operador `MATCH`.

Virtual columns

Actualmente, la consulta paralela no se utiliza para tablas que contienen una columna virtual, independientemente de si la consulta se refiere a columnas virtuales.

Mecanismos de almacenamiento en caché integrados

Aurora incluye mecanismos de almacenamiento en caché integrados, es decir, el grupo de búfer y el caché de las consultas. El optimizador de Aurora elige entre estos mecanismos de almacenamiento en caché y las consultas en paralelo en función de cuál es más eficaz para una consulta concreta.

Cuando una consulta en paralelo filtra filas y transforma y extrae valores de columna, los datos se transmiten de vuelta al nodo director como tuplas en lugar de como páginas de datos. Por tanto, ejecutar una consulta en paralelo no añade ninguna página al grupo de búfer ni desaloja páginas que ya estén en el grupo de búfer.

Aurora verifica el número de páginas de datos de tabla que están presentes en el grupo de búfer y qué proporción de los datos de tabla representa ese número. Aurora utiliza esa información para determinar si es más eficaz para usar consultas en paralelo (y omitir los datos del grupo de búfer). De manera alternativa, Aurora podría usar la ruta de ejecución de consultas no paralelas, que usa los datos almacenados en caché en el grupo de búfer. Las páginas que se almacenan en caché y cómo afectan las consultas con uso intensivo de datos al almacenamiento en caché y la expulsión dependen de la configuración relacionada con el grupo de búfer. Por tanto, puede ser difícil predecir si una consulta concreta usará consultas en paralelo porque la elección depende de los datos que cambian en el grupo de búfer.

Además, Aurora impone límites de simultaneidad en las consultas paralelas. Puesto que no todas las consultas usan consultas en paralelo, las tablas a las que acceden múltiples consultas simultáneamente suelen tener una porción importante de sus datos en el grupo de búfer. Por tanto, Aurora no suele elegir estas tablas para consultas en paralelo.

Cuando ejecute una secuencia de consultas no paralelas en la misma tabla, la primera consulta podría ser lenta debido a que los datos no estén en el grupo de búfer. Después, la segunda consulta y las posteriores son mucho más rápidas puesto que el grupo de búfer ha "calentado". Las consultas en paralelo suelen mostrar un rendimiento uniforme desde la primera consulta respecto a la tabla. Al realizar pruebas de rendimiento, haga un análisis comparativo de las consultas no paralelas con un grupo de búfer frío y otro caliente. En algunos casos, los resultados con un grupo de búfer caliente se pueden comparar bien con los tiempos de consultas en paralelo. En estos casos, tenga en cuenta factores como la frecuencia de consultas en esa tabla. También considere si vale la pena mantener los datos de esa tabla en el grupo de búferes.

El caché de la consulta evita volver a ejecutar una consulta cuando se ha enviado una consulta idéntica y los datos de la tabla subyacente no han cambiado. Las consultas optimizadas por la característica de consultas en paralelo pueden ir a la caché de consultas, lo que hace que sean instantáneas la siguiente vez que se ejecuten.

> **Note**
>
> Al realizar comparaciones de rendimiento, la caché de consultas puede producir números de tiempos artificialmente bajos. Por tanto, en situaciones de análisis comparativo, puede usar la señal `sql_no_cache`. Esta señal evita que el resultado se sirva desde la caché de consultas, incluso aunque se haya ejecutado la misma consulta previamente. La señal va inmediatamente después de la instrucción `SELECT` en una consulta. Muchos ejemplos de consultas en paralelo de este tema incluyen esta señal, para que los tiempos de consultas sean comparables entre las versiones de la consulta que están activadas con las consultas en paralelo y las que no.
> Asegúrese de quitar esta señal de su origen cuando pase a usar en producción las consultas en paralelo.

Sugerencias del optimizador

Otra forma de controlar el optimizador es mediante el uso de sugerencias del optimizador, que se pueden especificar en instrucciones individuales. Por ejemplo, puede activar la optimización para

una tabla en una instrucción y, a continuación, desactivarla para otra tabla. Para obtener detalles sobre estas sugerencias, consulte Optimizer Hints (Sugerencias del optimizador) en el Manual de referencia de MySQL.

Puede utilizar consejos de SQL con consultas de Aurora MySQL para ajustar el rendimiento. También puede utilizar sugerencias para evitar que los planes de ejecución de consultas importantes cambien en función de condiciones impredecibles.

Hemos ampliado la característica de sugerencias de SQL para ayudarle a controlar las opciones del optimizador para sus planes de consultas. Estas sugerencias se aplican a las consultas que utilizan la optimización de consultas paralelas. Para obtener más información, consulte Sugerencias de Aurora MySQL.

Tablas temporales MyISAM

La optimización de consultas en paralelo solo se aplica a las tablas InnoDB. Dado que Aurora MySQL usa MyISAM en segundo plano para las tablas temporales, las fases de consultas internas que impliquen tablas temporales nunca usan consultas en paralelo. Estas fases de consultas se indican mediante `Using temporary` en la salida `EXPLAIN`.

Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL

Puede usar la característica de auditoría avanzada de alto desempeño de Amazon Aurora MySQL para auditar la actividad de la base de datos. Para ello, debe habilitar el conjunto de registros de auditoría definiendo varios parámetros de clúster de base de datos. Cuando la auditoría avanzada está habilitada, puede usarla para registrar cualquier combinación de eventos compatibles.

Puede ver o descargar los registros de auditoría para revisar la información de auditoría de una instancia de base de datos a la vez. Para ello, puede utilizar los procedimientos en Supervisión de archivos de registro de Amazon Aurora.

> **Tip**
>
> Para un clúster de base de datos de Aurora que contiene varias instancias de base de datos, podría resultar más conveniente examinar los registros de auditoría de todas las instancias del clúster. Para ello, puede utilizar CloudWatch Logs. Puede activar una configuración en el nivel de clúster para publicar los datos de registro de auditoría de Aurora MySQL en un grupo de registro en CloudWatch. A continuación, puede ver, filtrar y buscar los registros de auditoría a través de la interfaz de CloudWatch. Para obtener más información, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.

Habilitar la auditoría avanzada

Use los parámetros que se describen en esta sección para habilitar y configurar la auditoría avanzada para su clúster de base de datos.

Utilice el parámetro `server_audit_logging` para habilitar o desactivar la auditoría avanzada.

Utilice `server_audit_events` para especificar qué eventos se van a registrar.

Use los parámetros `server_audit_incl_users` y `server_audit_excl_users` para especificar a quién se debe auditar. De forma predeterminada, se auditan todos los usuarios. Para obtener más información sobre cómo funcionan estos parámetros cuando uno o ambos se dejan vacíos o se especifican los mismos nombres de usuario en ambos, consulte server_audit_incl_users y server_audit_excl_users.

Configure la auditoría avanzada definiendo estos parámetros en el grupo de parámetros utilizado por su clúster de base de datos. Puede usar el procedimiento que se muestra en Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora para modificar los parámetros de clúster de base de datos usando la Consola de administración de AWS. Puede utilizar el comando modify-db-cluster-parameter-group de AWS CLI o la operación ModifyDBClusterParameterGroup de la API de Amazon RDS para modificar los parámetros del clúster de base de datos mediante programación.

Para modificar estos parámetros no se requiere un reinicio del clúster de base de datos cuando el grupo de parámetros ya está asociado al clúster. Al asociar el grupo de parámetros al clúster por primera vez, es necesario reiniciar el clúster.

Temas

- server_audit_logging
- server_audit_events
- server_audit_incl_users
- server_audit_excl_users

server_audit_logging

Habilita o deshabilita la auditoría avanzada. Este parámetro tiene el ajuste OFF predeterminado; cámbielo a ON para habilitar la auditoría avanzada.

No aparecen datos de auditoría en los registros a menos que defina también uno o varios tipos de eventos que auditar mediante el parámetro `server_audit_events`.

Para confirmar que se registran los datos de auditoría de una instancia de base de datos, verifique que algunos archivos de registro de esa instancia tengan nombres del formulario `audit/audit.log.`*`other_identifying_information`*. Para ver los nombres de los archivos de registro, siga el procedimiento en Visualización y descripción de archivos de registro de base de datos.

server_audit_events

Contiene una lista delimitada por comas de los eventos que se deben registrar. Los eventos se deben especificar en mayúsculas y no debe haber espacios en blanco entre los elementos de la lista, por ejemplo: `CONNECT,QUERY_DDL`. De manera predeterminada, este parámetro es una cadena vacía.

Puede registrar cualquier combinación de los siguientes eventos:

- CONNECT: registra las conexiones correctas y con error y también las desconexiones. Este evento incluye información de usuario.
- QUERY: registra todas las consultas en texto sin formato, incluidas las que no se pueden completar porque contienen errores de sintaxis o de permisos.

> **Tip**
>
> Con este tipo de evento activado, los datos de auditoría incluyen información sobre la supervisión continua y la información de comprobación de estado que Aurora hace automáticamente. Si solo le interesan determinados tipos de operaciones, puede utilizar los tipos de eventos más específicos. También puede utilizar la interfaz de CloudWatch para buscar en los registros eventos relacionados con bases de datos, tablas o usuarios específicos.

- QUERY_DCL: similar al evento QUERY, pero solo devuelve consultas en lenguaje de control de datos (DCL) (GRANT, REVOKE, etc.).
- QUERY_DDL: similar al evento QUERY, pero solo devuelve consultas en lenguaje de definición de datos (DDL) (CREATE, ALTER, etc.).
- QUERY_DML: similar al evento QUERY, pero solo devuelve consultas en lenguaje de manipulación de datos (DML) (INSERT, UPDATE, etc. y también SELECT).
- TABLE: registra las tablas que se han visto afectadas por la ejecución de la consulta.

> **Note**
>
> Aurora no tiene ningún filtro que excluya determinadas consultas de los registros de auditoría. Para excluir las consultas `SELECT`, debe excluir todas las instrucciones de DML. Si un usuario determinado informa de estas consultas `SELECT` internas en los registros de auditoría, puede excluirlo configurando el parámetro de clúster de base de datos server_audit_excl_users. Sin embargo, si ese usuario también se usa en otras actividades y no se puede omitir, no hay otra opción para excluir las consultas `SELECT`.

server_audit_incl_users

Contiene la lista delimitada por comas de los nombres de los usuarios cuya actividad se registra. No debe haber espacios en blanco entre los elementos de la lista, por ejemplo: `user_3,user_4`. De manera predeterminada, este parámetro es una cadena vacía. La longitud máxima es de 1024 caracteres. Los nombres de usuario especificados deben coincidir con los valores correspondientes de la columna `User` de la tabla `mysql.user`. Para obtener más información acerca de los nombres de usuario, consulte Account User Names and Passwords (Nombres de usuario y contraseñas de cuentas) en la documentación de MySQL.

Si `server_audit_incl_users` y `server_audit_excl_users` están vacíos (ajuste predeterminado), se auditan todos los usuarios.

Si se añaden usuarios a `server_audit_incl_users` y se deja `server_audit_excl_users` vacío, solo se auditan esos usuarios.

Si se agregan usuarios a `server_audit_excl_users` y se deja vacío `server_audit_incl_users`, todos los usuarios se auditan, excepto los enumerados en `server_audit_excl_users`.

Si se agregan los mismos usuarios a `server_audit_excl_users` y a `server_audit_incl_users`, se audita a esos usuarios. Cuando aparece el mismo usuario en ambas configuraciones, `server_audit_incl_users` tiene una mayor prioridad.

Los eventos de conexión y desconexión no se ven afectados por esta variable, siempre se registran si se ha especificado. Un usuario se registra aunque ese usuario también se haya especificado en el parámetro `server_audit_excl_users`, ya que `server_audit_incl_users` tiene una prioridad más alta.

server_audit_excl_users

Contiene la lista delimitada por comas de los nombres de los usuarios cuya actividad no se registra. No debe haber espacios en blanco entre los elementos de la lista, por ejemplo: `rdsadmin,user_1,user_2`. De manera predeterminada, este parámetro es una cadena vacía. La longitud máxima es de 1024 caracteres. Los nombres de usuario especificados deben coincidir con los valores correspondientes de la columna `User` de la tabla `mysql.user`. Para obtener más información acerca de los nombres de usuario, consulte Account User Names and Passwords (Nombres de usuario y contraseñas de cuentas) en la documentación de MySQL.

Si `server_audit_incl_users` y `server_audit_excl_users` están vacíos (ajuste predeterminado), se auditan todos los usuarios.

Si se agregan usuarios a `server_audit_excl_users` y se deja `server_audit_incl_users` vacío, solo se excluyen de la auditoría esos usuarios que se enumeren en `server_audit_excl_users`, y se auditan los restantes.

Si se agregan los mismos usuarios a `server_audit_excl_users` y a `server_audit_incl_users`, se audita a esos usuarios. Cuando aparece el mismo usuario en ambas configuraciones, `server_audit_incl_users` tiene una mayor prioridad.

Los eventos de conexión y desconexión no se ven afectados por esta variable, siempre se registran si se ha especificado. Un usuario se registra si ese usuario también se especifica en el parámetro `server_audit_incl_users`, ya que ese ajuste tiene una prioridad más alta que `server_audit_excl_users`.

Visualización de registros de auditoría

Puede ver y descargar los registros de auditoría mediante la consola. En la página Databases (Bases de datos), elija la instancia de base de datos para mostrar sus detalles y, a continuación, desplácese hasta la sección Logs (Registros). Los registros de auditoría producidos por la característica Auditoría avanzada tienen nombres del formulario `audit/audit.log.`*`other_identifying_information`*.

Para descargar un archivo de registro, elija ese archivo en la sección Logs (Registros) y, a continuación, elija Download (Descargar).

También puede obtener una lista de los archivos de registro usando el comando describe-db-log-files de la AWS CLI. Puede descargar el contenido de un archivo de registro mediante el comando download-db-log-file-portion de la AWS CLI. Para obtener más información, consulte Visualización y descripción de archivos de registro de base de datos y Descarga de un archivo de registro de base de datos.

Detalles de los logs de auditoría

Los archivos de registro se representan como archivos de variables separadas por comas (CSV) en formato UTF-8. Las consultas también se escriben entre comillas simples (').

El registro de auditoría se almacena por separado en el almacenamiento local de cada instancia de base de datos de Aurora MySQL. Cada instancia distribuye escrituras en los cuatro archivos de

registro a la vez. El tamaño máximo de un archivo de registro es de 100 MB. Cuando se alcanza este límite no configurable, Aurora rota el archivo y genera uno nuevo.

> **Tip**
>
> Las entradas del archivo de registro no están en orden secuencial. Para ordenar las entradas, utilice el valor de marca temporal. Para ver los eventos más recientes, es posible que sea necesario revisar todos los archivos de registro. Para obtener más flexibilidad en la ordenación y búsqueda de los datos de registro, active la configuración para cargar los registros de auditoría en CloudWatch y verlos mediante la interfaz de CloudWatch.
> Para ver los datos de auditoría con más tipos de campos y con salida en formato JSON, también puede utilizar la característica Flujos de actividad de base de datos. Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.

Los archivos de registro de auditoría incluyen la siguiente información delimitada por comas en las filas en el orden especificado:

Campo	Descripción
timestamp	Marca temporal de Unix para el evento registrado con una precisión de microsegundos.
serverhost	Nombre de la instancia para la que se ha registrado el evento.
username	Nombre de usuario conectado del usuario.
host	Host desde el que se ha conectado el usuario.
connectionid	Número de ID de conexión de la operación registrada.
queryid	Número de ID de la consulta que se puede usar para buscar los eventos de la tabla relacional y las consultas relacionadas. Para los eventos `TABLE`, se añaden varias líneas.
operación	Tipo de acción registrado. Los posibles valores son: `CONNECT`, `QUERY`, `READ`, `WRITE`, `CREATE`, `ALTER`, `RENAME` y `DROP`.

Campo	Descripción
base de datos	Base de datos activa, definida por el comando `USE`.
objeto	Para los eventos de `QUERY`, este valor indica la consulta realizada por la base de datos. En los eventos `TABLE`, indica el nombre de la tabla.
retcode	Código devuelto de la operación registrada.

Replicación con Amazon Aurora MySQL

Las características de reproducción de Aurora MySQL son claves para la alta disponibilidad y el rendimiento de su clúster. Aurora facilita la creación o el cambio de tamaño de clústeres con hasta 15 réplicas de Aurora.

Todas las réplicas funcionan desde los mismos datos subyacentes. Si algunas instancias de base de datos se quedan sin conexión, otras permanecen disponibles para continuar procesando consultas o para hacerse cargo como escritor si es necesario. Aurora extiende automáticamente las conexiones de solo lectura en varias instancias de base de datos, lo que ayuda a un clúster de Aurora a admitir cargas de trabajo con uso intensivo de consultas.

En los siguientes temas, puede encontrar información sobre cómo funciona la replicación de Aurora MySQL y como ajustar la configuración de replicación para lograr una disponibilidad y un rendimiento óptimos.

Temas

- Uso de réplicas de Aurora
- Opciones de replicación para Amazon Aurora MySQL
- Consideraciones sobre el rendimiento de la replicación de Amazon Aurora MySQL
- Reinicio sin tiempo de inactividad (ZDR) para Amazon Aurora MySQL
- Configuración de filtros de replicación con Aurora MySQL
- Monitoreo de replicación de Amazon Aurora MySQL
- Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS
- Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario)
- Uso de la replicación basada en GTID

Uso de réplicas de Aurora

Las réplicas de Aurora son puntos de enlace independientes de un clúster de base de datos Aurora que se utilizan preferentemente para ajustar la escala de las operaciones de lectura e incrementar la disponibilidad. Se puede distribuir un máximo de 15 réplicas de Aurora entre las distintas zonas de disponibilidad que abarca un clúster de bases de datos dentro de una Región de AWS. Aunque el volumen del clúster de base de datos se compone de varias copias de los datos del clúster de

base de datos, los datos del volumen de clúster se representan como un único volumen lógico para la instancia principal y para las réplicas de Aurora del clúster de base de datos. Para obtener más información acerca de las réplicas de Aurora, consulte Réplicas de Aurora.

Las réplicas de Aurora funcionan bien para el escalado de lectura porque están totalmente dedicadas a las operaciones de lectura en el volumen del clúster. Las operaciones de escritura se administran en la instancia principal. Como el volumen del clúster se comparte entre todas las instancias del clúster de base de datos Aurora MySQL, no se requiere trabajo adicional para replicar una copia de los datos para cada réplica de Aurora. En cambio, las réplicas de lectura de MySQL deben volver a reproducir, en un solo subproceso, todas las operaciones de escritura de la instancia de base de datos de origen en su almacén de datos local. Esta limitación puede afectar a la capacidad de las réplicas de lectura de MySQL de admitir grandes volúmenes de tráfico de lectura.

Con Aurora MySQL, cuando se elimina una réplica de Aurora, su punto de enlace de instancia se quita inmediatamente y la réplica de Aurora se quita del punto de enlace del lector. Si hay instrucciones que se ejecutan en la réplica de Aurora que se van a eliminar, hay un periodo de gracia de tres minutos. Las instrucciones existentes pueden finalizar correctamente durante el periodo de gracia. Cuando termina dicho periodo, se apaga la réplica de Aurora y se elimina.

Important

Las réplicas de Aurora en Aurora MySQL siempre usan el nivel de aislamiento de transacción predeterminado `REPEATABLE READ` para las operaciones en las tablas de InnoDB. Puede usar el comando `SET TRANSACTION ISOLATION LEVEL` para cambiar el nivel de transacción solo para la instancia principal de un clúster de base de datos Aurora MySQL. Esta restricción evita los bloqueos de nivel de usuario en las réplicas de Aurora y permite escalar las réplicas de Aurora para dar cabida a miles de conexiones de usuario activas manteniendo el retardo de las réplicas en un valor mínimo.

Note

Las instrucciones DDL que se ejecutan en la instancia primaria podrían interrumpir las conexiones de la base de datos en las réplicas de Aurora asociadas. Si una conexión de réplica de Aurora está utilizando activamente un objeto de base de datos, como por ejemplo una tabla, y ese objeto se modifica en la instancia primaria con una declaración DDL, se interrumpe la conexión con la réplica de Aurora.

> **Note**
>
> La región China (Ningxia) no es compatible con réplicas de lectura entre regiones.

Opciones de replicación para Amazon Aurora MySQL

Puede configurar la replicación entre cualquiera de las opciones siguientes:

- Dos clústeres de base de datos de Aurora MySQL de diferentes Regiones de AWS mediante la creación de una réplica de lectura entre regiones de un clúster de base de datos de Aurora MySQL.

  Para obtener más información, consulte Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS.
- Dos clústeres de base de datos de Aurora MySQL en la misma Región de AWS mediante la utilización de la reproducción del registro binario (binlog) de MySQL.

  Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).
- Una instancia de base de datos de RDS for MySQL como el origen y un clúster de base de datos de Aurora MySQL mediante la creación de una réplica de lectura de Aurora de una instancia de base de datos de RDS for MySQL.

  Puede utilizar este enfoque para introducir cambios de datos actuales y continuos Aurora MySQL durante la migración a Aurora. Para obtener más información, consulte Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora.

  También puede utilizar este enfoque para aumentar la escalabilidad de las consultas de lectura de sus datos. Para ello, consulta los datos utilizando una o más instancias de base de datos dentro de un clúster Aurora MySQL de solo lectura. Para obtener más información, consulte Escalado de lecturas para su base de datos MySQL con Amazon Aurora.
- Un clúster de base de datos de Aurora MySQL en una Región de AWS y hasta cinco clústeres de base de datos de Aurora de solo lectura de Aurora MySQL en diferentes regiones, mediante la creación de una base de datos global de Aurora.

Puede utilizar una base de datos global Aurora para admitir aplicaciones con presencia mundial. El clúster de base de datos Aurora MySQL principal tiene una instancia de escritor y hasta 15 réplicas Aurora. Los clústeres de base de datos Aurora MySQL secundarios de solo lectura pueden estar formados por hasta 16 réplicas Aurora. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

> **Note**
>
> Al reiniciarse la instancia principal de un clúster de base de datos Amazon Aurora también se reinician automáticamente las réplicas de Aurora de ese clúster de base de datos para restablecer un punto de entrada que garantice la coherencia de lectura/escritura en el clúster de base de datos.

Consideraciones sobre el rendimiento de la replicación de Amazon Aurora MySQL

las siguientes características le ayudan a ajustar el rendimiento de la replicación de Aurora MySQL.

La característica de compresión de registros de réplica reduce automáticamente en ancho de banda de la red para los mensajes de replicación. Dado que cada mensaje se transmite a todas las réplicas de Aurora, los beneficios son mayores para los clústeres de mayor tamaño. Esta característica implica algo de gasto general de CPU en el nodo escritor para realizar la compresión. Siempre está habilitada en la versión 2 y la versión 3 de Aurora MySQL.

La característica de filtrado de binlog reduce automáticamente en ancho de banda de la red para los mensajes de replicación. Puesto que las réplicas de Aurora no usan la información de binlog que se incluye en los mensajes de replicación, esos datos se omiten de los mensajes enviados a esos nodos.

En la versión 2 de Aurora MySQL, puede controlar esta característica cambiando el parámetro `aurora_enable_repl_bin_log_filtering`. Este parámetro está activado de forma predeterminada. Dado que la optimización está pensada para ser transparente, podría desactivar este ajuste solo durante el diagnóstico o la resolución de problemas relacionados con la replicación. Por ejemplo, para hacer coincidir el comportamiento de un clúster de Aurora MySQL más antiguo en el que esta característica no estuviera disponible.

El filtrado de binlog siempre está habilitado en la versión 3 de Aurora MySQL.

Reinicio sin tiempo de inactividad (ZDR) para Amazon Aurora MySQL

La característica de reinicio sin tiempo de inactividad (ZDR) puede conservar algunas o todas las conexiones activas a instancias de base de datos durante ciertos tipos de reinicios. El ZDR se aplica a los reinicios que Aurora realiza de forma automática para resolver las condiciones de error, por ejemplo, cuando una réplica comienza a retrasarse demasiado respecto al origen.

> **Important**
>
> El mecanismo de ZDR funciona sobre la base del mejor esfuerzo. Las versiones de Aurora MySQL, las clases de instancias, las condiciones de error, las operaciones SQL compatibles y otros factores que determinan dónde se aplica el ZDR están sujetos a cambios en cualquier momento.

El ZDR para 2.x de Aurora MySQL requiere la versión 2.10 y posteriores. ZDR está disponible en todas las versiones secundarias de Aurora MySQL 3.x. En Aurora MySQL versión 2 y 3, el mecanismo de ZDR está activado de forma predeterminada y Aurora no utiliza el parámetro `aurora_enable_zdr`.

Aurora informa en la página Events (Eventos) las actividades relacionadas con el reinicio del tiempo de inactividad cero. Aurora registra un evento cuando intenta reiniciar mediante el mecanismo ZDR. En este evento se indica por qué Aurora realiza el reinicio. Luego, Aurora registra otro evento cuando finaliza el reinicio. En este último evento se informa cuánto tiempo tardó el proceso y cuántas conexiones se han conservado o eliminado durante el reinicio. Puede consultar el registro de errores de la base de datos para ver más detalles sobre lo que ocurrió durante el reinicio.

Aunque las conexiones permanecen intactas tras una operación de ZDR correcta, se reinicializan algunas variables y características. Los siguientes tipos de información no se conservan durante un reinicio causado por un reinicio sin tiempo de inactividad:

- Variables globales Aurora restaura las variables de sesión, pero no restaura las variables globales después del reinicio.
- Variables de estado. En particular, se restablece el valor de tiempo de actividad que informa el estado del motor.
- `LAST_INSERT_ID`.

- Estado `auto_increment` en memoria para tablas. El estado de incremento automático en memoria se reinicializa. Para obtener más información sobre los valores de incremento automático, consulte el Manual de referencia de MySQL.
- Información de diagnóstico de las tablas `INFORMATION_SCHEMA` y `PERFORMANCE_SCHEMA`. Esta información de diagnóstico también aparece en la salida de comandos como `SHOW PROFILE` y `SHOW PROFILES`.

En la tabla siguiente se muestran las versiones, los roles de instancia y otras circunstancias que determinan si Aurora puede utilizar el mecanismo de ZDR al reiniciar instancias de base de datos en el clúster.

Aurora MySQL version	¿Se aplica el ZDR al escritor?	¿Se aplica el ZDR a los lectores?	¿El ZDR siempre está activado?	Notas
2.x, inferior a la 2.10.0	No	No	N/A	El ZDR no está disponible para estas versiones .
2.10.0—2.11.0	Sí	Sí	Sí	Aurora revierte cualquier transacción que esté en curso en las conexiones activas. La aplicación debe volver a intentar las transacciones. Aurora cancela cualquier conexión que utilice TLS/SSL, tablas temporales, bloqueos de tablas o bloqueos de usuario.
2.11.1 y versiones posteriores	Sí	Sí	Sí	Aurora revierte cualquier transacción que esté en curso en las conexiones activas. La aplicación debe volver a intentar las transacciones.

Aurora MySQL version	¿Se aplica el ZDR al escritor?	¿Se aplica el ZDR a los lectores?	¿El ZDR siempre está activado?	Notas
				Aurora cancela cualquier conexión que utilice tablas temporales, bloqueos de tablas o bloqueos de usuario.
3.01—3.03	Sí	Sí	Sí	Aurora revierte cualquier transacción que esté en curso en las conexiones activas. La aplicación debe volver a intentar las transacciones. Aurora cancela cualquier conexión que utilice TLS/SSL, tablas temporales, bloqueos de tablas o bloqueos de usuario.
3.04 y versiones posteriores	Sí	Sí	Sí	Aurora revierte cualquier transacción que esté en curso en las conexiones activas. La aplicación debe volver a intentar las transacciones. Aurora cancela cualquier conexión que utilice tablas temporales, bloqueos de tablas o bloqueos de usuario.

Configuración de filtros de replicación con Aurora MySQL

Puede utilizar filtros de replicación para especificar qué bases de datos y tablas se replican con una réplica de lectura. Los filtros de replicación pueden incluir bases de datos y tablas en la replicación o excluirlas de la replicación.

Los siguientes son algunos casos de uso para filtros de replicación:

- Para reducir el tamaño de una réplica de lectura. Con el filtrado de replicación, puede excluir las bases de datos y las tablas que no son necesarias en la réplica de lectura.

- Para excluir bases de datos y tablas de réplicas de lectura por razones de seguridad.
- Para replicar diferentes bases de datos y tablas para casos de uso específicos en diferentes réplicas de lectura. Por ejemplo, puede utilizar réplicas de lectura específicas para análisis o fragmentación.
- Con un clúster de base de datos que tiene réplicas de lectura en diferentes Regiones de AWS, para replicar diferentes bases de datos o tablas en diferentes Regiones de AWS.
- Para especificar qué bases de datos y tablas se replican con un clúster de base de datos de Aurora MySQL que está configurado como una réplica en una topología de replicación entrante. Para obtener más información acerca de esta configuración, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Temas

- Configuración de parámetros de filtrado de replicación para Aurora MySQL
- Limitaciones del filtrado de replicación para Aurora MySQL
- Ejemplos de filtrado de replicación para Aurora MySQL
- Visualización de los filtros de replicación para una réplica de lectura

Configuración de parámetros de filtrado de replicación para Aurora MySQL

Para configurar filtros de replicación, establezca los siguientes parámetros:

- `binlog-do-db`: replicar los cambios en los registros binarios especificados. Cuando se establece este parámetro para un clúster de origen de binlog, solo se replican los registros binarios especificados en el parámetro.
- `binlog-ignore-db`: no replicar los cambios en los registros binarios especificados. Cuando el parámetro `binlog-do-db` se establece para un clúster de origen de binlog, este parámetro no se evalúa.
- `replicate-do-db` – Replicar los cambios en las bases de datos especificadas. Cuando se establece este parámetro para un clúster de réplicas de binlog, solo se replican las bases de datos especificadas en el parámetro.
- `replicate-ignore-db` – No replicar los cambios en las bases de datos especificadas. Cuando el parámetro `replicate-do-db` se establece para un clúster de réplicas de binlog, este parámetro no se evalúa.

- `replicate-do-table` – Replicar los cambios en las tablas especificadas. Cuando se establece este parámetro para una réplica de lectura, solo se replican las tablas especificadas en el parámetro. Además, cuando se establece el parámetro `replicate-do-db` o `replicate-ignore-db`, asegúrese de incluir la base de datos que incluye las tablas especificadas en la replicación con el clúster de réplicas de binlog.
- `replicate-ignore-table` – No replicar los cambios en las tablas especificadas. Cuando el parámetro `replicate-do-table` se establece para un clúster de réplicas de binlog, este parámetro no se evalúa.
- `replicate-wild-do-table` – Replicar tablas en función de la base de datos y los patrones de nombre de tabla especificados. Se admiten los caracteres comodín `%` y `_`. Cuando se establece el parámetro `replicate-do-db` o `replicate-ignore-db`, asegúrese de incluir la base de datos que incluye las tablas especificadas en la replicación con el clúster de réplicas de binlog.
- `replicate-wild-ignore-table` – No replicar tablas en función de la base de datos y los patrones de nombre de tabla especificados. Se admiten los caracteres comodín `%` y `_`. Cuando los parámetros `replicate-do-table` o `replicate-wild-do-table` se establece para un clúster de réplica binlog, este parámetro no se evalúa.

Los parámetros se evalúan en el orden en que se enumeran. Para obtener más información sobre cómo funcionan estos parámetros, consulte la documentación de MySQL.

- Para obtener información general, consulte Opciones y variables del servidor de réplica.
- Para obtener información acerca de cómo se evalúan los parámetros de filtrado de replicación de bases de datos, consulte Evaluación de opciones de registros binarios y replicación a nivel de base de datos.
- Para obtener información acerca de cómo se evalúan los parámetros de filtrado de replicación de tablas, consulte Evaluación de las opciones de replicación a nivel de tabla.

Por defecto, cada uno de estos parámetros tiene un valor vacío. En cada clúster de binlog, puede utilizar estos parámetros para establecer, cambiar y eliminar los filtros de replicación. Cuando establezca uno de estos parámetros, separe cada filtro de los demás con una coma.

Puede utilizar los caracteres comodín `%` y `_` en los parámetros `replicate-wild-do-table` y `replicate-wild-ignore-table`. El comodín `%` coincide con cualquier número de caracteres y el comodín `_` solo coincide con un carácter.

El formato de registro binario de la instancia de base de datos de origen es importante para la replicación, ya que determina el registro de los cambios en los datos. La configuración del parámetro `binlog_format` determina si la replicación está basada en filas o en instrucciones. Para obtener más información, consulte Configuración del registro binario de Aurora MySQL para bases de datos de Single-AZ.

> **Note**
>
> Todas las instrucciones de lenguaje de definición de datos (DDL) se replican como instrucciones, independientemente de la configuración de `binlog_format` en la instancia de base de datos de origen.

Limitaciones del filtrado de replicación para Aurora MySQL

Las siguientes limitaciones se aplican al filtrado de replicación para Aurora MySQL:

- Los filtros de replicación solo son compatibles con Aurora MySQL versión 3.
- Cada parámetro de filtrado de replicación tiene un límite de 2000 caracteres.
- Las comas no son compatibles con los filtros de replicación.
- El filtrado de replicación no es compatible con transacciones XA.

  Para obtener más información, consulte Restricciones a las transacciones XA en la documentación de MySQL.

Ejemplos de filtrado de replicación para Aurora MySQL

Para configurar el filtrado de replicación para una réplica de lectura, modifique los parámetros de filtrado de replicación en el grupo de parámetros del clúster de base de datos asociado a la réplica de lectura.

> **Note**
>
> No puede modificar un grupo de parámetros de clúster de base de datos predeterminado. Si la réplica de lectura emplea un grupo de parámetros predeterminado, cree un nuevo grupo de parámetros y asócielo con la réplica de lectura. Para obtener más información acerca de

los grupos de parámetros de clústeres de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Puede establecer parámetros en un grupo de parámetros de clústeres de base de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS. Para obtener información acerca de cómo configurar los parámetros, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora. Cuando se establecen parámetros en un grupo de parámetros de clústeres de base de datos, todos los clústeres de base de datos asociados al grupo de parámetros utilizan la configuración de los parámetros. Si establece los parámetros de filtrado de replicación en un grupo de parámetros de clústeres de base de datos, asegúrese de que el grupo de parámetros está asociado solo con clústeres de réplicas de lectura. Deje los parámetros de filtrado de replicación vacíos para las instancias de base de datos de origen

En los siguientes ejemplos se establecen los parámetros mediante el uso de AWS CLI. Estos ejemplos establecen `ApplyMethod` en `immediate` para que los cambios de los parámetros se produzcan inmediatamente después de que se complete el comando de la CLI. Si desea que se aplique un cambio pendiente después de reiniciar la réplica de lectura, establezca `ApplyMethod` en `pending-reboot`.

Los siguientes ejemplos establecen filtros de replicación:

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example Inclusión de bases de datos en la replicación

En el ejemplo siguiente se incluyen las bases de datos `mydb1` y `mydb2` en la replicación.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
```

```
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

En:Windows

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Example Inclusión de tablas en la replicación

En el siguiente ejemplo se incluyen las tablas `table1` y `table2` en la base de datos `mydb1` en la replicación.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

En:Windows

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Example Inclusión de tablas en la replicación mediante el uso de caracteres comodín

En el ejemplo siguiente se incluyen tablas con nombres que empiezan con `order` y `return` en la base de datos `mydb` en la replicación.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
```

```
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

En:Windows

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Example Exclusión de bases de datos de la replicación

En el siguiente ejemplo se excluyen las bases de datos `mydb5` y `mydb6` de la replicación.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

En:Windows

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6,ApplyMethod=immediate"
```

Example Exclusión de tablas de la replicación

En el siguiente ejemplo, se excluyen de la replicación las tablas `table1` en la base de datos `mydb5` y `table2` en la base de datos `mydb6`.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

En:Windows

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Example Exclusión de tablas de la replicación mediante el uso de caracteres comodín

En el siguiente ejemplo se excluyen las tablas con nombres que empiezan con `order` y `return` en la base de datos `mydb7` de la replicación.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

En:Windows

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Visualización de los filtros de replicación para una réplica de lectura

Puede ver los filtros de replicación para una réplica de lectura de las siguientes maneras:

- Verifique la configuración de los parámetros de filtrado de replicación en el grupo de parámetros asociado a la réplica de lectura.

  Para obtener instrucciones, consulte Visualización de los valores de parámetros de un grupo de parámetros de base de datos en Amazon Aurora.
- En un cliente de MySQL, conéctese a la réplica de lectura y ejecute la instrucción `SHOW REPLICA STATUS`.

  En la salida, los siguientes campos muestran los filtros de replicación para la réplica de lectura:

- `Binlog_Do_DB`
- `Binlog_Ignore_DB`
- `Replicate_Do_DB`
- `Replicate_Ignore_DB`
- `Replicate_Do_Table`
- `Replicate_Ignore_Table`
- `Replicate_Wild_Do_Table`
- `Replicate_Wild_Ignore_Table`

Para obtener más información acerca de estos campos, consulte Comprobación del estado de replicación en la documentación de MySQL.

Monitoreo de replicación de Amazon Aurora MySQL

El escalado de lectura y la alta disponibilidad dependen de un tiempo de retardo mínimo. Puede monitorizar el retardo de una réplica de Aurora con respecto a la instancia principal del clúster de base de datos de Aurora MySQL mediante la monitorización de la métrica `AuroraReplicaLag` de Amazon CloudWatch. La métrica `AuroraReplicaLag` se registra en cada réplica de Aurora.

La instancia de base de datos principal también registra las métricas `AuroraReplicaLagMaximum`, `AuroraReplicaLagMinimum` y Amazon CloudWatch. La métrica `AuroraReplicaLagMaximum` registra la cantidad máxima de retraso entre la instancia de base de datos principal y cada réplica de Aurora en el clúster de base de datos. La métrica `AuroraReplicaLagMinimum` registra la cantidad mínima de retraso entre la instancia de base de datos principal y cada réplica de Aurora en el clúster de base de datos.

Si necesita el valor más actualizado del retardo de réplica de Aurora, puede comprobar la métrica de `AuroraReplicaLag` en Amazon CloudWatch. El retraso de réplica de Aurora también se registra en cada réplica de Aurora del clúster de base de datos de Aurora MySQL en la tabla `information_schema.replica_host_status`. Para obtener más información sobre esta tabla, consulte information_schema.replica_host_status.

Para obtener más información acerca de la monitorización de instancias de RDS y de las métricas de CloudWatch, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS

Puede crear un clúster de base de datos de Amazon Aurora MySQL como réplica de lectura en una Región de AWS distinta a la del clúster de base de datos de origen. Utilizar este método puede mejorar las capacidades de recuperación de desastres, permitirle escalar las operaciones de lectura en una Región de AWS que esté más cerca de sus usuarios y facilitar la migración de una Región de AWS a otra.

Puede crear réplicas de lectura de clústeres de base de datos cifrados y sin cifrar. La réplica de lectura se debe cifrar si el clúster de base de datos de origen está cifrado.

Por cada clúster de base de datos origen, puede tener hasta cinco clústeres de base de datos réplica de lectura entre regiones.

> **Note**
>
> Como alternativa a las réplicas de lectura entre regiones, puede escalar las operaciones de lectura con un retraso mínimo mediante una base de datos global Aurora. Una base de datos global de Aurora tiene un clúster de base de datos primaria de Aurora en una Región de AWS y hasta 10 clústeres de base de datos secundaria de solo lectura en diferentes regiones. Cada clúster de base de datos secundario puede incluir hasta 16 réplicas Aurora (en lugar de 15). La replicación desde el clúster de base de datos principal a todos los secundarios es manejada por la capa de almacenamiento Aurora en lugar del motor de base de datos, por lo que el tiempo de demora para replicar cambios suele ser mínimo, menos de 1 segundo. Mantener el motor de base de datos fuera del proceso de replicación significa que el motor de base de datos está dedicado al procesamiento de cargas de trabajo. También significa que no necesita configurar o administrar la replicación de binlog (registro binario) de Aurora MySQL. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

Cuando se crea una réplica de lectura del clúster de base de datos de Aurora MySQL en otra Región de AWS, se debe tener en cuenta lo siguiente:

- Tanto el clúster de base de datos origen como el clúster de base de datos réplica de lectura entre regiones pueden tener un máximo de 15 réplicas de Aurora junto con la instancia principal del

clúster de base de datos. Usando esta funcionalidad, puede escalar las operaciones de lectura tanto para la Región de AWS de origen como para la Región de AWS de destino de la replicación.

- En una situación con varias regiones, hay más tiempo de retardo entre el clúster de base de datos de origen y la réplica de lectura porque los canales de red entre Regiones de AWS son más largos.
- Los datos transferidos en las replicaciones entre regiones conllevan cargos por transferencia de datos de Amazon RDS. Las siguientes acciones de replicación entre regiones generan cargos para los datos transferidos fuera de la Región de AWS de origen:
  - Cuando se crea la réplica de lectura, Amazon RDS realiza una instantánea del clúster de origen y transfiere la instantánea a la Región de AWS que contiene la réplica de lectura.
  - Para cada modificación de datos realizada en las bases de datos de origen, Amazon RDS transfiere los datos de la región de origen a la Región de AWS que contiene la réplica de lectura.

  Para obtener más información acerca de los precios de las transferencias de datos de Amazon RDS, consulte Precios de Amazon Aurora.
- Puede ejecutar varias acciones de creación o eliminación simultáneas para réplicas de lectura que hagan referencia al mismo clúster de base de datos origen. Sin embargo, debe permanecer dentro del límite de cinco réplicas de lectura por cada clúster de base de datos de origen.
- Para que la replicación sea eficaz, cada réplica de lectura debe tener la misma cantidad de recursos de computación y de almacenamiento que el clúster de base de datos origen. Si modifica la escala del clúster de base de datos origen, debe ajustar también la escala de las réplicas de lectura.

Temas

- Antes de empezar
- Creación de un clúster de base de datos de réplica de lectura entre regiones para Aurora MySQL
- Promoción de una réplica de lectura a un clúster de base de datos para Aurora MySQL
- Resolución de problemas de réplicas entre regiones para Amazon Aurora MySQL

Antes de empezar

Antes de crear un clúster de base de datos de Aurora MySQL que sea una réplica de lectura entre regiones, debe activar el registro binario en el clúster de base de datos de Aurora MySQL de origen. La replicación entre regiones de Aurora MySQL usa la replicación binaria de MySQL para volver a reproducir los cambios en el clúster de base de datos de la réplica de lectura entre regiones.

Para activar el registro binario en un clúster de base de datos Aurora MySQL, actualice el parámetro `binlog_format` para el clúster de base de datos de origen. El parámetro `binlog_format` es un parámetro de nivel de clúster que se encuentra en el grupo de parámetros de clúster predeterminado. Si su clúster de base de datos usa el grupo de parámetros de clúster de base de datos predeterminado, cree un nuevo grupo de parámetros de clúster de base de datos para modificar la configuración de `binlog_format`. Es recomendable que defina `binlog_format` como `MIXED`. Sin embargo, también puede configurar `binlog_format` como `ROW` o `STATEMENT` si necesita un formato binlog concreto. Reinicie el clúster de base de datos de Aurora para que el cambio entre en vigor.

Para obtener más información sobre la utilización del registro binario con Aurora MySQL, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario). Para obtener más información acerca de cómo modificar los parámetros de configuración de Aurora MySQL, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos y Grupos de parámetros para Amazon Aurora.

Creación de un clúster de base de datos de réplica de lectura entre regiones para Aurora MySQL

Puede crear un clúster de base de datos de Aurora que sea una réplica de lectura entre regiones mediante la Consola de administración de AWS, la AWS Command Line Interface (AWS CLI) o la API de Amazon RDS. Puede crear réplicas de lectura entre regiones desde clústeres de base de datos cifrados y sin cifrar.

Cuando se crea una réplica de lectura entre regiones para Aurora MySQL con la Consola de administración de AWS, Amazon RDS crea un clúster de base de datos en la Región de AWS de destino y, luego, crea automáticamente una instancia de base de datos que es la instancia principal de ese clúster de base de datos.

Cuando se crea una réplica de lectura entre regiones mediante la AWS CLI o la API de RDS, primero se crea el clúster de base de datos en la Región de AWS de destino y se espera a que pase a estar activo. Una vez que está activo, se crea una instancia de base de datos que es la instancia principal de ese clúster de base de datos.

La replicación comienza cuando la instancia principal del clúster de base de datos de la réplica de lectura pasa a estar disponible.

Use los siguientes procedimientos para crear una réplica de lectura entre regiones a partir de un clúster de base de datos de Aurora MySQL. Estos procedimientos sirven para crear réplicas de lectura desde clústeres de base de datos cifrados y sin cifrar.

Consola

Para crear un clúster de base de datos de Aurora MySQL que sea una réplica de lectura entre regiones con la Consola de administración de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la Consola de administración de AWS, seleccione la Región de AWS que aloja el clúster de base de datos de origen.
3. En el panel de navegación, seleccione Databases (Bases de datos).
4. Elija el clúster de base de datos para el que desea crear una réplica de lectura entre regiones.
5. En Actions (Acciones), elija Create cross-Region read replica (Crear réplica de lectura entre regiones).
6. En la página Create cross region read replica (Crear réplica de lectura entre regiones), elija la configuración de opciones para su clúster de base de datos réplica de lectura entre regiones, como se describe en la siguiente tabla.

Opción	Descripción
Destination region (Región de destino)	Elija la Región de AWS que alojará el nuevo clúster de base de datos de réplica de lectura entre regiones.
Destination DB subnet group (Destino del grupo de subredes de base de datos)	Elija el grupo de subredes de base de datos que se debe usar para el clúster de base de datos de réplica de lectura entre regiones.
Accesible públicamente	Elija Yes (Sí) para dar al clúster de base de datos de réplica de lectura entre regiones una dirección IP pública; de lo contrario, seleccione No.
Cifrado	Seleccione Enable Encryption (Habilitar cifrado) para activar el cifrado en reposo para este clúster de base

Opción	Descripción
	de datos. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
AWS KMS key	Solo está disponible si Encryption (Cifrado) se establece en Enable Encryption (Habilitar cifrado). Seleccione la AWS KMS key que se va a utilizar para el cifrado de este clúster de base de datos. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
DB instance class (Clase de instancia de base de datos)	Elija una clase de instancia de base de datos que defina los requisitos de procesamiento y memoria para la instancia principal del clúster de base de datos. Para obtener más información acerca de las opciones de clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora.
Multi-AZ deployment (Implementación Multi-AZ)	Elija Yes (Sí) para crear una réplica de lectura del nuevo clúster de base de datos en otra zona de disponibilidad de la Región de AWS de destino para permitir la conmutación por error. Para obtener más información acerca del uso de varias zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.
Read replica source (Origen de réplica de lectura)	Elija el clúster de base de datos de origen para el que desea crear una réplica de lectura entre regiones.

Opción	Descripción
DB Instance Identifier (Identificador de instancias de bases de datos)	Escriba un nombre para la instancia principal de su clúster de base de datos de réplica de lectura entre regiones. Este identificador se utiliza en la dirección del punto de enlace de la instancia principal del nuevo clúster de base de datos. El identificador de instancias de bases de datos tiene las siguientes limitaciones: • Debe contener de 1 a 63 caracteres alfanuméricos o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser único para todas las instancias de base de datos para cada Cuenta de AWS y por cada Región de AWS. Como el clúster de base de datos de réplica de lectura entre regiones se crea a partir de una instantánea del clúster de base de datos de origen, el nombre de usuario maestro y la contraseña maestra de la réplica de lectura coinciden con el nombre de usuario maestro y la contraseña maestra del clúster de base de datos de origen.

Opción	Descripción
DB clúster identifier (Identificador de clúster de base de datos)	Escriba un nombre para el clúster de base de datos de réplica de lectura entre regiones que sea único para su cuenta en la Región de AWS de destino de la réplica. Este identificador se utiliza en la dirección del punto de enlace del clúster para su clúster de base de datos. Para obtener información acerca del punto de enlace del clúster, consulte Conexiones de puntos de conexión de Amazon Aurora. El identificador del clúster de base de datos tiene las siguientes limitaciones: • Debe contener de 1 a 63 caracteres alfanuméricos o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser único para todos los clústeres de base de datos para cada Cuenta de AWS y por cada Región de AWS.
Priority (Prioridad)	Elija una prioridad de conmutación por error para la primera instancia del nuevo clúster de base de datos. Esta prioridad determina el orden en que se promueven las réplicas de Aurora cuando el sistema se recupera de un error en la instancia principal. Si no selecciona un valor, el ajuste predeterminado es tier-1. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

Opción	Descripción
Database port (Puerto de base de datos)	Especifique el puerto que deben usar las aplicaciones y las utilidades para obtener acceso a la base de datos. Los clústeres de base de datos de Aurora usan de forma predeterminada el puerto 3306 de MySQL. Los firewalls de algunas compañías bloquean las conexiones a este puerto. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el nuevo clúster de base de datos.
Enhanced monitoring (Supervisión mejorada)	Elija Enable enhanced monitoring (Habilitar monitoreo mejorado) para activar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta su clúster de base de datos. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
Monitoring Role (Rol de supervisión)	Solo está disponible si Enhanced Monitoring (Monitorización mejorada) se establece en Enable enhanced monitoring (Habilitar monitorización mejorada). Elija la función de IAM que ha creado para permitir que Amazon RDS se comunique con registros de Amazon Cloudwatch, o bien elija Default (Predeterminado) para que RDS cree un rol denominado `rds-monitoring-role` . Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
Granularity (Grado de detalle)	Solo está disponible si Enhanced Monitoring (Monitorización mejorada) se establece en Enable enhanced monitoring (Habilitar monitorización mejorada). Defina el intervalo, en segundos, en el que se recopilan las métricas para el clúster de base de datos.

Opción	Descripción
Auto minor version upgrade (Actualización automática de versiones secundarias)	Esta configuración no se aplica a los clústeres de base de datos Aurora MySQL. Para obtener más información acerca de las actualizaciones de motor de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

7. Elija Create (Crear) para crear una réplica de lectura entre regiones de Aurora.

AWS CLI

Para crear un clúster de base de datos de Aurora MySQL que sea una réplica de lectura entre regiones con la CLI

1. Llame al comando create-db-clúster de la AWS CLI en la en la que desee crear el clúster de base de datos de la réplica de lectura. Incluya la opción `--replication-source-identifier` y especifique el Nombre de recurso de Amazon (ARN) del clúster de base de datos de origen para el que desea crear una réplica de lectura.

   Para una replicación entre regiones en la que el clúster de base de datos identificado por `--replication-source-identifier` esté cifrado, especifique también las opciones `--kms-key-id` y `--storage-encrypted`.

   > **Note**
   >
   > Puede configurar la replicación entre regiones desde un clúster de base de datos sin cifrar en una réplica de lectura cifrada especificando `--storage-encrypted` y proporcionando un valor para `--kms-key-id`.

   No puede especificar los parámetros `--master-username` y `--master-user-password`. Esos valores se toman del clúster de base de datos de origen.

   El siguiente ejemplo de código crea una réplica de lectura en la región us-east-1 a partir de una instantánea de clúster de base de datos sin cifrar de la región us-west-2. Se llama al comando

en la región us-east-1. En este ejemplo se especifica la opción `--manage-master-user-password` para generar la contraseña del usuario maestro y administrarla en Secrets Manager. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager. También puede utilizar la opción `--master-password` para especificar y administrar la contraseña usted mismo.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

Para Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

El siguiente ejemplo de código crea una réplica de lectura en la región us-east-1 a partir de una instantánea de clúster de base de datos cifrado de la región us-west-2. Se llama al comando en la región us-east-1.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster \
  --kms-key-id my-us-east-1-key \
  --storage-encrypted
```

Para Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster ^
  --kms-key-id my-us-east-1-key ^
  --storage-encrypted
```

La opción `--source-region` es necesaria para la replicación entre regiones entre las regiones AWSGovCloud (Este de EE. UU.) y AWS GovCloud (Oeste EE. UU.), donde el clúster de base de datos identificado por `--replication-source-identifier` está cifrado. Para `--source-region`, especifique la Región de AWS del clúster de base de datos de origen.

Si no se ha especificado `--source-region`, especifique un valor de `--pre-signed-url`. Una URL prefirmada es una URL que contiene una solicitud firmada de Signature Version 4 para el comando `create-db-cluster` que se llama en la Región de AWS de origen. Para obtener más información acerca de la opción `pre-signed-url`, consulte create-db-clúster en la Referencia de los comandos de AWS CLI.

2. Compruebe que el clúster de base de datos está disponible para su uso con el comando describe-db-clústers de la AWS CLI, como se muestra en el siguiente ejemplo.

```
aws rds describe-db-clusters --db-cluster-identifier sample-replica-cluster
```

Cuando los resultados de **`describe-db-clusters`** muestren el estado `available`, cree la instancia principal del clúster de base de datos para que pueda comenzar la replicación. Para ello, utilice el comando create-db-instance de la AWS CLI como se muestra en el siguiente ejemplo.

Para Linux, macOS o Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier sample-replica-cluster \
  --db-instance-class db.r5.large \
  --db-instance-identifier sample-replica-instance \
  --engine aurora-mysql
```

Para Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier sample-replica-cluster ^
  --db-instance-class db.r5.large ^
  --db-instance-identifier sample-replica-instance ^
  --engine aurora-mysql
```

Cuando la instancia de base de datos se ha creado y está disponible, comienza la replicación. Puede determinar si la instancia de base de datos está disponible llamando al comando describe-db-instances de la AWS CLI.

API de RDS

Para crear un clúster de base de datos de Aurora MySQL que sea una réplica de lectura entre regiones con la API

1. Llame a la operación CreateDBclúster de la API de RDS en la Región de AWS en la que desea crear el clúster de base de datos de la réplica de lectura. Incluya el parámetro `ReplicationSourceIdentifier` y especifique el Nombre de recurso de Amazon (ARN) del clúster de base de datos de origen para el que desea crear una réplica de lectura.

   Para una replicación entre regiones en la que el clúster de base de datos identificado por `ReplicationSourceIdentifier` esté cifrado, especifique el parámetro `KmsKeyId` y establecer el parámetro `StorageEncrypted` en `true`.

   > **Note**
   >
   > Puede configurar la replicación entre regiones desde un clúster de base de datos sin cifrar en una réplica de lectura cifrada especificando `StorageEncrypted` como **`true`** y proporcionando un valor para `KmsKeyId`. En este caso, no es necesario especificar `PreSignedUrl`.

   No tiene que incluir los parámetros `MasterUsername` y `MasterUserPassword`, porque esos valores se toman del clúster de base de datos de origen.

El siguiente ejemplo de código crea una réplica de lectura en la región us-east-1 a partir de una instantánea de clúster de base de datos sin cifrar de la región us-west-2. Se llama a la acción en la región us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

El siguiente ejemplo de código crea una réplica de lectura en la región us-east-1 a partir de una instantánea de clúster de base de datos cifrado de la región us-west-2. Se llama a la acción en la región us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &KmsKeyId=my-us-east-1-key
  &StorageEncrypted=true
  &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBCluster
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526ReplicationSourceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster%25253Asample-master-cluster
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-
west-2%252Frds%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
```

```
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-
amz-content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Para la replicación entre regiones entre GovCloud (Este de EE. UU.) y GovCloud (Oeste EE. UU.), donde el clúster de la base de datos identificado por está codificado, especifique también el parámetro. La URL prefirmada debe ser una solicitud válida para la operación de la API `CreateDBCluster` que se pueda realizar en la Región de AWS de origen que contiene el clúster de base de datos cifrado que se va a replicar. El identificador de la clave de KMSS se utiliza para cifrar la réplica de lectura y debe ser una clave de KMS válida para la Región de AWS de destino. Para generar una URL prefirmada de forma automática y no manual, use el comando create-db-clúster de la AWS CLI con la opción `--source-region`.

2. Compruebe que el clúster de base de datos está disponible para el uso con la operación DescribeDBclústers de la API de RDS, como se muestra en el siguiente ejemplo.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=DescribeDBClusters
  &DBClusterIdentifier=sample-replica-cluster
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T002223Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=84c2e4f8fba7c577ac5d820711e34c6e45ffcd35be8a6b7c50f329a74f35f426
```

Cuando los resultados de `DescribeDBClusters` muestren el estado `available`, cree la instancia principal del clúster de base de datos para que pueda comenzar la replicación. Para ello, use la acción CreateDBInstance de la API de RDS, como se muestra en el siguiente ejemplo.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBInstance
  &DBClusterIdentifier=sample-replica-cluster
  &DBInstanceClass=db.r5.large
  &DBInstanceIdentifier=sample-replica-instance
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T003808Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=125fe575959f5bbcebd53f2365f907179757a08b5d7a16a378dfa59387f58cdb
```

Cuando la instancia de base de datos se ha creado y está disponible, comienza la replicación. Puede determinar si la instancia de base de datos está disponible llamando al comando DescribeDBInstances de la AWS CLI.

Visualización de réplicas entre regiones de Amazon Aurora MySQL

Para ver las relaciones de reproducción entre regiones de sus clústeres de base de datos de Amazon Aurora MySQL, llame al comando describe-db-clústers de la AWS CLI o a la operación DescribeDBclústers de la API de RDS. En la respuesta, consulte el campo `ReadReplicaIdentifiers` para los identificadores de clúster de base de datos de cualquier clúster de base de datos de réplica de lectura entre regiones. Consulte el elemento `ReplicationSourceIdentifier` para ver el ARN del clúster de base de datos de origen que es el origen de replicación.

Promoción de una réplica de lectura a un clúster de base de datos para Aurora MySQL

Puede promover una réplica de lectura de Aurora MySQL a un clúster de base de datos independiente. Cuando se promueve una réplica de lectura de Aurora MySQL, las instancias de base de datos se reinician antes de que estén disponibles.

Normalmente, una réplica de lectura de Aurora MySQL se promueve a un clúster de base de datos independiente como un esquema de recuperación de datos si el clúster de base de datos de origen devuelve un error.

Para ello, cree primero una réplica de lectura y, a continuación, monitoree el clúster de base de datos de origen para ver si se producen errores. En caso de error, haga lo siguiente:

1. Promocione la réplica de lectura.
2. Dirija el tráfico de la base de datos al clúster de base de datos promovido.
3. Cree una réplica de lectura de reemplazo que tenga el clúster de base de datos promocionados como origen.

Cuando promociona una réplica de lectura, esta se convierte en un clúster de base de datos de Aurora independiente. Este proceso de promoción puede tardar unos minutos o más, según el tamaño de la réplica de lectura. Una vez que haya promocionado la réplica de lectura a un nuevo clúster de base de datos, este será como cualquier otro clúster de base de datos. Por ejemplo, podrá crear réplicas de lectura a partir de él y realizar operaciones de restauración a un momento dado. También puede crear réplicas de Aurora para el clúster de base de datos.

Como el clúster de base de datos promocionado ya no es una réplica de lectura, no puede usarlo como destino de la replicación.

Los siguientes pasos muestran el proceso general para promocionar una réplica de lectura a un clúster de base de datos:

1. Detenga la escritura de transacciones en el clúster de base de datos de origen de la réplica de lectura y espere hasta que se hayan realizado todas las actualizaciones en la réplica de lectura. Las actualizaciones de la base de datos se producen en la réplica de lectura después de completarse en el clúster de base de datos de origen y el retraso de esta replicación puede variar considerablemente. Utilice la métrica `ReplicaLag` para determinar cuándo se han completado todas las actualizaciones en la réplica de lectura. La métrica `ReplicaLag` registra la cantidad de retraso de una instancia de base de datos de réplica de lectura con respecto a la instancia de base de datos de origen. Cuando la métrica `ReplicaLag` llegue a `0`, la réplica estará funcionando al mismo ritmo que la instancia de base de datos de origen.
2. Promocione la réplica de lectura mediante la opción Promote (Promover) de la consola de Amazon RDS, el comando promote-read-replica-db-clúster de la AWS CLI, o la operación PromoteReadReplicaDBclúster de la API de Amazon RDS.

Elija una instancia de base de datos de Aurora MySQL para promocionar la réplica de lectura. Una vez promocionada la réplica de lectura, el clúster de base de datos de Aurora MySQL se promociona a un clúster de base de datos independiente. La instancia de base de datos con la prioridad más alta se promueve a la instancia de base de datos principal del clúster de base de datos. Las demás instancias de base de datos se convierten en réplicas de Aurora.

> **Note**
>
> El proceso de promoción tarda algunos minutos en completarse. Cuando se promociona una réplica de lectura, la replicación se detiene y las instancias de base de datos se reinician. Una vez completado el reinicio, la réplica de lectura pasa a estar disponible como un nuevo clúster de base de datos.

Consola

Para promocionar una réplica de lectura de Aurora MySQL a un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la consola, elija Instances (Instancias).

   Aparece el panel Instance (Instancia).
3. En el panel Instances (Instancias), seleccione la réplica de lectura que desea promocionar.

   Las réplicas de lectura aparecen como instancias de base de datos de Aurora MySQL.
4. En Actions (Acciones), elija Promote read replica (Promover réplica de lectura).
5. En la página de confirmación, elija Promote Read Replica (Promocionar réplica de lectura).

AWS CLI

Para promover una réplica de lectura a un clúster de base de datos, utilice la operación promote-read-replica-db-clúster de la AWS CLI.

Example

Para Linux, macOS o:Unix

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier mydbcluster
```

En:Windows

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier mydbcluster
```

API de RDS

Para promover una réplica de lectura a un clúster de base de datos, llame a PromoteReadReplicaDBclúster.

Resolución de problemas de réplicas entre regiones para Amazon Aurora MySQL

A continuación, puede encontrar una lista de mensajes de error frecuentes que pueden aparecer al crear una réplica de lectura entre regiones de Amazon Aurora y cómo resolver los errores especificados.

Source clúster [DB clúster ARN] doesn't have binlogs enabled

Para resolver este problema, habilite el registro binario en el clúster de base de datos de origen. Para obtener más información, consulte Antes de empezar.

Source clúster [DB clúster ARN] doesn't have clúster parameter group in sync on writer

Este error aparece si se ha actualizado el parámetro `binlog_format` del clúster de base de datos pero no se ha reiniciado su instancia principal. Reinicie la instancia principal (es decir, la de escritura) del clúster de base de datos y vuelva a intentarlo.

Source clúster [DB clúster ARN] already has a read replica in this region

Puede tener hasta cinco clústeres de base de datos interregionales que sean réplicas de lectura para cada clúster de base de datos de origen en cualquier Región de AWS. Si ya tiene el número máximo de réplicas de lectura para un clúster de base de datos en una Región de AWS concreta, debe eliminar una existente antes de poder crear un nuevo clúster de base de datos entre regiones en dicha región.

El clúster de base de datos [ARN del clúster de base de datos] requiere una actualización del motor de base de datos para admitir la replicación entre regiones

Para resolver este problema, actualice la versión del motor de base de datos para todas las instancias del clúster de base de datos de origen a la versión más reciente del motor de base de datos e intente de nuevo crear una base de datos de réplica de lectura entre regiones.

Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario)

Dado que Amazon Aurora MySQL es compatible con MySQL, puede configurar la replicación entre una base de datos MySQL y un clúster de base de datos Amazon Aurora MySQL. Este tipo de replicación utiliza la replicación de registros binarios de MySQL, también conocida como replicación binlog. Si utiliza la replicación de registro binario con Aurora, le recomendamos que su base de datos MySQL ejecute MySQL versión 5.5 o posterior. Puede configurar la replicación donde el clúster de base de datos de Aurora MySQL sea el origen de replicación o la réplica. Puede replicar con una instancia de base de datos MySQL de Amazon RDS, una base de datos MySQL externa a Amazon RDS u otro clúster de base de datos de Aurora MySQL.

> **Note**
>
> No puede usar la replicación binlog hacia o desde ciertos tipos de clústeres de bases de datos de Aurora. En particular, la replicación de binlog no está disponible para clústeres de Aurora Serverless v1. Si las instrucciones `SHOW MASTER STATUS` y `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL) no devuelven ningún resultado, verifique que el clúster que está utilizando sea compatible con la replicación binlog.

También puede reproducir con una instancia de base de datos de RDS for MySQL o con un clúster de base de datos de Aurora MySQL en otra Región de AWS. Cuando esté realizando una replicación entre Regiones de AWS, asegúrese de que los clústeres y las instancias de base de datos sean accesibles públicamente. Si los clústeres de base de datos de Aurora MySQL están en subredes privadas de la VPC, utilice la interconexión de VPC entre las Regiones de AWS. Para obtener más información, consulte Acceso a un clúster de base de datos en una VPC desde una instancia EC2 de otra VPC.

Si desea configurar una replicación entre un clúster de base de datos de Aurora MySQL y un clúster de base de datos de Aurora MySQL en otra Región de AWS, puede crear un clúster de base de datos de Aurora MySQL como réplica de lectura en una Región de AWS diferente del clúster de base de datos de origen. Para obtener más información, consulte Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS.

Con las versiones 2 y 3 de Aurora MySQL, puede replicar entre Aurora MySQL y un origen o destino externos que utilicen identificadores de transacciones globales (GTID) para la replicación. Asegúrese de que los parámetros relacionados con GTID en el clúster de base de datos de Aurora MySQL disponga de una configuración que sea compatible con el estado del GTID de la base de datos externa. Para obtener información sobre como hacer esto, consulte Uso de la replicación basada en GTID. En Aurora MySQL versión 3.01 y posterior, puede elegir cómo asignar GTID a las transacciones que se replican desde un origen que no utiliza GTID. Para obtener información sobre el procedimiento almacenado que controla ese ajuste, consulte mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versión 3).

> **Warning**
>
> Cuando replique entre Aurora MySQL y MySQL, asegúrese de usar únicamente tablas InnoDB. Si tiene tablas de MyISAM que desea replicar, puede convertirlas a InnoDB antes de configurar la replicación con el siguiente comando.
>
> ```
> alter table <schema>.<table_name> engine=innodb, algorithm=copy;
> ```

En las siguientes secciones, configure la replicación, detenga la replicación, escale las lecturas de su base de datos, optimice la replicación binlog y configure el binlog mejorado.

Temas

- Configuración de la replicación de registros binarios para Aurora MySQL
- Detención de la replicación de registros binarios para Aurora MySQL
- Escalado de lecturas para su base de datos MySQL con Amazon Aurora
- Optimización de la replicación de registros binarios para Aurora MySQL
- Configuración del binlog mejorado para Aurora MySQL

Configuración de la replicación de registros binarios para Aurora MySQL

La configuración de la replicación de MySQL con Aurora MySQL incluye los siguientes pasos, que se describen en detalle:

Contenido

- 1. Activar el registro binario en el origen de replicación
- 2. Retenga los registros binarios en el origen de replicación hasta que dejen de ser necesarios
- 3. Creación de una copia o un volcado del origen de replicación
- 4. Carga del volcado en el destino de la réplica (si fuera necesario)
- 5. Cree un usuario de replicación en su origen de replicación
- 6. Active la replicación en el destino de la réplica
  - Establecimiento de una ubicación para detener la replicación en una réplica de lectura
- 7. Monitoree su réplica
- Sincronización de contraseñas entre el origen de replicación y el destino

1. Activar el registro binario en el origen de replicación

Puede encontrar instrucciones acerca del procedimiento para activar el registro binario en el origen de replicación de su motor de base de datos a continuación.

Motor de base de datos	Instrucciones
Aurora MySQL	Para activar el registro binario en un clúster de base de datos Aurora MySQL Establezca el parámetro del clúster de base de datos `binlog_format` en `ROW`, `STATEMENT` o `MIXED`. Se recomienda usar `MIXED` a menos que se requiera un formato de binlog específico. (El valor predeterminado es `OFF`). Para cambiar el parámetro `binlog_format` , cree un grupo de parámetros de clúster de base de datos personalizado y asócielo a su clúster de base de datos. No puede cambiar los parámetros de un grupo de parámetros de clúster de base de datos predeterminado.

Motor de base de datos	Instrucciones
	Si va a cambiar el parámetro `binlog_format` de `OFF` a otro valor, reinicie el clúster de base de datos de Aurora para que el cambio se aplique. Para obtener más información, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos y Grupos de parámetros para Amazon Aurora.
RDS for MySQL	Para activar el registro binario en una instancia de base de datos de Amazon RDS No puede activar el registro binario directamente para una instancia de base de datos de Amazon RDS, pero puede activarlo por medio de uno de los siguientes procedimientos: • Active las copias de seguridad automatizadas para la instancia de base de datos. Puede activar copias de seguridad automatizadas cuando cree una instancia de base de datos o puede activar copias de seguridad modificando una instancia de base de datos ya existente. Para obtener más información, consulte Creación de una instancia de base de datos en la Guía del usuario de Amazon RDS. • Cree una réplica de lectura para la instancia de base de datos. Para obtener más información, consulte Trabajo con réplicas de lectura en la Guía del usuario de Amazon RDS.

Motor de base de datos	Instrucciones
MySQL (externo)	Para configurar la replicación cifrada Para replicar los datos de forma segura con la versión 2 de Aurora MySQL, puede usar la replicación cifrada. **Note** Si no necesita usar la replicación cifrada, puede omitir estos pasos. A continuación, se indican los requisitos previos para utilizar la replicación cifrada: • La capa de conexión segura (SSL) debe estar habilitada en la base de datos de origen MySQL externa. • Debe disponerse de una clave cliente y un certificado cliente para el clúster de base de datos de Aurora MySQL. Durante la replicación cifrada, el clúster de base de datos Aurora MySQL actúa como un cliente en el servidor de base de datos MySQL. Los certificados y las claves del cliente de Aurora MySQL son archivos con formato .pem. 1. Compruebe que esté preparado para la replicación cifrada: • Si no tiene SSL habilitado en la base de datos de origen de MySQL externa y no dispone de una clave cliente ni de un certificado cliente, active SSL en el servidor de base de datos de MySQL y genere la clave cliente y el certificado cliente necesarios. • Si SSL está habilitado en el origen externo, proporcione una clave cliente y un certificado cliente para el clúster de base de datos de Aurora MySQL. Si no los tiene, genere una nueva clave y certificado para el clúster de base de datos Aurora MySQL. Para firmar el certificado cliente, debe tener la clave de la entidad de certificación que usó para configurar SSL en la base de datos de origen MySQL externa.

Motor de base de datos	Instrucciones

Para obtener más información, consulte Creating SSL Certificates and Keys Using openssl en la documentación de MySQL.

Necesita un certificado de la entidad de certificación, la clave cliente y el certifica do cliente.

2. Conéctese al clúster de base de datos Aurora MySQL como usuario maestro mediante SSL.

   Para obtener información acerca de la conexión a un clúster de base de datos Aurora MySQL con SSL, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

3. Ejecute el procedimiento almacenado `mysql.rds_import_binlog_ssl _material` para importar la información de SSL en el clúster de base de datos Aurora MySQL.

   Para el parámetro `ssl_material_value` , inserte la información de los archivos con formato .pem para el clúster de base de datos Aurora MySQL en la carga JSON correcta.

   En el siguiente ejemplo se importa la información de SSL en un clúster de base de datos Aurora MySQL. En los archivos con formato .pem, el código del cuerpo suele ser más grande que el que se muestra en el ejemplo.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
```

Motor de base de datos	Instrucciones

```
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICA
TE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE
 KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pc
jqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSF
JuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJ
tjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMu
cxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Para obtener más información, consulte mysql.rds_import_binlog_ssl_material y Conexiones TLS a clústeres de base de datos de Aurora MySQL.

> **Note**
>
> Después de ejecutar el procedimiento, los secretos se almacenan en archivos. Para borrar los archivos más tarde, puede ejecutar el procedimiento almacenado mysql.rds_remove_binlog_ssl_material.

Para activar el registro binario en una base de datos MySQL externa

1. En un shell de comando, detenga el servicio mysql.

Motor de base de datos	Instrucciones

```
sudo service mysqld stop
```

2. Edite el archivo `my.cnf` (normalmente se encuentra en `/etc`).

```
sudo vi /etc/my.cnf
```

Añada las opciones `log_bin` y `server_id` a la sección `[mysqld]`. La opción `log_bin` proporciona un identificador de nombre de archivo para los archivos de log binarios. La opción `server_id` proporciona un identificador único para el servidor en las relaciones origen-réplica.

Si no se requiere replicación cifrada, asegúrese de que la base de datos MySQL externa se inicia con los binlogs habilitados y SSL desactivado.

A continuación se indican las entradas pertinentes en el archivo `/etc/my.cnf` para los datos sin cifrar.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1
```

Si se requiere la replicación cifrada, asegúrese de que la base de datos MySQL externa se inicia con SSL y los binlogs habilitados.

Las entradas del archivo `/etc/my.cnf` incluyen las ubicaciones del archivo .pem para el servidor de base de datos MySQL.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1

# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
```

Motor de base de datos	Instrucciones

```
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

Además, la opción `sql_mode` de la instancia de base de datos MySQL debe estar definida como 0 o no se debe incluir en el archivo my.cnf.

Mientras está conectado a la base de datos MySQL externa, registre la posición del log binario de la base de datos MySQL externa.

```
mysql> SHOW MASTER STATUS;
```

El resultado debería ser similar al siguiente:

```
+------------------+----------+--------------+------------------
+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
 Executed_Gtid_Set |
+------------------+----------+--------------+------------------
+-------------------+
| mysql-bin.000031 |      107 |              |                  |
              |
+------------------+----------+--------------+------------------
+-------------------+
1 row in set (0.00 sec)
```

Para obtener más información, consulte Setting the replication source configuration en la documentación de MySQL.

3. Inicie el servicio mysql.

```
sudo service mysqld start
```

2. Retenga los registros binarios en el origen de replicación hasta que dejen de ser necesarios

Cuando se utiliza la replicación de registro binario de MySQL, Amazon RDS no administra el proceso de replicación. Como resultado, debe asegurarse de que los archivos binlog del origen de replicación se retienen hasta que los cambios se hayan aplicado a la réplica. Este mantenimiento le ayuda a restaurar la base de datos de origen si se produce un error.

Consulte las siguientes instrucciones para conservar registros binarios para el motor de base de datos.

Motor de base de datos	Instrucciones
Aurora MySQL	Para conservar registros binarios en un clúster de base de datos Aurora MySQL No tiene acceso a los archivos binlog de un clúster de base de datos de Aurora MySQL. Como resultado, debe elegir un plazo para retener los archivos binlog en el origen de replicación que sea lo suficientemente largo para garantizar que los cambios se han aplicado a la réplica antes de que Amazon RDS elimine el archivo binlog. Puede conservar los archivos binlog en un clúster de base de datos Aurora MySQL durante un máximo de 90 días. Si desea configurar una replicación con una base de datos MySQL o una instancia de base de datos de RDS para MySQL como réplica y la base de datos para la que va a crear una réplica es muy grande, elija un periodo más largo para retener los archivos binlog hasta que la copia inicial de la base de datos en la réplica se haya completado y el retardo de la réplica haya llegado a 0. Para definir el periodo de retención del registro binario, use el procedimiento mysql.rds_set_configuration y especifique un parámetro de configuración de `'binlog retention hours'` junto con el número de horas para retener los archivos binlog en el clúster de base de datos. El valor máximo para las versiones 2.11.0 y superiores y 3 de Aurora MySQL es 2160 (90 días). El ejemplo siguiente establece el periodo de retención de los archivos binlog a 6 días: `CALL mysql.rds_set_configuration('binlog retention hours', 144);`

Motor de base de datos	Instrucciones
	Una vez que haya comenzado la replicación, puede verificar que los cambios se hayan aplicado a la réplica ejecutando el comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL) en la réplica y verificando el campo `Seconds behind master`. Si el campo `Seconds behind master` es 0, entonces no hay retardo de réplica. Cuando no haya retardo de réplica, reduzca el periodo de tiempo durante el que se deben retener los archivos binlog definiendo el parámetro de configuración `binlog retention hours` en un periodo más corto. Si no se especifica esta configuración, el valor predeterminado para Aurora MySQL es 24 (1 día). Si especifica un valor para `'binlog retention hours'` que sea mayor que el máximo, Aurora MySQL utiliza el máximo.
RDS para MySQL	Para conservar los registros binarios en una instancia de base de datos de Amazon RDS Puede retener los archivos de registro binario en una instancia de base de datos de Amazon RDS mediante el establecimiento de las horas de retención de archivos binlog, como en el caso de un clúster de base de datos Aurora MySQL, tal como se ha descrito en la fila anterior. También puede retener los archivos binlog en una instancia de base de datos de Amazon RDS creando una réplica de lectura para la instancia de base de datos. Esta réplica de lectura es temporal y solo se usa para conservar los archivos binlog. Una vez creada la réplica de lectura, llame al procedimiento mysql.rds_stop_replication en la réplica de lectura. Mientras la replicación está detenida, Amazon RDS no elimina ninguno de los archivos binlog del origen de replicación. Una vez que haya configura do la reproducción con la réplica permanente, puede eliminar la réplica de lectura cuando el retardo de la réplica (campo `Seconds behind master`) entre el origen de reproducción y la réplica permanente llegue a 0.

Motor de base de datos	Instrucciones
MySQL (externo)	Para conservar los registros binarios en una base de datos MySQL externa Como los archivos binlog de una base de datos de MySQL externa no se administran con Amazon RDS, se conservan hasta que se eliminan. Una vez que haya comenzado la replicación, puede verificar que los cambios se hayan aplicado a la réplica ejecutando el comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL) en la réplica y verificando el campo `Seconds behind master`. Si el campo `Seconds behind master` es 0, entonces no hay retardo de réplica. Cuando no haya retardo de réplica, podrá eliminar los archivos binlog antiguos.

3. Creación de una copia o un volcado del origen de replicación

Debe usar una instantánea, un clon o un volcado del origen de replicación para cargar una copia de línea de base de los datos en la réplica. A continuación, debe empezar a replicar desde ese punto.

Use las siguientes instrucciones para crear una copia o un volcado del origen de la replicación de su motor de base de datos.

Motor de base de datos	Instrucciones
Aurora MySQL	Para crear una copia de un clúster de base de datos de Aurora MySQL Utilice uno de los siguientes métodos: • Restaure desde una instantánea de clúster de base de datos: 1. Cree un snapshot de clúster de base de datos de su clúster de base de datos de Amazon Aurora. Para obtener más información, consulte Creación de una instantánea de clúster de base de datos. 2. Cree un nuevo clúster de base de datos de Aurora restaurando a partir del snapshot de clúster de base de datos que acaba de crear.

Motor de base de datos	Instrucciones

Asegúrese de conservar el mismo grupo de parámetros de base de datos para el clúster de base de datos restaurado que en el clúster de base de datos original. De este modo, se asegurará de que la copia de su clúster de base de datos tiene el registro binario habilitado. Para obtener más información, consulte Restauración de una instantánea de clúster de base de datos.

- Clone su clúster de base de datos. Para obtener más información, consulte Clonación de un volumen de clúster de base de datos de Amazon Aurora.

Para obtener el nombre y la posición del archivo binlog

Utilice uno de los siguientes métodos:

- En:Consola de administración de AWS
  1. Elija Bases de datos y, a continuación, elija la instancia principal (escritor) del nuevo clúster de base de datos de Aurora para mostrar sus detalles.
  2. Desplácese hasta Recent Events (Eventos recientes). Aparecerá un mensaje de evento que muestra el nombre y la posición del archivo binlog. El mensaje de evento está en el siguiente formato.

     ```
     Binlog position from crash recovery is binlog-file-name  binlog-position
     ```

  3. Guarde los valores de nombre y posición del archivo binlog para cuando comience la replicación.
- Llame al comando describe-events de AWS CLI, como en el siguiente ejemplo.

```
aws rds describe-events

{
    "Events": [
        {
            "EventCategories": [],
            "SourceType": "db-instance",
```

Motor de base de datos	Instrucciones

```
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:
db:sample-restored-instance",
            "Date": "2016-10-28T19:43:46.862Z",
            "Message": "Binlog position from crash recovery is mysql-
bin-changelog.000003 4278",
            "SourceIdentifier": "sample-restored-instance"
        }
    ]
}
```

- Compruebe el registro de error de MySQL de la última posición del archivo binlog de MySQL.

Para crear un volcado de un clúster de base de datos de Aurora MySQL

Si el destino de la réplica es una base de datos de MySQL o una instancia de base de datos de RDS para MySQL, debe crear un archivo de volcado desde un clúster de base de datos de Aurora.

Recuerde ejecutar el comando `mysqldump` en la copia del clúster de base de datos de origen que ha creado. Esto es para evitar tener en cuenta los bloqueos a la hora de realizar el volcado. Si el volcado se realizó directamente en el clúster de base de datos de origen, sería necesario bloquear las tablas de origen para evitar escrituras simultáneas en ellas mientras el volcado está en curso.

1. Conecte el clúster de su base de datos con un cliente MySQL.
2. Ejecute el comando `mysqldump` . Por ejemplo:

```
PROMPT> mysqldump --databases database_name --single-transaction
--order-by-primary -r backup.sql -u local_user s -p
```

3. Una vez creado el archivo de volcado, puede eliminar la copia del clúster de base de datos.

Motor de base de datos	Instrucciones
RDS para MySQL	Para crear una instantánea de una instancia de la base de datos de Amazon RDS Cree una réplica de lectura para la instancia de base de datos de Amazon RDS. Para obtener más información, consulte Creación de una réplica de lectura en la Guía del usuario de Amazon Relational Database Service. 1. Conéctese a la réplica de lectura y detenga la replicación ejecutando el procedimiento mysql.rds_stop_replication. 2. Con la réplica de lectura Detenida, conéctese a la réplica de lectura y ejecute el comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL). Obtenga el nombre del archivo de log binario actual del campo `Relay_Master_Log_File` y la posición del archivo de log del campo `Exec_Master_Log_Pos` . Guarde estos valores para cuando comience la replicación. 3. Con la réplica de lectura en el estado Stopped (Detenido), cree una instantánea de base de datos de la réplica de lectura. Para obtener más información, consulte Creación de una instantánea de base de datos en la Guía del usuario de Amazon Relational Database Service. 4. Elimine la réplica de lectura.

Motor de base de datos	Instrucciones

MySQL (externo)	Para crear un volcado de una base de datos MySQL externa
	1. Antes de crear un volcado, debe asegurarse de que la ubicación del binlog para el volcado está actualizada con los datos de la instancia de origen. Para ello, debe detener primero las operaciones de escritura en la instancia con el siguiente comando:
	`mysql> FLUSH TABLES WITH READ LOCK;`
	2. Cree un volcado de su base de datos de MySQL con el comando `mysqldump` , como se muestra a continuación:
	`PROMPT> sudo mysqldump --databases database_name --master-data=2 --single-transaction \`
	`--order-by-primary -r backup.sql -u local_user -p`
	3. Una vez que haya creado el volcado, desbloquee las tablas de su base de datos MySQL con el siguiente comando:
	`mysql> UNLOCK TABLES;`

4. Carga del volcado en el destino de la réplica (si fuera necesario)

Si va a cargar datos desde un volcado de una base de datos de MySQL que sea externa a Amazon RDS, puede crear una instancia de EC2 para copiar en ella los archivos del volcado. A continuación, puede cargar los datos en el clúster de base de datos o la instancia de base de datos desde esa instancia de EC2. Con este método, puede comprimir los archivos de volcado antes de copiarlos en la instancia de EC2 con el fin de reducir los costos de la red asociados con la copia de datos en Amazon RDS. También puede cifrar el archivo o los archivos de volcado para proteger los datos mientras se transfieren por la red.

> **Note**
>
> Si crea un nuevo clúster de base de datos de Aurora MySQL como destino de la réplica, no necesita cargar un archivo de volcado:
>
> - Puede restaurar desde una instantánea de clúster de base de datos para crear un nuevo clúster de base de datos. Para obtener más información, consulte Restauración de una instantánea de clúster de base de datos.
> - Puede clonar el clúster de base de datos de origen para crear un nuevo clúster de base de datos. Para obtener más información, consulte Clonación de un volumen de clúster de base de datos de Amazon Aurora.
> - Puede migrar los datos de una instantánea de instancia de base de datos a un nuevo clúster de base de datos. Para obtener más información, consulte Migración de datos a un clúster de base de datos de Amazon Aurora MySQL.

Use las siguientes instrucciones para cargar el volcado del origen de replicación en el destino de la réplica para su motor de base de datos.

Motor de base de datos	Instrucciones
Aurora MySQL	Para cargar un volcado en un clúster de base de datos de Aurora MySQL 1. Copie el resultado del comando `mysqldump` del origen de replicación en una ubicación que también pueda conectarse al clúster de base de datos de Aurora MySQL. 2. Conéctese al clúster de base de datos Aurora MySQL mediante el comando `mysql`. A continuación se muestra un ejemplo. `PROMPT> mysql -h host_name -port=3306 -u db_master_user -p` 3. En el símbolo del sistema `mysql`, ejecute el comando `source` y pásele el nombre del archivo de volcado de la base de datos para cargar los datos en el clúster de base de datos Aurora MySQL, por ejemplo:

Motor de base de datos	Instrucciones
	`mysql> source backup.sql;`

RDS para MySQL	Para cargar un volcado en una instancia de base de datos Amazon RDS
	1. Copie el resultado del comando `mysqldump` del origen de replicación en una ubicación que también pueda conectarse a su instancia de base de datos de MySQL.
	2. Conéctese a la instancia de base de datos MySQL usando el comando `mysql`. A continuación se muestra un ejemplo.
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	3. En el símbolo del sistema `mysql`, ejecute el comando `source` y pásele el nombre del archivo de volcado de la base de datos para cargar los datos en la instancia de base de datos MySQL, por ejemplo:
	`mysql> source backup.sql;`

Motor de base de datos	Instrucciones

MySQL (externo)	Para cargar un volcado en una base de datos MySQL externa
	
	No puede cargar una instantánea de base de datos o una instantánea de clúster de base de datos en una base de datos MySQL externa. En su lugar, debe usar la salida del comando `mysqldump` .
	
	1. Copie la salida del comando `mysqldump` del origen de replicación en una ubicación que también pueda conectarse a su base de datos de MySQL.
	2. Conéctese a la base de datos de MySQL usando el comando `mysql`. A continuación se muestra un ejemplo.
	
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	
	3. En el símbolo del sistema `mysql`, ejecute el comando `source` y pásele el nombre del archivo de volcado de la base de datos para cargar los datos en la base de datos MySQL. A continuación se muestra un ejemplo.
	
	`mysql> source backup.sql;`

5. Cree un usuario de replicación en su origen de replicación

Además, cree un ID de usuario en el origen que solo se utilice para la replicación. El siguiente ejemplo es para bases de datos de origen de RDS para MySQL o MySQL externas.

```
mysql> CREATE USER 'repl_user'@'domain_name' IDENTIFIED BY 'password';
```

Para las bases de datos de origen de Aurora MySQL, el parámetro del clúster de base de datos `skip_name_resolve` está establecido en 1 (`ON`) y no se puede modificar, por lo que debe usar una dirección IP para el host en lugar de un nombre de dominio. Para obtener más información, consulte skip_name_resolve en la documentación de MySQL.

```
mysql> CREATE USER 'repl_user'@'IP_address' IDENTIFIED BY 'password';
```

El usuario requiere los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE`. Conceda estos privilegios al usuario.

Si necesita usar la replicación cifrada, exija conexiones SSL al usuario de la replicación. Por ejemplo, puede utilizar una de estas instrucciones para exigir el uso de conexiones SSL en la cuenta de usuario `repl_user`.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'IP_address';
```

```
GRANT USAGE ON *.* TO 'repl_user'@'IP_address' REQUIRE SSL;
```

> **Note**
>
> Si `REQUIRE SSL` no se incluye, la conexión de replicación podría cambiarse inadvertidamente por una conexión sin cifrar.

6. Active la replicación en el destino de la réplica

Antes de activar la replicación, se recomienda que realice una instantánea manual del clúster de base de datos de Aurora MySQL o del destino de la réplica de la instancia de base de datos de RDS for MySQL. Si surge un problema y tiene que restablecer la replicación con el clúster de base de datos o el destino de la réplica de instancia de base de datos, puede restaurar el clúster de base de datos o la instancia de base de datos desde esta instantánea en lugar de tener que volver a importar los datos en el destino de la réplica.

Utilice las siguientes instrucciones para activar la replicación para su motor de base de datos.

Motor de base de datos	Instrucciones
Aurora MySQL	Para activar la replicación desde un clúster de base de datos Aurora MySQL 1. Encuentre el punto de inicio de la replicación. Necesita el nombre del archivo binlog y la posición de binlog. Si el destino de réplica del clúster de base de datos se creó a partir de lo siguiente:

Motor de base de datos	Instrucciones

- Instantánea o clon del clúster de base de datos: recupere el nombre y la posición del archivo binlog a partir de los eventos recientes del clúster de base de datos nuevo, tal y como se muestra en 3. Creación de una copia o un volcado del origen de replicación.
- Instantánea de base de datos: ha recuperado el nombre de archivo y la posición de binlog a partir del comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL) cuando creó la instantánea de su origen de replicación.

2. Conéctese al clúster de base de datos y ejecute los procedimientos siguientes para comenzar la replicación con el origen de la replicación usando el nombre y la ubicación del archivo del registro binario del paso anterior:
   - mysql.rds_set_external_source (Aurora MySQL versión 3)
   - mysql.rds_set_external_master (Aurora MySQL versión 2)
   - mysql.rds_start_replication (todas las versiones)

   El siguiente ejemplo es para Aurora MySQL versión 3.

```
CALL mysql.rds_set_external_source ('mydbinstance.123456789012
.us-east-1.rds.amazonaws.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107,
 0);
CALL mysql.rds_start_replication;
```

Para utilizar el cifrado SSL, establezca el valor final a 1 en vez de `0`.

Motor de base de datos	Instrucciones
RDS para MySQL	Para activar la replicación desde una instancia de base de datos de Amazon RDS 1. Si el destino de la réplica de la instancia de base de datos se creó a partir de una instantánea, necesitará el archivo binlog y la posición del binlog que son el punto de partida para la replicación. Recuperó estos valores del comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL) cuando creó la instantánea de su origen de replicación. 2. Conéctese a la instancia de base de datos y llame a los procedimientos mysql.rds_set_external_master (Aurora MySQL versión 2) o mysql.rds_set_external_source (Aurora MySQL versión 3) y mysql.rds_start_replication para iniciar la replicación con su origen de replicación. Utilice el nombre y la ubicación del archivo del registro binario del paso anterior. A continuación se muestra un ejemplo. `CALL mysql.rds_set_external_master ('mydbcluster.cluster-12345 6789012.us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Para utilizar el cifrado SSL, establezca el valor final a 1 en vez de `0`.

Motor de base de datos	Instrucciones

MySQL (externo)	Para activar la replicación desde una base de datos MySQL externa
	1. Obtenga el archivo binlog y la posición del binlog que son el punto de partida para la replicación. Recuperó estos valores del comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL) cuando creó la instantánea de su origen de replicación. Si el destino de la réplica de MySQL externa se rellenó desde la salida del comando `mysqldump` con la opción `--master-data=2` , el archivo binlog y la posición del binlog se incluirán en la salida. A continuación se muestra un ejemplo.
	`-- -- Position to start replication or point-in-time recovery from -- -- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;`
	2. Conéctese al destino de la réplica de MySQL externa y ejecute los comandos `CHANGE MASTER TO` y `START SLAVE` (versión 2 de Aurora MySQL) o `START REPLICA` (versión 3 de Aurora MySQL) para comenzar la replicación con el origen de replicación usando el nombre y la ubicación del archivo de registro binario del paso anterior. Por ejemplo:
	`CHANGE MASTER TO MASTER_HOST = 'mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com', MASTER_PORT = 3306, MASTER_USER = 'repl_user', MASTER_PASSWORD = 'password', MASTER_LOG_FILE = 'mysql-bin-changelog.000031', MASTER_LOG_POS = 107; -- And one of these statements depending on your engine version: START SLAVE; -- Aurora MySQL version 2 START REPLICA; -- Aurora MySQL version 3`

Si la replicación falla, puede provocar un gran aumento de la E/S no intencionada en la réplica, lo que puede degradar el rendimiento. Si la replicación falla o ya no se necesita, puede ejecutar el procedimiento almacenado mysql.rds_reset_external_master (Aurora MySQL versión 2) o mysql.rds_reset_external_source (Aurora MySQL versión 3) para eliminar la configuración de la replicación.

Establecimiento de una ubicación para detener la replicación en una réplica de lectura

En la versión 3.04 y versiones posteriores de Aurora MySQL, puede utilizar el procedimiento almacenado mysql.rds_start_replication_until (versión 3 de Aurora MySQL) para iniciar la replicación y volver a detenerla en una ubicación concreta del registro binario.

Para iniciar la replicación en una réplica de lectura y detenerla en una ubicación concreta

1. Use un cliente de MySQL para conectarse como usuario maestro a la réplica del clúster de base de datos MySQL.
2. Ejecute el procedimiento almacenado mysql.rds_start_replication_until (versión 3 de Aurora MySQL).

   En el ejemplo siguiente se inicia la replicación y se replican los cambios hasta que alcanza la ubicación `120` del archivo registro binario `mysql-bin-changelog.000777`. En una situación de recuperación de desastres, supongamos que la ubicación `120` es justo anterior al desastre.

   ```
   call mysql.rds_start_replication_until(
     'mysql-bin-changelog.000777',
     120);
   ```

La replicación se detiene automáticamente cuando se alcanza el punto de detención. Se genera el siguiente evento de RDS: `Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure`.

Si usa una replicación basada en GTID, use el procedimiento almacenado mysql.rds_start_replication_until_gtid (Aurora MySQL versión 3) en lugar del procedimiento almacenado mysql.rds_start_replication_until (versión 3 de Aurora MySQL). Para obtener más información sobre la replicación basada en GTID, consulte Uso de la replicación basada en GTID.

7. Monitoree su réplica

Al configurar una replicación de MySQL con un clúster de base de datos Aurora MySQL, debe monitorizar los eventos de conmutación por error para el clúster de base de datos Aurora MySQL cuando se trate del destino de la réplica. Si se produce una conmutación por error, el clúster de base de datos que es el destino de la réplica se podrá volver a crear en un nuevo host con una dirección de red diferente. Para obtener información acerca de la monitorización de los eventos de conmutación por error, consulte Uso de notificaciones de eventos de Amazon RDS.

También puede supervisar el retardo entre el origen de replicación y el destino de la réplica conectándose al destino de la réplica y ejecutando el comando `SHOW SLAVE STATUS` (versión 2 de Aurora MySQL) o `SHOW REPLICA STATUS` (versión 3 de Aurora MySQL). En la salida del comando, el campo `Seconds Behind Master` indica cuánto retardo tiene el destino de la réplica con respecto al origen.

> **Important**
>
> Si actualiza el clúster de base de datos y especifica un grupo de parámetros personalizado, asegúrese de reiniciar manualmente el clúster una vez finalizada la actualización. De este modo, el clúster utilizará la nueva configuración de parámetros personalizada y reiniciará la replicación de binlog.

Sincronización de contraseñas entre el origen de replicación y el destino

Cuando cambia cuentas de usuario y contraseñas en el origen de replicación mediante instrucciones SQL, dichos cambios se replican automáticamente en el destino de replicación.

Si utiliza la Consola de administración de AWS, la AWS CLI o la API de RDS para cambiar la contraseña maestra en el origen de replicación, esos cambios no se replican automáticamente en el destino de replicación. Si desea sincronizar el usuario maestro y la contraseña maestra entre los sistemas de origen y destino, debe realizar el mismo cambio en el destino de replicación usted mismo.

Detención de la replicación de registros binarios para Aurora MySQL

Para detener la replicación del registro binario con una instancia de base de datos MySQL, una base de datos de MySQL externa u otro clúster de base de datos de Aurora, siga estos pasos, que se describen en detalle en este tema.

1. Detenga la replicación del registro binario en el destino de la réplica

2. Desactivar el registro binario en el origen de replicación

1. Detenga la replicación del registro binario en el destino de la réplica

Siga estas instrucciones para detener la replicación del registro binario de su motor de base de datos.

Motor de base de datos	Instrucciones
Aurora MySQL	Para detener la replicación del registro binario en el destino de la réplica de un clúster de base de datos Aurora MySQL Conéctese al clúster de base de datos de Aurora que es el destino de la réplica y llame al procedimiento mysql.rds_stop_replication.
RDS para MySQL	Para detener la replicación de binlog en una instancia de base de datos de Amazon RDS Conéctese a la instancia de base de datos de RDS que es el destino de la réplica y llame al procedimiento mysql.rds_stop_replication.
MySQL (externo)	Para detener la replicación del registro binario en una base de datos MySQL externa Conéctese a la base de datos MySQL y ejecute el comando `STOP SLAVE` (versión 5.7) o `STOP REPLICA` (versión 8.0).

2. Desactivar el registro binario en el origen de replicación

Siga las instrucciones de la tabla siguiente para desactivar el registro binario en el origen de replicación de su motor de base de datos.

Motor de base de datos	Instrucciones
Aurora MySQL	Para desactivar el registro binario en un clúster de base de datos Amazon Aurora 1. Conéctese al clúster de base de datos de Aurora que es el origen de la replicación. 2. Utilice el procedimiento mysql.rds_set_configuration y especifique el parámetro de configuración `binlog retention hours` con el valor `NULL`, tal y como se muestra en el siguiente ejemplo. `CALL mysql.rds_set_configuration('binlog retention hours', NULL);` **Note** No puede utilizar el valor `0` para `binlog retention hours`. 3. Establezca el parámetro `binlog_format` en `OFF` en el origen de replicación. El parámetro `binlog_format` está en el clúster de base de datos personalizado asociado a su clúster de base de datos. Una vez que haya cambiado el valor del parámetro `binlog_format` , reinicie su clúster de base de datos para que el cambio se aplique. Para obtener más información, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos y Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.
RDS for MySQL	Para desactivar el registro binario en una instancia de base de datos de Amazon RDS No puede desactivar el registro binario directamente para una instancia de base de datos de Amazon RDS, pero puede desactivarlo por medio de los siguientes procedimientos: 1. Desactive las copias de seguridad automatizadas para la instancia de base de datos. Puede desactivar copias de seguridad automatizadas modificando una instancia de base de datos ya existente y definiendo el valor de Backup Retention

Motor de base de datos	Instrucciones
	Period (Periodo de retención de copia de seguridad) como 0. Para obtener más información, consulte Modificación de una instancia de base de datos de Amazon RDS y Trabajo con copias de seguridad en la Guía del usuario de Amazon Relational Database Service. 2. Elimine todas las réplicas de lectura para la instancia de base de datos. Para obtener más información, consulte Trabajo con réplicas de lectura de instancias de base de datos MariaDB, MySQL y PostgreSQL en la Guía del usuario de Amazon Relational Database Service.

MySQL (externo)	Para desactivar el registro binario en una base de datos MySQL externa
	Conéctese a la base de datos de MySQL y llame al comando `STOP REPLICATION` .
	1. En un shell de comando, detenga el servicio `mysqld`,
	`sudo service mysqld stop`
	2. Edite el archivo `my.cnf` (normalmente se encuentra en `/etc`).
	`sudo vi /etc/my.cnf`
	Elimine las opciones `log_bin` y `server_id` de la sección `[mysqld]`.
	Para obtener más información, consulte Setting the replication source configuration en la documentación de MySQL.
	3. Inicie el servicio mysql.
	`sudo service mysqld start`

Escalado de lecturas para su base de datos MySQL con Amazon Aurora

Puede usar Amazon Aurora con su instancia de base de datos MySQL para aprovechar las capacidades de escalado de lectura de Amazon Aurora y ampliar la carga de trabajo de lectura para

su instancia de base de datos MySQL. Para usar Aurora para escalar las lecturas de su instancia de base de datos de MySQL, cree un clúster de base de datos de Amazon Aurora MySQL y haga que sea una réplica de lectura de su instancia de base de datos de MySQL. Esto es válido para una instancia de base de datos de RDS for MySQL o una base de datos de MySQL que se ejecute fuera de Amazon RDS.

Para obtener más información acerca de la creación de un clúster de base de datos Amazon Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora.

Cuando configure la replicación entre su instancia de base de datos MySQL y su clúster de base de datos de Amazon Aurora, asegúrese de seguir estas directrices:

- Use la dirección del punto de enlace del clúster de base de datos Amazon Aurora cuando haga referencia a su clúster de base de datos Amazon Aurora MySQL. Si se produce una conmutación por error, la réplica de Aurora que se asciende a instancia principal del clúster de base de datos Aurora MySQL seguirá usando la dirección del punto de enlace del clúster de base de datos.
- Mantenga los binlogs en la instancia de escritor hasta que haya comprobado que se han aplicado a la réplica de Aurora. Este mantenimiento garantiza que puede restaurar la instancia de escritor si se produce un error.

Important

Si usa la replicación autoadministrada, será responsable de monitorizar y resolver los problemas de replicación que se pueden producir. Para obtener más información, consulte Diagnóstico y resolución de retardos entre réplicas de lectura.

Note

Los permisos requeridos para comenzar la replicación en un clúster de base de datos de Aurora MySQL están restringidos y no están disponibles para su usuario maestro de Amazon RDS. Por este motivo, debe usar los procedimientos mysql.rds_set_external_master (Aurora MySQL versión 2), mysql.rds_set_external_source (Aurora MySQL versión 3) y mysql.rds_start_replication para configurar la replicación entre su clúster de base de datos de Aurora MySQL y su instancia de base de datos de MySQL.

Comience la replicación entre una instancia de origen externa y un clúster de base de datos de Aurora MySQL

1. Configure la instancia de base de datos MySQL de origen como de solo lectura:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Ejecute el comando `SHOW MASTER STATUS` en la instancia de base de datos MySQL para determinar la ubicación del binlog. Se recibe un resultado similar al del siguiente ejemplo:

```
File                        Position
-------------------------------------
 mysql-bin-changelog.000031      107
-------------------------------------
```

3. Copie la base de datos la instancia de base de datos MySQL externa en el clúster de base de datos Amazon Aurora MySQL usando `mysqldump`. Para bases de datos muy grandes, es posible que quiera usar el procedimiento de Importación de datos a una base de datos de Amazon RDS para MySQL con tiempo de inactividad reducido en la Guía del usuario de Amazon Relational Database Service.

   Para Linux, macOS o Unix:

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -p local_password | mysql \
        --host aurora_cluster_endpoint_address \
        --port 3306 \
        -u RDS_user_name \
        -p RDS_password
```

   Para Windows:

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
```

```
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -p local_password | mysql ^
        --host aurora_cluster_endpoint_address ^
        --port 3306 ^
        -u RDS_user_name ^
        -p RDS_password
```

> **Note**
>
> Asegúrese de que no haya ningún espacio entre la opción `-p` y la contraseña que haya escrito.

Use las opciones `--host`, `--user (-u)`, `--port` y `-p` del comando `mysql` para especificar el nombre de host, el nombre de usuario, el puerto y la contraseña para conectarse a su clúster de base de datos Aurora. El nombre de host es el nombre DNS del punto de enlace del clúster de base de datos Amazon Aurora, por ejemplo, `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com`. Puede encontrar el valor del punto de enlace en los detalles del clúster en la Management Console de Amazon RDS.

4. Haga que la instancia de base de datos MySQL de origen vuelva a admitir la escritura:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

A fin de obtener más información sobre cómo realizar copias de seguridad para su uso con la reproducción, consulte Backing up a source or replica by making it read only en la documentación de MySQL.

5. En la Management Console de Amazon RDS, añada la dirección IP del servidor que aloja la base de datos MySQL de origen al grupo de seguridad de VPC para el clúster de base de datos Amazon Aurora. Para obtener más información acerca de la modificación de un grupo de seguridad de VPC, consulte Grupos de seguridad de su VPC en la Guía del usuario de Amazon Virtual Private Cloud.

   Es posible que también necesite configurar su red local para permitir las conexiones desde la dirección IP de su clúster de base de datos de Amazon Aurora con el fin de que se pueda

comunicar con la instancia de MySQL de origen. Para encontrar la dirección IP del clúster de base de datos Amazon Aurora, use el comando `host`.

```
host aurora_endpoint_address
```

El nombre de host es el nombre DNS del punto de enlace del clúster de base de datos Amazon Aurora.

6. Utilice el cliente que prefiera para conectarse a la instancia de MySQL externa y cree un usuario de MySQL que se usará para la replicación. Esta cuenta se usa únicamente para la replicación y debe estar limitada a su dominio para mejorar la seguridad. A continuación se muestra un ejemplo.

```
CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
```

7. Para la instancia de MySQL externa, conceda a `REPLICATION CLIENT` y a `REPLICATION SLAVE` privilegios para el usuario de replicación. Por ejemplo, para conceder los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE` en todas las bases de datos al usuario "`repl_user`" del dominio, ejecute el siguiente comando.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com'
    IDENTIFIED BY 'password';
```

8. Tome una instantánea manual del clúster de base de datos de Aurora MySQL para que sea la réplica de lectura antes de configurar la replicación. Si tiene que restablecer la replicación con el clúster de base de datos como réplica de lectura, puede restaurar el clúster de base de datos de Aurora MySQL desde esta instantánea en lugar de tener que importar los datos desde su instancia de base de datos de MySQL en un nuevo clúster de base de datos de Aurora MySQL.

9. Convierta el clúster de base de datos de Amazon Aurora DB en la réplica. Conéctese al clúster de base de datos de Amazon Aurora como usuario maestro e identifique la base de datos origen de MySQL como origen de replicación usando los procedimientos mysql.rds_set_external_master (Aurora MySQL versión 2) o mysql.rds_set_external_source (Aurora MySQL versión 3) y mysql.rds_start_replication.

   Use la posición y el nombre del archivo binlog que determinó en el paso 2. A continuación se muestra un ejemplo.

```
For Aurora MySQL version 2:
CALL mysql.rds_set_external_master ('mymasterserver.example.com', 3306,
```

```
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);

For Aurora MySQL version 3:
CALL mysql.rds_set_external_source ('mymasterserver.example.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);
```

10. En el clúster de base de datos de Amazon Aurora, ejecute el procedimiento mysql.rds_start_replication para comenzar la replicación.

```
CALL mysql.rds_start_replication;
```

Una vez que haya establecido la replicación entre su instancia de base de datos MySQL de origen y su clúster de base de datos Amazon Aurora, podrá añadir réplicas de Aurora a su clúster de base de datos Amazon Aurora. A continuación, puede conectarse a las réplicas de Aurora para escalar la lectura de sus datos. Para obtener más información acerca de la creación de una réplica de Aurora, consulte Adición de réplicas de Aurora a un clúster de base de datos.

Optimización de la replicación de registros binarios para Aurora MySQL

A continuación, puede aprender a optimizar el rendimiento de replicación de registros binarios y solucionar problemas relacionados en Aurora MySQL.

> **Tip**
>
> Esta discusión supone que está familiarizado con el mecanismo de replicación del registro binario de MySQL y cómo funciona. Para obtener información de fondo, consulte Implementación de replicación en la documentación de MySQL.

Replicación de registros binarios de varios subprocesos

Con la replicación de registros binarios de múltiples procesos, un subproceso SQL lee los eventos del registro de retransmisión y los pone en cola para que se apliquen los subprocesos de trabajo de SQL. Los subprocesos de trabajo SQL se administran mediante un subproceso coordinador. Los eventos de registros binarios se aplican en paralelo cuando es posible.

La replicación de registros binarios de varios subprocesos se admite en la versión 3 de Aurora MySQL y la versión 2.12.1 de Aurora MySQL y versiones posteriores.

Cuando se configura una instancia de base de datos de Aurora MySQL para utilizar la replicación de registros binarios, la instancia de réplica utiliza de forma predeterminada la replicación de un solo subproceso para las versiones de Aurora MySQL anteriores a la 3.04. Para habilitar la replicación de múltiples procesos, actualice el parámetro `replica_parallel_workers` con un valor superior a cero en el grupo de parámetros personalizado.

En la versión 3.04 y las versiones posteriores de Aurora MySQL, la replicación es multiproceso de forma predeterminada y `replica_parallel_workers` está establecido en `4`. Puede modificar este parámetro en su grupo de parámetros personalizado.

Las siguientes opciones de configuración le ayudan a ajustar la replicación de múltiples procesos. Para obtener más información, consulte Opciones y variables de replicación y registro binario en el Manual de referencia de MySQL.

La configuración óptima depende de varios factores. Por ejemplo, el rendimiento de la replicación de registros binarios se ve afectado por las características de la carga de trabajo de la base de datos y la clase de instancia de base de datos en la que se ejecuta la réplica. Por lo tanto, le recomendamos que pruebe detenidamente todos los cambios en estos parámetros de configuración antes de aplicar la nueva configuración de parámetros a una instancia de producción.

- `binlog_group_commit_sync_delay`
- `binlog_group_commit_sync_no_delay_count`
- `binlog_transaction_dependency_history_size`
- `binlog_transaction_dependency_tracking`
- `replica_preserve_commit_order`
- `replica_parallel_type`
- `replica_parallel_workers`

En Aurora MySQL versión 3.06 y versiones posteriores, puede mejorar el rendimiento de las réplicas de registros binarios al replicar transacciones para tablas grandes con más de un índice secundario. Esta característica introduce un grupo de subprocesos para aplicar cambios de índice secundarios en paralelo en una réplica de binlog. La característica se controla mediante el parámetro del clúster de base de datos `aurora_binlog_replication_sec_index_parallel_workers`, que controla el número total de subprocesos paralelos disponibles para aplicar los cambios de índice secundarios. El parámetro está establecido en `0` (deshabilitado) de forma predeterminada. Para habilitar esta característica, no es necesario reiniciar la instancia. Para habilitar esta característica,

detenga la replicación en curso, establezca el número deseado de subprocesos de trabajo paralelos y, a continuación, vuelva a iniciar la replicación.

También puede utilizar este parámetro como variable global, donde *n* es el número de subprocesos de trabajo paralelos:

```
SET global aurora_binlog_replication_sec_index_parallel_workers=n;
```

Optimización de la reproducción binlog (Aurora MySQL 2.10 y versiones posteriores)

En Aurora MySQL versión 2.10 y versiones posteriores, Aurora aplica automáticamente una optimización conocida como caché de E/S de binlog a la reproducción de registros binarios. Al almacenar en caché los eventos de binlog confirmados más recientemente, esta optimización se ha diseñado para mejorar el rendimiento del subproceso de copia de datos de binlog y, a la vez, limitar el impacto de las transacciones en primer plano en la instancia de origen de binlog.

> **Note**
>
> La memoria utilizada para esta característica es independiente de la configuración `binlog_cache` de MySQL.
> Esta característica no se aplica a las instancias de base de datos de Aurora que utilizan las clases de instancias `db.t2` y `db.t3`.

No es necesario ajustar ningún parámetro de configuración para activar esta optimización. En particular, si ajusta el parámetro de configuración `aurora_binlog_replication_max_yield_seconds` a un valor distinto de cero en versiones anteriores de Aurora MySQL, vuelva a establecerlo en cero para Aurora MySQL 2.10 o versiones posteriores.

Las variables de estado `aurora_binlog_io_cache_reads` y `aurora_binlog_io_cache_read_requests` se encuentran disponibles en Aurora MySQL 2.10 y versiones posteriores. Estas variables de estado lo ayudan a monitorear la frecuencia con que se leen los datos de la caché de E/S de binlog.

- `aurora_binlog_io_cache_read_requests` muestra el número de solicitudes de lectura de E/S de binlog de la caché.
- `aurora_binlog_io_cache_reads` muestra el número de lecturas de E/S de binlog que recuperan información de la caché.

La siguiente consulta SQL calcula el porcentaje de solicitudes de lectura de binlog que aprovechan la información almacenada en caché. En este caso, cuanto más cerca esté la proporción de 100, mejor será.

```
mysql> SELECT
  (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_reads')
  / (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_read_requests')
  * 100
  as binlog_io_cache_hit_ratio;
+---------------------------+
| binlog_io_cache_hit_ratio |
+---------------------------+
|         99.99847949080622 |
+---------------------------+
```

La característica de caché de E/S de binlog también incluye métricas nuevas relacionadas con los subprocesos de copia de datos de binlog. Los subprocesos de volcado son los subprocesos que se crean cuando se conectan réplicas de binlog nuevas a la instancia de origen de binlog.

Las métricas de subprocesos de copia de datos se imprimen en el registro de la base de datos cada 60 segundos con el prefijo `[Dump thread metrics]`. Las métricas incluyen información para cada réplica de binlog como `Secondary_id`, `Secondary_uuid`, el nombre del archivo de binlog y la posición que lee cada réplica. Las métricas también incluyen `Bytes_behind_primary`, que representan la distancia en bytes entre el origen de la reproducción y la réplica. Esta métrica mide el retraso del subproceso de E/S de réplica. Esta cifra es diferente del retraso del subproceso del aplicador SQL de la réplica, que se representa mediante la métrica `seconds_behind_master` de la réplica de binlog. Puede determinar si las réplicas de binlog alcanzan al origen o se quedan atrás al comprobar si la distancia disminuye o aumenta.

Optimización de la replicación de binlog (Aurora MySQL versión 2 a 2.09)

Para optimizar la replicación de registros binarios para Aurora MySQL, ajuste los siguientes parámetros de optimización a nivel de clúster. Estos parámetros le ayudan a especificar el equilibrio correcto entre la latencia en la instancia de origen binlog y el retraso de replicación.

- `aurora_binlog_use_large_read_buffer`
- `aurora_binlog_read_buffer_size`
- `aurora_binlog_replication_max_yield_seconds`

> **Note**
>
> Para clústeres compatibles con MySQL 5.7, puede usar estos parámetros en las versiones 2 a 2.09.* de Aurora MySQL. En Aurora MySQL 2.10.0 y versiones posteriores, estos parámetros se sustituyen por la optimización de la caché de E/S de binlog y no es necesario usarlos.

Temas

- Descripción general del búfer de lectura grande y optimizaciones de rendimiento máximo
- Parámetros relacionados
- Activación del mecanismo de rendimiento máximo de replicación de registros binarios
- Desactivación de la optimización de rendimiento máximo de replicación de registros binarios
- Desactivación del búfer de lectura grande

Descripción general del búfer de lectura grande y optimizaciones de rendimiento máximo

Es posible que experimente un rendimiento reducido de replicación de registros binarios cuando el subproceso de volcado de registros binarios accede al volumen del clúster Aurora mientras el clúster procesa un gran número de transacciones. Puede utilizar los parámetros `aurora_binlog_use_large_read_buffer`, `aurora_binlog_replication_max_yield_seconds` y `aurora_binlog_read_buffer_size` para ayudar a minimizar este tipo de contención.

Supongamos que tiene una situación en la que `aurora_binlog_replication_max_yield_seconds` está establecido en mayor que 0 y el archivo binlog actual del subproceso de volcado está activo. En este caso, el subproceso de volcado de registro binario espera un número especificado de segundos para llenar el archivo binlog actual con transacciones. Este período de espera evita la contención que puede surgir al replicar cada evento binlog individualmente. Sin embargo, al hacerlo aumenta el retraso de réplica para las réplicas de registros binarios. Esas réplicas pueden quedarse atrás de la fuente en el mismo número de segundos que la configuración `aurora_binlog_replication_max_yield_seconds`.

El archivo binlog actual significa el archivo binlog que el subproceso de volcado está leyendo actualmente para realizar la replicación. Consideramos que un archivo binlog está activo cuando el archivo binlog se está actualizando o abierto para ser actualizado por transacciones entrantes. Después de que Aurora MySQL llena el archivo binlog activo, MySQL crea y cambia a un

nuevo archivo binlog. El antiguo archivo binlog se vuelve inactivo. Ya no se actualiza mediante transacciones entrantes.

> **Note**
>
> Antes de ajustar estos parámetros, mida la latencia y el rendimiento de la transacción a lo largo del tiempo. Es posible que descubra que el rendimiento de la replicación de registros binarios es estable y tiene baja latencia incluso si hay contención ocasional.

`aurora_binlog_use_large_read_buffer`

Si este parámetro se establece en 1, Aurora MySQL optimiza la replicación del registro binario en función de la configuración de los parámetros `aurora_binlog_read_buffer_size` y `aurora_binlog_replication_max_yield_seconds`. Si `aurora_binlog_use_large_read_buffer` es 0, Aurora MySQL ignora los valores de los parámetros `aurora_binlog_read_buffer_size` y `aurora_binlog_replication_max_yield_seconds`.

`aurora_binlog_read_buffer_size`

Los subprocesos de volcado de registros binarios con un búfer de lectura más grande minimizan la cantidad de operaciones de E/S de lectura al leer más eventos para cada E/S. El parámetro `aurora_binlog_read_buffer_size` establece el tamaño del búfer de lectura. El búfer de lectura grande puede reducir la contención de logs binarios para cargas de trabajo que generan una gran cantidad de datos de binlog.

> **Note**
>
> Este parámetro solo tiene un efecto cuando el clúster también tiene la configuración `aurora_binlog_use_large_read_buffer=1`.
> Aumentar el tamaño del búfer de lectura no afecta al rendimiento de la replicación de registros binarios. Los subprocesos de volcado de registro binario no esperan a que las transacciones de actualización llenen el búfer de lectura.

`aurora_binlog_replication_max_yield_seconds`

Si la carga de trabajo requiere una latencia de transacción baja y puede tolerar algún retraso de replicación, puede aumentar el parámetro

`aurora_binlog_replication_max_yield_seconds`. Este parámetro controla la propiedad de rendimiento máximo de la replicación de registros binarios en el clúster.

> **Note**
>
> Este parámetro solo tiene un efecto cuando el clúster también tiene la configuración `aurora_binlog_use_large_read_buffer=1`.

Aurora MySQL reconoce inmediatamente cualquier cambio en el valor del `aurora_binlog_replication_max_yield_seconds` parámetro. No necesita reiniciar la instancia de base de datos. Sin embargo, cuando activa esta configuración, el subproceso de volcado solo comienza a rendir cuando el archivo binlog actual alcanza su tamaño máximo de 128 MB y se gira a un nuevo archivo.

Parámetros relacionados

Utilice los siguientes parámetros de clúster de base de datos para activar la optimización de binlog.

Parámetro	Valor predeterminado	Valores válidos	Descripción
`aurora_binlog_use_large_read_buffer`	1	0, 1	Conmutador para activar la característica de mejora de replicación. Cuando es 1, el subproceso de volcado de registro binario utiliza `aurora_binlog_read_buffer_size` para la replicación del registro binario; de lo contrario, se utiliza el tamaño de búfer predeterminado (8K). No se utiliza en la

Parámetro	Valor predeterminado	Valores válidos	Descripción
			versión 3 de Aurora MySQL.
`aurora_binlog_read_buffer_size`	5242880	8192-536870912	Tamaño del búfer de lectura utilizado por el subproceso de volcado de registro binario cuando el parámetro `aurora_binlog_use_large_read_buffer` se establece en 1. No se utiliza en la versión 3 de Aurora MySQL.
`aurora_binlog_replication_max_yield_seconds`	0	0-36000	El valor máximo aceptado para las versiones 2.07.* de Aurora MySQL es 45. Puede ajustarlo a un valor más alto en las versiones 2.09 y posteriores. Para la versión 2, este parámetro solo funciona cuando el parámetro `aurora_binlog_use_large_read_buffer` está establecido en 1.

Activación del mecanismo de rendimiento máximo de replicación de registros binarios

Puede activar la optimización de rendimiento máximo de replicación de registros binarios de la siguiente manera. Al hacerlo, se minimiza la latencia de las transacciones en la instancia de origen binlog. Sin embargo, es posible que experimente un mayor retraso en la replicación.

Para activar la optimización de binlog de rendimiento máximo para un clúster de Aurora MySQL

1. Cree o edite un grupo de parámetros del clúster de base de datos con la siguiente configuración de parámetros:
   - `aurora_binlog_use_large_read_buffer`: activar con un valor de `ON` o `1`.
   - `aurora_binlog_replication_max_yield_seconds`: especifique un valor mayor que `0`.
2. Asocie el grupo de parámetros de clúster de base de datos con el clúster Aurora MySQL que funciona como origen binlog. Para ello, siga los procedimientos en Grupos de parámetros para Amazon Aurora.
3. Confirme que el cambio de parámetro surte efecto. Para ello, ejecute la siguiente consulta en la instancia de origen binlog.

   ```
   SELECT @@aurora_binlog_use_large_read_buffer,
    @@aurora_binlog_replication_max_yield_seconds;
   ```

   El resultado debería ser similar al siguiente.

   ```
   +----------------------------------------
   +-----------------------------------------------+
   | @@aurora_binlog_use_large_read_buffer |
    @@aurora_binlog_replication_max_yield_seconds |
   +----------------------------------------
   +-----------------------------------------------+
   |                                     1 |
     45 |
   +----------------------------------------
   +-----------------------------------------------+
   ```

Desactivación de la optimización de rendimiento máximo de replicación de registros binarios

Puede desactivar la optimización de rendimiento máximo de replicación de registro binario de la siguiente manera. Al hacerlo, se minimiza el retraso de replicación. Sin embargo, es posible que experimente una latencia mayor para las transacciones en la instancia de origen binlog.

Para desactivar la optimización de rendimiento máximo para un clúster Aurora MySQL

1. Asegúrese de que el grupo de parámetros de clúster de base de datos asociado al clúster de Aurora MySQL disponga de `aurora_binlog_replication_max_yield_seconds` establecido en 0. Para obtener más información sobre el establecimiento de parámetros de configuración con grupos de consultas, consulte Grupos de parámetros para Amazon Aurora.
2. Confirme que el cambio de parámetro surte efecto. Para ello, ejecute la siguiente consulta en la instancia de origen binlog.

   ```
   SELECT @@aurora_binlog_replication_max_yield_seconds;
   ```

   El resultado debería ser similar al siguiente.

   ```
   +-----------------------------------------------+
   | @@aurora_binlog_replication_max_yield_seconds |
   +-----------------------------------------------+
   |                                             0 |
   +-----------------------------------------------+
   ```

Desactivación del búfer de lectura grande

Puede desactivar toda la característica de búfer de lectura grande de la siguiente manera.

Para desactivar el búfer de lectura de registro binario grande para un clúster Aurora MySQL

1. Restablezca `aurora_binlog_use_large_read_buffer` en `OFF` o 0.

   Asegúrese de que el grupo de parámetros de clúster de base de datos asociado al clúster de Aurora MySQL disponga de `aurora_binlog_use_large_read_buffer` establecido en 0. Para obtener más información sobre el establecimiento de parámetros de configuración con grupos de consultas, consulte Grupos de parámetros para Amazon Aurora.
2. En la instancia de origen binlog, ejecute la siguiente consulta.

```
SELECT @@ aurora_binlog_use_large_read_buffer;
```

El resultado debería ser similar al siguiente.

```
+---------------------------------------+
| @@aurora_binlog_use_large_read_buffer |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
```

Configuración del binlog mejorado para Aurora MySQL

El binlog mejorado reduce la sobrecarga de rendimiento de computación provocada por la activación del binlog, que puede alcanzar hasta un 50 % en algunos casos. Con el binlog mejorado, esta sobrecarga se puede reducir a aproximadamente un 13 %. Para reducir la sobrecarga, el binlog mejorado escribe los registros binarios y de transacciones en el almacenamiento en paralelo, lo que minimiza los datos escritos en el momento de la confirmación de la transacción.

El uso del binlog mejorado también mejora el tiempo de recuperación de la base de datos después de los reinicios y las conmutaciones por error hasta en un 99 % en comparación con el binlog comunitario de MySQL. El binlog mejorado es compatible con las cargas de trabajo basadas en binlogs existentes y usted interactúa con él de la misma manera que interactúa con el binlog de MySQL de la comunidad.

El binlog mejorado está disponible en la versión 3.03.1 de Aurora MySQL y versiones posteriores.

Temas

- Configuración de los parámetros del binlog mejorado
- Otros parámetros relacionados
- Diferencias entre el binlog mejorado y el binlog comunitario de MySQL
- Métricas de Amazon CloudWatch para un binlog mejorado
- Limitaciones del binlog mejorado

Configuración de los parámetros del binlog mejorado

Puede cambiar entre el binlog comunitario de MySQL y el binlog mejorado activando o desactivando los parámetros del binlog mejorado. Los usuarios actuales del binlog pueden seguir leyendo y consumiendo los archivos binlog sin lagunas en la secuencia de archivos binlog.

Para activar el binlog mejorado, defina los siguientes parámetros:

Parámetro	Predeterminado/a	Descripción
`binlog_format`	–	Establezca el parámetro `binlog_format` en el formato de registro binario de su elección para activar el registro mejorado. Asegúrese de que `binlog_format parameter` no esté desactivado. Para obtener más información, consulte Configuración del registro binario de Aurora MySQL.
`aurora_enhanced_binlog`	`0`	Establezca el valor de este parámetro en 1 en el grupo de parámetros del clúster de base de datos asociado al clúster de Aurora MySQL. Al cambiar el valor de este parámetro, debe reiniciar la instancia del escritor cuando el valor `DBClusterParameterGroupStatus` aparezca como `pending-reboot` .
`binlog_backup`	`1`	Desactive este parámetro para activar el binlog mejorado.

Parámetro	Predeterminado/a	Descripción
		Para ello, establezca el valor de este parámetro en `0`.
`binlog_replication_globaldb`	`1`	Desactive este parámetro para activar el binlog mejorado. Para ello, establezca el valor de este parámetro en `0`.

> **Important**
>
> Puede desactivar los parámetros `binlog_backup` y `binlog_replication_globaldb` solo si usa el binlog mejorado.

Para desactivar el binlog mejorado, defina los siguientes parámetros:

Parámetro	Descripción
`aurora_enhanced_binlog`	Establezca el valor de este parámetro en `0` en el grupo de parámetros del clúster de base de datos asociado al clúster de Aurora MySQL. Al cambiar el valor de este parámetro, debe reiniciar la instancia del escritor cuando el valor `DBClusterParameterGroupStatus` aparezca como `pending-reboot` .
`binlog_backup`	Desactive este parámetro cuando desactive el binlog mejorado. Para ello, establezca el valor de este parámetro en 1.
`binlog_replication_globaldb`	Desactive este parámetro cuando desactive el binlog mejorado. Para ello, establezca el valor de este parámetro en 1.

Para comprobar si el binlog mejorado está activado, utilice el siguiente comando en el cliente MySQL:

```
mysql>show status like 'aurora_enhanced_binlog';

+-------------------------+--------+
| Variable_name           | Value  |
+-------------------------+--------+
| aurora_enhanced_binlog | ACTIVE |
+-------------------------+--------+
1 row in set (0.00 sec)
```

Cuando el binlog mejorado está activado, el resultado muestra `ACTIVE` para `aurora_enhanced_binlog`.

Otros parámetros relacionados

Al activar el binlog mejorado, se ven afectados los siguientes parámetros:

- El parámetro `max_binlog_size` está visible pero no se puede modificar. El valor predeterminado `134217728` se ajusta automáticamente a `268435456` cuando se activa el binlog mejorado.
- A diferencia del binlog comunitario de MySQL, `binlog_checksum` no actúa como parámetro dinámico cuando el binlog mejorado está activado. Para que el cambio en este parámetro surta efecto, debe reiniciar manualmente el clúster de base de datos incluso cuando `ApplyMethod` esté en `immediate`.
- El valor que establezca en el parámetro `binlog_order_commits` no afecta al orden de las confirmaciones cuando el binlog mejorado está activado. Las confirmaciones siempre se ordenan sin afectar al rendimiento.

Diferencias entre el binlog mejorado y el binlog comunitario de MySQL

El binlog mejorado interactúa de manera diferente con los clones, las copias de seguridad y la base de datos global de Aurora en comparación con el binlog comunitario de MySQL. Recomendamos que entienda las siguientes diferencias antes de usar el binlog mejorado.

- Los archivos binlog mejorados del clúster de base de datos de origen no están disponibles en un clúster de base de datos clonado.

- Los archivos binlog mejorados no se incluyen en las copias de seguridad de Aurora. Por lo tanto, los archivos binlog mejorados del clúster de base de datos de origen no están disponibles después de restaurar un clúster de base de datos, a pesar del período de retención establecido en él.
- Cuando se usan con una base de datos global de Aurora, los archivos binlog mejorados del clúster de base de datos principal no se replican en el clúster de base de datos de las regiones secundarias.

Ejemplos

En los siguientes ejemplos, se muestra la diferencia entre el binlog mejorado y el binlog comunitario de MySQL.

En un clúster de base de datos restaurado o clonado

Cuando el binlog mejorado está activado, los archivos binlog históricos no están disponibles en el clúster de base de datos restaurado o clonado. Tras una operación de restauración o clonación, si el binlog está activado, el nuevo clúster de base de datos comienza a escribir su propia secuencia de archivos binlog, empezando por 1 (mysql-bin-changelog.000001).

Para activar el binlog mejorado después de una operación de restauración o clonación, defina los parámetros del clúster de base de datos requeridos en el clúster de base de datos restaurado o clonado. Para obtener más información, consulte Configuración de los parámetros del binlog mejorado.

Example Ejemplo: operación de clonación o restauración cuando el binlog mejorado está activado

Clúster de base de datos de origen:

```
mysql> show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog turned on
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

En un clúster de base de datos restaurado o clonado, no se realizan copias de seguridad de los archivos binlog cuando se activa el binlog mejorado. Para evitar la discontinuidad en los datos del binlog, tampoco están disponibles los archivos binlog escritos antes de activar el binlog mejorado.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> New sequence of Binlog files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Ejemplo: operación de clonación o restauración cuando el binlog mejorado está desactivado

Clúster de base de datos de origen:

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

El binlog mejorado se desactiva después de `mysql-bin-changelog.000003`. En un clúster de base de datos restaurado o clonado, los archivos binlog escritos están disponibles después de desactivar el binlog mejorado.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

En una base de datos global de Amazon Aurora

En una base de datos global de Amazon Aurora, los datos binlog del clúster de base de datos principal no se replican en los clústeres de base de datos secundarios. Tras un proceso de conmutación por error entre regiones, los datos del binlog no están disponibles en el clúster de base de datos principal promocionado recientemente. Si el binlog está activado, el clúster de base de datos promocionado recientemente comienza a escribir su propia secuencia de archivos binlog, empezando por 1 (mysql-bin-changelog.000001).

Para activar el binlog mejorado después de la conmutación por error, debe configurar los parámetros del clúster de base de datos requeridos en el clúster de base de datos secundario. Para obtener más información, consulte Configuración de los parámetros del binlog mejorado.

Example Ejemplo: la operación de conmutación por error de base de datos global se realiza cuando el binlog mejorado está activado

Clúster de base de datos principal antiguo (antes de la conmutación por error):

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog enabled
```

```
+--------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Nuevo clúster de base de datos principal (después de la conmutación por error):

Los archivos binlog no se replican en regiones secundarias cuando se activa el binlog mejorado. Para evitar la discontinuidad en los datos del binlog, los archivos binlog escritos antes de activar el binlog mejorado no estarán disponibles.

```
mysql>show binary logs;

+--------------------------+-----------+-----------+
| Log_name                 | File_size | Encrypted |
+--------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> Fresh sequence of Binlog
 files
+--------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Ejemplo: la operación de conmutación por error de base de datos global se realiza cuando el binlog mejorado está desactivado

Clúster de base de datos de origen:

```
mysql>show binary logs;

+--------------------------+-----------+-----------+
| Log_name                 | File_size | Encrypted |
+--------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+--------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Clúster de base de datos restaurado o clonado:

El binlog mejorado se desactiva después de `mysql-bin-changelog.000003`. Los archivos binlog que se escriben después de desactivar el binlog mejorado se replican y están disponibles en el clúster de base de datos promocionado recientemente.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
3 rows in set (0.00 sec)
```

Métricas de Amazon CloudWatch para un binlog mejorado

Las siguientes métricas de Amazon CloudWatch solo se publican cuando el binlog mejorado está activado.

Métrica de CloudWatch	Descripción	Unidades
ChangeLogBytesUsed	Es la cantidad de almacenamiento utilizada por el binlog mejorado.	Bytes
ChangeLogReadIOPs	Es el número de operaciones de E/S de lectura realizadas en el binlog mejorado en un intervalo de 5 minutos.	Recuento cada 5 minutos
ChangeLogWriteIOPs	Es el número de operaciones de E/S de escritura en disco realizadas en el binlog mejorado en un intervalo de 5 minutos.	Recuento cada 5 minutos

Limitaciones del binlog mejorado

Las siguientes limitaciones se aplican a los clústeres de base de datos de Amazon Aurora cuando se activa el binlog mejorado.

- El binlog mejorado solo es compatible en la versión 3.03.1 de Aurora MySQL y versiones posteriores.
- Los archivos binlog mejorados escritos en el clúster de base de datos principal no se copian en los clústeres de base de datos clonados o restaurados.
- Cuando se usan con una base de datos global de Amazon Aurora, los archivos binlog mejorados del clúster de base de datos principal no se replican en los clústeres de base de datos secundarios. Por lo tanto, tras el proceso de conmutación por error, los datos históricos del binlog no estarán disponibles en el nuevo clúster de base de datos principal.
- Se ignoran los siguientes parámetros de configuración del binlog:
  - `binlog_group_commit_sync_delay`
  - `binlog_group_commit_sync_no_delay_count`
  - `binlog_max_flush_queue_time`
- No puede eliminar ni cambiar el nombre de una tabla dañada de una base de datos. Para eliminar estas tablas, puede ponerse en contacto con Soporte.
- La caché de E/S del binlog se deshabilita cuando se activa el binlog mejorado. Para obtener más información, consulte Optimización de la replicación de registros binarios para Aurora MySQL.

  > **Note**
  >
  > El binlog mejorado proporciona mejoras en el rendimiento de lectura similares a las de la caché de E/S del binlog y aún más mejoras en el rendimiento de escritura.

- No se admite la característica Backtrack. El binlog mejorado no se puede activar en un clúster de base de datos bajo las siguientes condiciones:
  - Clúster de base de datos con la característica Backtrack activada actualmente.
  - Clúster de base de datos con la característica Backtrack activada previamente, pero ahora deshabilitada.
  - Clúster de base de datos restaurado desde un clúster de base de datos de origen o una instantánea con la característica Backtrack habilitada.

Uso de la replicación basada en GTID

El siguiente contenido explica cómo utilizar los identificadores de transacciones globales (GTID) con replicación de registro binario (binlog) entre un clúster de Aurora MySQL y un origen externo.

> **Note**
>
> Para Aurora, solo puede usar esta característica con clústeres de Aurora MySQL que usan la replicación del binlog en una base de datos de MySQL externa o desde ella. La otra base de datos puede ser una instancia de Amazon RDS MySQL, una base de datos de MySQL en las instalaciones o un clúster de base de datos de Aurora en una Región de AWS diferente. Para obtener información sobre cómo configurar ese tipo de replicación, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Si utiliza la replicación del binlog y no conoce la replicación basada en GTID con MySQL, consulte Replication with global transaction identifiers en la documentación de MySQL.

La replicación basada en GTID no es compatible con las versiones 2 y 3 de Aurora MySQL.

Temas

- Información general de identificadores de transacciones globales (GTID)
- Parámetros de replicación basada en GTID
- Activación de la replicación basada en GTID para un clúster de Aurora MySQL
- Desactivación de la reproducción basada en GTID para un clúster de base de datos de Aurora MySQL

Información general de identificadores de transacciones globales (GTID)

Los identificadores de transacciones globales (GTID) son indentificadores únicos generados por transacciones confirmadas por MySQL. Puede utilizar GTID para que la replicación del binlog sea más simple y sencilla para la solución de problemas.

> **Note**
>
> Cuando Aurora sincroniza datos entre las instancias de base de datos en un clúster, ese mecanismo de replicación no incluye el registro binario (binlog). Para Aurora MySQL, la

replicación basada de GTID solo se aplica cuando también utiliza la replicación del binlog para realizar la replicación dentro o fuera de un clúster de base de datos de Aurora MySQL a partir de una base de datos compatible con MySQL externa.

MySQL usa dos tipos distintos de transacciones para la replicación del binlog:

- Transacciones de GTID: transacciones que se identifican mediante GTID.
- Transacciones anónimas: transacciones que no tienen un GTID asignado.

En una configuración de replicación, los GTID son únicos en todas las instancias de base de datos. Los GTID simplifican la configuración de replicación porque cuando se usan no es necesario hacer referencia a las posiciones de los archivos de registro. Los GTID también simplifican el seguimiento de las transacciones replicadas y determinan si las instancias de origen y las réplicas son coherentes.

Normalmente utiliza la replicación basada en GTID con Aurora cuando efectúa la replicación desde una base de datos compatible con MySQL externa en un clúster de Aurora. Puede configurar esta configuración de replicación como parte de una migración desde una base de datos de Amazon RDS o local en Aurora MySQL. Si la base de datos externa ya utiliza GTID, habilitar la replicación basada en GTID para el clúster de Aurora simplifica el proceso de replicación.

Configure la replicación basada en GTID para un clúster de Aurora MySQL configurando primero los parámetros de configuración relevantes en un grupo de parámetros de clúster de base de datos. A continuación, asocie ese grupo de parámetros al clúster.

Parámetros de replicación basada en GTID

Use los parámetros siguientes para configurar replicación basada en GTID.

Parámetro	Valores válidos	Descripción
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF` especifica que las nuevas transacciones son anónimas (es decir, no tienen GTID) y que una transacción debe ser anónima para replicarse.

Parámetro	Valores válidos	Descripción
		`OFF_PERMISSIVE` especifica que las nuevas transacciones son anónimas, pero que todas las transacciones pueden replicarse. `ON_PERMISSIVE` especifica que las nuevas transacciones son de GTID, pero que todas las transacciones pueden replicarse. `ON` especifica que las nuevas transacciones son de GTID y que una transacción debe ser de GTID para poder replicarse.
`enforce_gtid_consistency`	`OFF`, `ON`, `WARN`	`OFF` permite que las transacciones infrinjan la uniformidad de GTID. `ON` evita que las transacciones infrinjan la uniformidad de GTID. `WARN` permite que las transacciones infrinjan la uniformidad de GTID, pero genera un aviso cuando se proudce una infracción.

> **Note**
>
> En la Consola de administración de AWS, el parámetro `gtid_mode` aparece como `gtid-mode`.

Para la replicación basada en GTID, utilice esta configuración para el grupo de parámetros de clúster de base de datos para el clúster de base de datos de Aurora MySQL:

- `ON` y `ON_PERMISSIVE` se aplican solo a la replicación saliente de una instancia de base de datos de RDS o un clúster de Aurora MySQL. Estos valores provocan que su clúster de base de datos de Aurora utilice GTID para transacciones que se replican en una base de datos externa. `ON` requiere que la base de datos externa también utilice la replicación basada en GTID. `ON_PERMISSIVE` hace que la replicación basada en GTID sea opcional en la base de datos externa.

- Si se establece `OFF_PERMISSIVE`, significa que su instancia de base de datos de Aurora puede aceptar la replicación entrante de una base de datos externa. Se puede aplicar este procedimiento si la base de datos tanto si la base de datos utiliza la replicación basada en GTID como si no.
- Si se establece `OFF`, significa que su clúster de base de datos de Aurora solo acepta la replicación entrante desde bases de datos externas que no utilizan la replicación basada en GTID.

> **Tip**
>
> La replicación entrante es el escenario de replicación del binlog más frecuente para los clústeres de Aurora MySQL. Para la replicación entrante, recomendamos que establezca el modo de GTID en `OFF_PERMISSIVE`. Esta configuración permite la replicación entrante desde bases de datos externas independientemente de la configuración de GTID en el origen de replicación.

Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.

Activación de la replicación basada en GTID para un clúster de Aurora MySQL

Cuando la replicación basada en GTID está habilitada para un clúster de base de datos de Aurora MySQL, la configuración de GTID se aplica a la replicación del binlog entrante y saliente.

Para habilitar la replicación basada en GTID para un clúster de Aurora MySQL, realice el siguiente procedimiento:

1. Cree o edite un grupo de parámetros del clúster de base de datos con la siguiente configuración de parámetros:
   - `gtid_mode` – `ON` o `ON_PERMISSIVE`
   - `enforce_gtid_consistency` – `ON`
2. Asocie el grupo de parámetros del clúster de base de datos con el clúster de Aurora MySQL. Para ello, siga los procedimientos en Grupos de parámetros para Amazon Aurora.
3. (Opcional) Especifique cómo asignar GTID a las transacciones que no los incluyen. Para ello, llame al procedimiento almacenado en mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versión 3).

Desactivación de la reproducción basada en GTID para un clúster de base de datos de Aurora MySQL

Puede desactivar la reproducción basada en GTID para un clúster de base de datos de Aurora MySQL. Si realiza esto, significa que el clúster de Aurora no puede realizar la reproducción del binlog interna o externa con bases de datos externas que utilizan la reproducción basada en GTID.

> **Note**
>
> En el siguiente procedimiento, réplica de lectura hace referencia al destino de replicación en una configuración de Aurora con la replicación del binlog en una base de datos externa o desde ella. No hace referencia a las instancias de base de datos de réplica de Aurora de solo lectura. Por ejemplo, cuando un clúster de Aurora acepta la replicación entrante desde una fuente externa, la instancia principal de Aurora funciona como la réplica de lectura para la replicación del binlog.

Para obtener más información sobre los procedimientos almacenados mencionados en esta sección, consulte Referencia de procedimientos almacenados en Aurora MySQL.

Para desactivar la replicación basada en GTID para un clúster de base de datos de Aurora MySQL

1. En las réplicas de Aurora, ejecute el siguiente procedimiento:

   Para la versión 3

   ```
   CALL mysql.rds_set_source_auto_position(0);
   ```

   Para la versión 2

   ```
   CALL mysql.rds_set_master_auto_position(0);
   ```

2. Restablezca `gtid_mode` en `ON_PERMISSIVE` .

   a. Asegúrese de que el grupo de parámetros de clúster de base de datos asociado al clúster de Aurora MySQL disponga de `gtid_mode` establecido en `ON_PERMISSIVE`.

      Para obtener más información sobre el establecimiento de parámetros de configuración con grupos de consultas, consulte Grupos de parámetros para Amazon Aurora.

b. Reinicie el clúster de base de datos de Aurora MySQL.

3. Restablezca `gtid_mode` en `OFF_PERMISSIVE` .

   a. Asegúrese de que el grupo de parámetros de clúster de base de datos asociado al clúster de Aurora MySQL disponga de `gtid_mode` establecido en `OFF_PERMISSIVE`.

   b. Reinicie el clúster de base de datos de Aurora MySQL.

4. Espere a que todas las transacciones de GTID se hayan aplicado en la instancia principal de Aurora. Para comprobar que se hayan aplicado, realice los siguientes pasos:

   a. En la de la base de datos de MySQL, ejecute el comando `SHOW MASTER STATUS`.

      El resultado debería ser similar al que se indica a continuación.

```
File                        Position
-------------------------------------
mysql-bin-changelog.000031      107
-------------------------------------
```

      Tenga en cuenta el archivo y la posición en su resultado.

   b. En cada réplica de lectura, use la información de archivo y posición de su instancia de origen en el paso anterior para ejecutar la siguiente consulta:

      Para la versión 3

```
SELECT SOURCE_POS_WAIT('file', position);
```

      Para la versión 2

```
SELECT MASTER_POS_WAIT('file', position);
```

      Por ejemplo, si el nombre del archivo es `mysql-bin-changelog.000031` y la posición es `107`, ejecute la siguiente instrucción:

      Para la versión 3

```
SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
```

Para la versión 2

```
SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
```

5. Restablezca los parámetros de GTID para deshabilitar la replicación basada en GTID.

   a. Asegúrese de que el grupo de parámetros del clúster de base de datos asociado al clúster de Aurora MySQL tenga la siguiente configuración de parámetros:

      - `gtid_mode` – `OFF`
      - `enforce_gtid_consistency` – `OFF`

   b. Reinicie el clúster de base de datos de Aurora MySQL.

Uso del reenvío de escritura local en un clúster de base de datos de Amazon Aurora MySQL

El reenvío de escritura local (en el clúster) permite a sus aplicaciones emitir transacciones de lectura/escritura directamente en una réplica de Aurora. A continuación, estas transacciones se reenvían a la instancia de base de datos del escritor para su confirmación. Puede utilizar el reenvío de escritura local cuando sus aplicaciones requieran coherencia de lectura después de escritura, que es la capacidad de leer la última escritura de una transacción.

Las réplicas de lectura reciben actualizaciones del escritor de forma asincrónica. Sin el reenvío de escritura, tiene que realizar transacciones de cualquier lectura que requiera coherencia de lectura después de escritura en la instancia de base de datos del escritor. O bien, tiene que desarrollar una lógica de aplicación personalizada y compleja para aprovechar numerosas réplicas de lectura para la escalabilidad. Sus aplicaciones deben dividir completamente todo el tráfico de lectura y escritura, manteniendo dos conjuntos de conexiones a bases de datos para enviar el tráfico al punto de conexión correcto. Esta sobrecarga de desarrollo complica el diseño de la aplicación cuando las consultas forman parte de una única sesión lógica, o transacción, dentro de la aplicación. Además, dado que el retraso de replicación puede variar entre las réplicas de lectura, es difícil lograr una coherencia de lectura global en todas las instancias de la base de datos.

El reenvío de escritura evita la necesidad de dividir esas transacciones o enviarlas exclusivamente al escritor, lo que simplifica el desarrollo de la aplicación. Esta nueva capacidad facilita la escalabilidad de lectura para las cargas de trabajo que necesitan leer la última escritura de una transacción y no son sensibles a la latencia de escritura.

El reenvío de escritura local es diferente del reenvío de escritura global, que reenvía las escrituras de un clúster de base de datos secundario al clúster de base de datos principal en una base de datos global de Aurora. Puede utilizar el reenvío de escritura local en un clúster de base de datos que forme parte de una base de datos global de Aurora. Para obtener más información, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

El reenvío de escritura local requiere las versiones 3.04 y posteriores de Aurora MySQL.

Temas

- Habilitación del reenvío de escritura local
- Comprobación de si un clúster de base de datos tiene habilitado el reenvío de escritura
- Compatibilidad de las aplicaciones con el reenvío de escritura

- Niveles de aislamiento para el reenvío de escritura
- Coherencia de lectura para el reenvío de escritura
- Ejecutar instrucciones multiparte con reenvío de escritura
- Transacciones con reenvío de escritura
- Parámetros de configuración para el reenvío de escritura
- Métricas de Amazon CloudWatch y variables de estado de Aurora MySQL para el reenvío de escritura
- Identificación de transacciones y consultas reenviadas

Habilitación del reenvío de escritura local

De forma predeterminada, el reenvío de escritura local no está habilitado para los clústeres de bases de datos de Aurora MySQL. El reenvío de escritura local se habilita a nivel de clúster, no a nivel de instancia.

> **Important**
>
> También puede habilitar el reenvío de escritura local para las réplicas de lectura entre regiones que utilizan el registro binario, pero las operaciones de escritura no se reenvían a la Región de AWS de origen. Se reenvían a la instancia de base de datos del escritor del clúster de réplicas de lectura de binlog.
> Utilice este método solo si tiene un caso de uso que requiera escribir en la réplica de lectura de binlog en la Región de AWS secundaria. De no ser así, podría acabar en una situación de "nodos malinformados" en la que los conjuntos de datos replicados no sean coherentes entre sí.
> «Recomendamos utilizar el reenvío de escritura global en las bases de datos globales, en lugar del reenvío de escritura local en las réplicas de lectura entre regiones, a menos que sea absolutamente necesario. Para obtener más información, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

Consola

Si usa la Consola de administración de AWS, seleccione la casilla de verificación Activar el reenvío de escritura loca debajo de Reenvío de escritura de réplicas de lectura al crear o modificar un clúster de base de datos.

AWS CLI

Para habilitar el reenvío de escritura con la AWS CLI, utilice la opción `--enable-local-write-forwarding`. Esta opción funciona cuando crea un nuevo clúster de basede datos mediante el comando `create-db-cluster`. También funciona cuando modifica un clúster de base de datos existente mediante el comando `modify-db-cluster`. Puede deshabilitar el reenvío de escritura mediante la opción `--no-enable-local-write-forwarding` con estos mismos comandos de la CLI.

En el siguiente ejemplo se crea un clúster de base de datos de Aurora MySQL con el reenvío de escritura habilitado.

```
aws rds create-db-cluster \
  --db-cluster-identifier write-forwarding-test-cluster \
  --enable-local-write-forwarding \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.04.0 \
  --master-username myuser \
  --master-user-password mypassword \
  --backup-retention 1
```

A continuación, crea instancias de base de datos del escritor y lector para poder utilizar el reenvío de escritura. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

API de RDS

Para habilitar el reenvío de escritura mediante la API de Amazon RDS, establezca el parámetro `EnableLocalWriteForwarding` en `true`. Este parámetro funciona cuando crea un nuevo clúster de base de datos mediante la operación `CreateDBCluster`. También funciona cuando modifica un clúster de base de datos existente mediante la operación `ModifyDBCluster`. Puede deshabilitar el reenvío de escritura estableciendo el parámetro `EnableLocalWriteForwarding` en `false`.

Habilitación del reenvío de escritura para las sesiones de bases de datos

El parámetro `aurora_replica_read_consistency` es un parámetro de base de datos y un parámetro de clúster de base de datos que permite el reenvío de escritura. Puede especificar `EVENTUAL`, `SESSION` o `GLOBAL` para el nivel de coherencia de lectura. Para obtener más información sobre los niveles de consistencia, consulte Coherencia de lectura para el reenvío de escritura.

Las siguientes reglas se aplican a este parámetro:

- El valor predeterminado es '' (null).
- El reenvío de escritura solo está disponible si establece `aurora_replica_read_consistency` en `EVENTUAL`, `SESSION` o `GLOBAL`. Este parámetro solo es pertinente en instancias del lector de clústeres de bases de datos que tengan habilitado el reenvío de escritura.
- No puede establecer este parámetro (cuando está vacío) o desestablecerlo (cuando ya está configurado) dentro de una transacción de múltiples instrucciones. Puede cambiarlo de un valor válido a otro valor válido durante una transacción de este tipo, pero no recomendamos esta acción.

Comprobación de si un clúster de base de datos tiene habilitado el reenvío de escritura

Para determinar si puede utilizar el reenvío de escritura en un clúster de base de datos, confirme que el clúster tenga el atributo `LocalWriteForwardingStatus` establecido en `enabled`.

En la Consola de administración de AWS, en la pestaña Configuración de la página de detalles del clúster, verá el estado Habilitado para Reenvío de escritura de réplica de lectura local.

Para ver el estado de la configuración de reenvío de escritura de todos los clústeres, ejecute el siguiente comando de la AWS CLI.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
    {
        "LocalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-1"
    },
    {
        "LocalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "requested",
        "DBClusterIdentifier": "test-global-cluster-2"
```

```
    },
    {
        "LocalWriteForwardingStatus": "null",
        "DBClusterIdentifier": "aurora-mysql-v2-cluster"
    }
]
```

Un clúster de base de datos puede tener los siguientes valores para `LocalWriteForwardingStatus`:

- `disabled`: el reenvío de escritura está deshabilitado.
- `disabling`: el reenvío de escritura está en proceso de deshabilitación.
- `enabled`: el reenvío de escritura está habilitado.
- `enabling`: el reenvío de escritura está en proceso de habilitación.
- `null`: el reenvío de escritura no está disponible para este clúster de base de datos.
- `requested`: se ha solicitado el reenvío de escritura, pero aún no está activo.

Compatibilidad de las aplicaciones con el reenvío de escritura

Puede utilizar los siguientes tipos de instrucciones SQL con reenvío de escritura:

- Instrucciones de lenguaje de manipulación de datos (DML) como `INSERT`, `DELETE` y `UPDATE`. Existen algunas restricciones sobre las propiedades de estas instrucciones que puede utilizar con el reenvío de escritura, como se describe a continuación.
- Instrucciones `SELECT ... LOCK IN SHARE MODE` y `SELECT FOR UPDATE`.
- Instrucciones `PREPARE` y `EXECUTE`.

Algunas instrucciones no están permitidas o pueden producir resultados obsoletos cuando se utilizan en un clúster de base de datos con reenvío de escritura. Además, no se admiten funciones ni procedimientos definidos por el usuario. Por lo tanto, la configuración `EnableLocalWriteForwarding` está deshabilitada de forma predeterminada para los clústeres de bases de datos. Antes de la habilitación, asegúrese de que el código de la aplicación no se verá afectado por ninguna de estas restricciones.

Las siguientes restricciones se aplican a las instrucciones SQL que utiliza con el reenvío de escritura. En algunos casos, puede utilizar las instrucciones en clústeres de bases de datos con reenvío de

escritura habilitado. Este enfoque funciona si el reenvío de escritura no está habilitado dentro de la sesión por el parámetro de configuración `aurora_replica_read_consistency`. Si intenta usar una instrucción cuando no está permitida debido al reenvío de escritura, verá un mensaje de error similar al siguiente:

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

Lenguaje de definición de datos (DDL)

Conéctese a la instancia de base de datos del escritor para ejecutar instrucciones DDL. No puede ejecutarlas desde instancias de base de datos del lector.

Actualizar una tabla permanente con datos de una tabla temporal

Puede utilizar tablas temporales en clústeres de base de datos con el reenvío de escritura habilitado. Sin embargo, no puede utilizar una instrucción DML para modificar una tabla permanente si la instrucción hace referencia a una tabla temporal. Por ejemplo, no puede utilizar una instrucción `INSERT ... SELECT` que saque los datos de una tabla temporal.

Transacciones XA

No puede utilizar las siguientes instrucciones en un clúster de base de datos cuando el reenvío de escritura esté activado dentro de la sesión. Puede utilizar estas instrucciones en clústeres de bases de datos que no tengan habilitado el reenvío de escritura o en sesiones en las que la configuración `aurora_replica_read_consistency` esté vacía. Antes de habilitar el reenvío de escritura dentro de una sesión, compruebe si su código utiliza estas instrucciones.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Instrucciones LOAD para tablas permanentes

No puede utilizar las siguientes instrucciones en un clúster de base de datos con reenvío de escritura habilitado.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
```

```
LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Instrucciones de complemento

No puede utilizar las siguientes instrucciones en un clúster de base de datos con reenvío de escritura habilitado.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Declaraciones SAVEPOINT

No puede utilizar las siguientes instrucciones en un clúster de base de datos cuando el reenvío de escritura esté activado dentro de la sesión. Puede utilizar estas instrucciones en clústeres de bases de datos que no tengan habilitado el reenvío de escritura o en sesiones en las que la configuración `aurora_replica_read_consistency` esté en blanco. Antes de habilitar el reenvío de escritura dentro de una sesión, compruebe si su código utiliza estas instrucciones.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Niveles de aislamiento para el reenvío de escritura

En las sesiones que utilizan reenvío de escritura, solo puede utilizar el nivel de aislamiento `REPEATABLE READ`. Aunque también puede utilizar el nivel de aislamiento `READ COMMITTED` con réplicas de Aurora, ese nivel de aislamiento no funciona con el reenvío de escritura. Para obtener información acerca de los niveles de aislamiento de `REPEATABLE READ` y `READ COMMITTED`, consulte Niveles de aislamiento de Aurora MySQL.

Coherencia de lectura para el reenvío de escritura

Puede controlar cuál es el grado de coherencia de lectura en un clúster de base de datos. El nivel de coherencia de lectura determina cuánto espera el clúster base de datos antes de cada operación de lectura para garantizar que algunos de los cambios o todos los cambios se repliquen desde el escritor. Puede ajustar el nivel de coherencia de lectura para asegurarse de que todas las operaciones de escritura reenviadas desde la sesión estén visibles en el clúster de base

de datos antes de cualquier consulta posterior. También puede utilizar esta configuración para asegurarse de que las consultas del clúster de base de datos siempre vean las actualizaciones más recientes del escritor. Esta configuración también se aplica a las consultas enviadas por otras sesiones u otros clústeres. Para especificar este tipo de comportamiento para la aplicación, elija un valor para el parámetro del clúster de base de datos o el parámetro de base de datos `aurora_replica_read_consistency`.

> **Important**
>
> Configure siempre el parámetro del clúster de base de datos o el parámetro de base de datos `aurora_replica_read_consistency` cuando desee utilizar el reenvío de escritura. Si no lo hace, Aurora no reenvía las escrituras. Este parámetro tiene un valor vacío por defecto, por lo que debe elegir un valor específico cuando utilice este parámetro. El parámetro `aurora_replica_read_consistency` solo afecta a los clústeres o instancias de base de datos que tengan habilitado el reenvío de escritura.

A medida que aumenta el nivel de coherencia, la aplicación pasa más tiempo esperando que los cambios se propaguen entre instancias de base de datos. Puede buscar el equilibrio entre un tiempo de respuesta rápido y la garantía de que los cambios realizados en otras instancias de base de datos estén completamente disponibles antes de que se ejecuten las consultas.

Puede especificar los siguientes valores para el parámetro `aurora_replica_read_consistency`:

- `EVENTUAL`: los resultados de las operaciones de escritura de la misma sesión no están visibles hasta que la operación de escritura se realice en la instancia de base de datos del escritor. La consulta no espera a que los resultados actualizados estén disponibles. Por lo tanto, podría recuperar los datos antiguos o los datos actualizados, en función del momento de las instrucciones y la cantidad de retraso de replicación. Se trata de la misma coherencia que en los clústeres de bases de datos de Aurora MySQL que no utilizan el reenvío de escritura.
- `SESSION`: todas las consultas que utilizan el reenvío de escritura ven los resultados de todos los cambios realizados en esa sesión. Los cambios son visibles independientemente de si la transacción está confirmada. Si es necesario, la consulta espera a que se repliquen los resultados de las operaciones de escritura reenviadas.
- `GLOBAL`: una sesión ve todos los cambios confirmados en todas las sesiones e instancias del clúster de base de datos. Cada consulta puede esperar un tiempo, que variará en función de la

cantidad de retardo de la sesión. La consulta continúa cuando el clúster de base de datos está actualizado con todos los datos confirmados del escritor, a partir del momento en que comenzó la consulta.

Para obtener información sobre los parámetros de configuración relacionados con el reenvío de escritura, consulte Parámetros de configuración para el reenvío de escritura.

> **Note**
>
> También puede usar `aurora_replica_read_consistency` como una variable de sesión, por ejemplo:
>
> ```
> mysql> set aurora_replica_read_consistency = 'session';
> ```

Ejemplos de uso del reenvío de escritura

El siguiente ejemplo muestra los efectos del parámetro `aurora_replica_read_consistency` en la ejecución de instrucciones `INSERT` seguidas de instrucciones `SELECT`. Los resultados pueden diferir según el valor de `aurora_replica_read_consistency` y el momento en que se produzcan las instrucciones.

Para lograr una mayor coherencia, puede esperar brevemente antes de emitir la instrucción `SELECT`. O Aurora puede esperar automáticamente hasta que los resultados terminen de replicarse antes de continuar con `SELECT`.

Para obtener información sobre la configuración de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Example con **`aurora_replica_read_consistency`** establecido en **`EVENTUAL`**

Ejecutar una instrucción `INSERT`, seguida inmediatamente de una instrucción `SELECT`, devuelve un valor para `COUNT(*)` con el número de filas antes de insertar la nueva fila. Al ejecutar `SELECT` de nuevo poco tiempo después se devuelve el recuento de filas actualizado. Las instrucciones `SELECT` no esperan.

```
mysql> select count(*) from t1;
+----------+
```

```
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Example con **`aurora_replica_read_consistency`** establecido en **`SESSION`**

Una instrucción `SELECT` inmediatamente después de una instrucción `INSERT` espera hasta que los cambios de la instrucción `INSERT` sean visibles. Las instrucciones `SELECT` posteriores no esperan.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)

mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
```

```
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Con la configuración de coherencia de lectura todavía establecida en `SESSION`, al introducir una breve espera después de realizar una instrucción `INSERT`, el recuento de filas actualizado estará disponible para cuando se ejecute la siguiente instrucción `SELECT`.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Example con **`aurora_replica_read_consistency`** establecido en **`GLOBAL`**

Cada instrucción `SELECT` espera a que todos los cambios de datos que se realicen a partir de la hora de inicio de la instrucción sean visibles antes de realizar la consulta. El tiempo de espera para cada instrucción `SELECT` varía según la cantidad de retraso de replicación.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)

mysql> select count(*) from t1;
+----------+
```

```
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Ejecutar instrucciones multiparte con reenvío de escritura

Una instrucción DML puede constar de varias partes, como una instrucción `INSERT ... SELECT` o una instrucción `DELETE ... WHERE`. En este caso, la instrucción completa se reenvía a la instancia de base de datos del escritor y se ejecuta allí.

Transacciones con reenvío de escritura

Si el modo de acceso de la transacción está configurado en solo lectura, no se utiliza el reenvío de escritura. Puede especificar el modo de acceso de la transacción mediante la instrucción `SET TRANSACTION` o la instrucción `START TRANSACTION`. También puede cambiar el valor de la variable de sesión transaction_read_only para especificar el modo de acceso de la transacción. Solo puede cambiar este valor de sesión cuando esté conectado a un clúster de base de datos que tenga habilitado el reenvío de escritura.

Si una transacción de larga duración no emite ninguna instrucción durante un período de tiempo significativo, podría exceder el período de tiempo de espera de inactividad. Este período tiene un valor predeterminado de un minuto. Puede configurar el parámetro `aurora_fwd_writer_idle_timeout` para aumentarlo hasta un día. La instancia del escritor cancela las transacciones que superen el tiempo de espera de inactividad. La siguiente instrucción que envíe recibirá un error de tiempo de espera. A continuación, Aurora revertirá la transacción.

Este tipo de error puede producirse en otros casos cuando el reenvío de escritura deja de estar disponible. Por ejemplo, Aurora cancela las transacciones que utilicen el reenvío de escritura si reinicia el clúster de base de datos o si deshabilita la configuración de reenvío de escritura.

Cuando se reinicia una instancia del escritor de un clúster que utiliza el reenvío de escritura local, todas las transacciones y consultas activas reenviadas en las instancias del lector que utilizan el reenvío de escritura local se cierran automáticamente. Cuando la instancia del escritor vuelva a estar disponible, podrá volver a intentar estas transacciones.

Parámetros de configuración para el reenvío de escritura

Los grupos de parámetros de base de datos de Aurora contienen ajustes para la característica de reenvío de escritura. Los detalles sobre estos parámetros se resumen en la tabla siguiente, con notas de uso después de la tabla.

Parámetro	Alcance	Tipo	Valor predeterminado	Valores válidos
`aurora_fwd_writer_idle_timeout`	Clúster	Entero sin signo	60	1-86 400
`aurora_fwd_writer_max_connections_pct`	Clúster	Entero largo sin signo	10	0–90
`aurora_replica_read_consistency`	Instancia o clúster	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Para controlar las solicitudes de escritura entrantes, utilice esta configuración:

- `aurora_fwd_writer_idle_timeout`: el número de segundos que la instancia de base de datos del escritor espera a que se produzca actividad en una conexión que se reenvía desde una instancia del lector antes de cerrarla. Si la sesión permanece inactiva al finalizar este período, Aurora la cancela.
- `aurora_fwd_writer_max_connections_pct`: el límite superior en conexiones de base de datos que se puede utilizar en una instancia de base de datos del escritor para gestionar las consultas reenviadas desde las instancias del lector. Se expresa como un porcentaje de la configuración `max_connections` del escritor. Por ejemplo, si `max_connections` es 800 y `aurora_fwd_master_max_connections_pct` o

`aurora_fwd_writer_max_connections_pct` es 10, el escritor permite un máximo de 80 sesiones reenviadas simultáneas. Estas conexiones provienen del mismo grupo de conexiones administrado por la configuración `max_connections`.

Esta configuración solo se aplica al escritor cuando tiene habilitado el reenvío de escritura. Si disminuye el valor, las conexiones existentes no se ven afectadas. Aurora tendrá en cuenta el nuevo valor de la configuración al intentar crear una nueva conexión desde un clúster de base de datos. El valor predeterminado es 10, que representa el 10% del valor `max_connections`.

> **Note**
>
> Como `aurora_fwd_writer_idle_timeout` y `aurora_fwd_writer_max_connections_pct` son parámetros de clúster de base de datos, todas las instancias de base de datos de cada clúster tienen los mismos valores para estos parámetros.

Para obtener más información acerca de `aurora_replica_read_consistency`, consulte Coherencia de lectura para el reenvío de escritura.

Para obtener más información acerca de los grupos de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Métricas de Amazon CloudWatch y variables de estado de Aurora MySQL para el reenvío de escritura

Las siguientes métricas de Amazon CloudWatch y variables de estado de Aurora MySQL se aplican cuando se utiliza el reenvío de escritura para Aurora MySQL. Para obtener más información sobre métricas para las instancias de base de datos de escritor y lector en Aurora MySQL, consulte los siguientes temas.

Temas

- Metrics for write forwarding for Aurora MySQL writer DB instances
- Metrics for write forwarding for Aurora MySQL reader DB instances

Metrics for write forwarding for Aurora MySQL writer DB instances

Las siguientes métricas de Amazon CloudWatch se aplican cuando se utiliza el reenvío de escritura en uno o más clústeres de base de datos. Todas estas métricas se miden en la instancia de base de datos de escritor.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingWriterDMLLatency`	Milisegundos	Tiempo medio para procesar cada instrucción DML reenviada en la instancia de base de datos de escritor. No incluye el tiempo que tarda el clúster de base de datos en reenviar la solicitud de escritura ni el tiempo necesario para replicar los cambios en el escritor.
`ForwardingWriterDMLThroughput`	Recuento por segundo	Número de instrucciones DML reenviadas procesadas cada segundo por esta instancia de base de datos de escritor.
`ForwardingWriterOpenSessions`	Recuento	Número de sesiones reenviadas en la instancia de base de datos de escritor.

Las siguientes variables de estado de Aurora MySQL se aplican cuando se utiliza el reenvío de escritura en uno o más clústeres de base de datos. Todas estas variables de estado se miden en la instancia de base de datos de escritor.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_writer_dml_stmt_count`	Recuento	Número total de instrucciones DML reenviadas a esta instancia de base de datos de escritor.
`Aurora_fwd_writer_dml_stmt_duration`	Microsegundos	Duración total de las instrucciones DML reenviadas a esta instancia de base de datos de escritor.
`Aurora_fwd_writer_open_sessions`	Recuento	Número de sesiones reenviadas en la instancia de base de datos de escritor.
`Aurora_fwd_writer_select_stmt_count`	Recuento	Número total de instancias `SELECT` reenviadas a esta instancia de base de datos de escritor.
`Aurora_fwd_writer_select_stmt_duration`	Microsegundos	Duración total de las instrucciones `SELECT` reenviadas a esta instancia de base de datos de escritor.

Metrics for write forwarding for Aurora MySQL reader DB instances

Las siguientes métricas de CloudWatch se miden en cada instancia de base de datos de lector de un clúster de base de datos con el reenvío de escritura habilitado.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingReplicaDMLLatency`	Milisegundos	Tiempo medio de respuesta de DML reenviadas en la réplica.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingReplicaDMLThroughput`	Recuento por segundo	Número de sentencias DML reenviadas procesadas por segundo.
`ForwardingReplicaOpenSessions`	Recuento	Número de sesiones que utilizan el reenvío de escritura en una instancia de base de datos del lector.
`ForwardingReplicaReadWaitLatency`	Milisegundos	Tiempo de espera medio que una instrucción `SELECT` de una instancia de base de datos del lector espera para ponerse al día con el escritor. El grado en que la instancia de base de datos de lector espera antes de procesar una consulta depende de la configuración `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Recuento por segundo	Número total de `SELECT` instrucciones procesadas cada segundo en todas las sesiones que reenvían escrituras.
`ForwardingReplicaSelectLatency`	Milisegundos	Latencia `SELECT` reenviada, promediada sobre todas las instrucciones `SELECT` reenviadas dentro del período de supervisión.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingReplicaSelectThroughput`	Recuento por segundo	Rendimiento `SELECT` reenviado, media por segundo dentro del período de supervisión.

Las siguientes variables de estado de Aurora MySQL se miden en cada instancia de base de datos de lector de un clúster de base de datos con reenvío de escritura habilitado.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_replica_dml_stmt_count`	Recuento	Número total de instrucciones DML reenviadas desde esta instancia de base de datos de lector.
`Aurora_fwd_replica_dml_stmt_duration`	Microsegundos	Duración total de las instrucciones DML reenviadas desde esta instancia de base de datos de lector.
`Aurora_fwd_replica_errors_session_limit`	Recuento	Número de sesiones rechazadas por el clúster principal debido a una de las siguientes condiciones de error: • writer full • Too many forwarded statements in progress.
`Aurora_fwd_replica_open_sessions`	Recuento	Número de sesiones que utilizan el reenvío de escritura en una instancia de base de datos del lector.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_replica_read_wait_count`	Recuento	Número total de esperas de lectura tras escritura en esta instancia de base de datos del lector.
`Aurora_fwd_replica_read_wait_duration`	Microsegundos	Duración total de las esperas debido a la configuración de coherencia de lectura en esta instancia de base de datos de lector.
`Aurora_fwd_replica_select_stmt_count`	Recuento	Número total de instrucciones `SELECT` reenviadas desde esta instancia de base de datos de lector.
`Aurora_fwd_replica_select_stmt_duration`	Microsegundos	Duración total de las instrucciones `SELECT` reenviadas desde esta instancia de base de datos de lector.

Identificación de transacciones y consultas reenviadas

Puede utilizar la tabla `information_schema.aurora_forwarding_processlist` para identificar las transacciones y consultas reenviadas. Para obtener más información sobre esta tabla, consulte information_schema.aurora_forwarding_processlist.

El siguiente ejemplo muestra todas las conexiones reenviadas en una instancia de base de datos del escritor.

```
mysql> select * from information_schema.AURORA_FORWARDING_PROCESSLIST where
 IS_FORWARDED=1 order by REPLICA_SESSION_ID;
```

```
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+---------------------+----------------+
| ID  | USER     | HOST               | DB       | COMMAND | TIME | STATE        |
 INFO                                     | IS_FORWARDED | REPLICA_SESSION_ID |
 REPLICA_INSTANCE_IDENTIFIER     | REPLICA_CLUSTER_NAME | REPLICA_REGION |
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+--------------------------------------+
| 648 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest58 SET k=k+1 WHERE id=4802579 |            1 |                637 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
| 650 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest54 SET k=k+1 WHERE id=2503953 |            1 |                639 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+---------------------+----------------+
```

En la instancia de base de datos del lector de reenvío, puede ver los subprocesos asociados a estas conexiones de base de datos del escritor si ejecuta `SHOW PROCESSLIST`. Los valores `REPLICA_SESSION_ID` del escritor, 637 y 639, son los mismos que los valores `Id` del lector.

```
mysql> select @@aurora_server_id;

+---------------------------------+
| @@aurora_server_id              |
+---------------------------------+
| my-db-cluster-instance-2        |
+---------------------------------+
1 row in set (0.00 sec)

mysql> show processlist;

+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+
| Id  | User     | Host               | db       | Command | Time | State        | Info
                                        |
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+
| 637 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest12 SET k=k+1 WHERE id=4802579  |
```

```
| 639 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest61 SET k=k+1 WHERE id=2503953  |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
12 rows in set (0.00 sec)
```

Integración de Amazon Aurora MySQL con otros servicios de AWS

Amazon Aurora MySQL se integra con otros servicios de AWS con el fin de permitirle ampliar su clúster de base de datos de Aurora MySQL para usar funcionalidades adicionales en la nube de AWS. El clúster de base de datos de Aurora MySQL puede usar los servicios de AWS para hacer lo siguiente:

- Invoque de forma síncrona o asíncrona una función de AWS Lambda mediante las funciones nativas `lambda_sync` o `lambda_async`. Para obtener más información, consulte Invocación de una función de Lambda desde un clúster de base de datos de Amazon Aurora MySQL.
- Cargar datos desde archivos de texto o XML almacenados en un bucket de Amazon Simple Storage Service (Amazon S3) en el clúster de base de datos usando el comando `LOAD DATA FROM S3` o `LOAD XML FROM S3`. Para obtener más información, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.
- Guardar datos en archivos de texto almacenados en un bucket de Amazon S3 desde su clúster de base de datos usando el comando `SELECT INTO OUTFILE S3`. Para obtener más información, consulte Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3.
- Añada o quite de forma automática réplicas de Aurora con Application Auto Scaling. Para obtener más información, consulte Amazon Aurora Auto Scaling con réplicas de Aurora.
- Realice el análisis de opiniones con Amazon Comprehend o una amplia variedad de algoritmos de machine learning con IA de SageMaker. Para obtener más información, consulte Uso de machine learning de Amazon Aurora.

Aurora protege la capacidad de acceder a otros servicios de AWS por medio de AWS Identity and Access Management (IAM). Puede conceder permiso para acceder a otros servicios de AWS mediante la creación de un rol de IAM con los permisos necesarios y la asociación del rol con el clúster de base de datos. Para obtener detalles e instrucciones acerca del procedimiento para permitir que un clúster de base de datos de Aurora MySQL acceda a otros servicios de AWS en su nombre, consulte Autorización a Amazon Aurora MySQL a acceder a otros servicios de AWS en su nombre.

Autorización a Amazon Aurora MySQL a acceder a otros servicios de AWS en su nombre

Para que un clúster de base de datos de Aurora MySQL pueda acceder a otros servicios en su nombre, cree y configure un rol de AWS Identity and Access Management (IAM). Este rol autoriza a los usuarios de las bases de datos del clúster de base de datos a tener acceso a otros servicios AWS. Para obtener más información, consulte Configuración de roles de IAM para acceder a servicios de AWS.

También debe configurar el clúster de base de datos de Aurora para permitir conexiones salientes hacia el servicio de AWS de destino. Para obtener más información, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

Al hacerlo, los usuarios de base de datos podrán ejecutar las acciones siguientes con otros servicios de AWS:

- Invoque de forma síncrona o asíncrona una función de AWS Lambda mediante las funciones nativas `lambda_sync` o `lambda_async`. O bien invoque de forma asíncrona una función de AWS Lambda con el procedimiento `mysql.lambda_async`. Para obtener más información, consulte Invocación de una función de Lambda con una función nativa de Aurora MySQL.
- Cargar datos desde archivos de texto o XML almacenados en un bucket de Amazon S3 en el clúster de base de datos usando la instrucción `LOAD DATA FROM S3` o `LOAD XML FROM S3`. Para obtener más información, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.
- Guardar datos del clúster de base de datos en archivos de texto almacenados en un bucket de Amazon S3 usando el comando `SELECT INTO OUTFILE S3`. Para obtener más información, consulte Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3.
- Exportar datos de log a Amazon CloudWatch Logs MySQL. Para obtener más información, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.
- Añada o quite de forma automática réplicas de Aurora con Application Auto Scaling. Para obtener más información, consulte Amazon Aurora Auto Scaling con réplicas de Aurora.

Configuración de roles de IAM para acceder a servicios de AWS

Para permitir el acceso de un clúster de base de datos de Aurora a otro servicio de AWS, haga lo siguiente:

1. Cree una política de IAM que otorgue permiso al servicio de AWS. Para obtener más información, consulte los siguientes temas.
   - Creación de una política de IAM para acceder a los recursos de Amazon S3
   - Creación de una política de IAM para acceder a los recursos de AWS Lambda
   - Creación de una política de IAM para acceder a los recursos de CloudWatch Logs
   - Creación de una política de IAM para acceder a los recursos de AWS KMS
2. Cree un rol de IAM y adjúntele la política que ha creado. Para obtener más información, consulte Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.
3. Asocie el rol de IAM al clúster de base de datos Aurora. Para obtener más información, consulte Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.

Creación de una política de IAM para acceder a los recursos de Amazon S3

Aurora puede tener acceso a recursos de Amazon S3 para cargar datos o para guardar datos desde un clúster de base de datos Aurora. Sin embargo, primero debe crear una política de IAM que asigne los permisos de bucket y objeto que hacen posible el acceso de Aurora a Amazon S3.

La tabla siguiente indica las características de Aurora con acceso a un bucket de Amazon S3 en su nombre y los permisos de bucket y objeto mínimos requeridos por cada una.

Característica	Permisos de bucket	Permisos de objeto
`LOAD DATA FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`
`LOAD XML FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`

`SELECT INTO OUTFILE S3`	`ListBucket`	`AbortMultipartUpload`
		`DeleteObject`
		`GetObject`
		`ListMultipartUploadParts`

Característica	Permisos de bucket	Permisos de objeto
		`PutObject`

La siguiente política agrega los permisos que podría requerir Aurora para acceder a un bucket de Amazon S3 en su nombre.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

> **Note**
>
> Asegúrese de incluir las dos entradas para el valor `Resource`. Aurora necesita los permisos tanto en el propio bucket como en todos los objetos dentro del bucket.

> Según su caso de uso, es posible que no necesite agregar todos los permisos en la política de ejemplo. Es posible que se requieran otros permisos. Por ejemplo, si su bucket de Amazon S3 está cifrado, debe añadir permisos `kms:Decrypt`.

Puede seguir los pasos indicados a continuación para crear una política de IAM que otorgue los permisos mínimos necesarios para que Aurora tenga acceso a un bucket de Amazon S3 en su nombre. Para permitir el acceso de Aurora a todos los buckets de Amazon S3 puede omitir estos pasos y usar la política de IAM predefinida `AmazonS3ReadOnlyAccess` o `AmazonS3FullAccess` en lugar de crear la suya propia.

Para crear una política de IAM para dar acceso a los recursos de Amazon S3

1. Abra la Management Console de IAM.
2. En el panel de navegación, seleccione Policies (Políticas).
3. Elija Create Policy.
4. En la pestaña Visual editor (Editor visual), seleccione Choose a service (Elegir un servicio) y, a continuación, S3.
5. Elija Expand all (Ampliar todo) en Actions (Acciones) y, a continuación, elija los permisos de bucket y permisos de objeto necesarios para la política de IAM.

   Los permisos de objeto se refieren a las operaciones de objeto en Amazon S3 y deben concederse para los objetos de un bucket, y no para el bucket en sí. Para obtener más información acerca de los permisos para operaciones de objeto en Amazon S3, consulte Permisos para operaciones de objetos.
6. Elija Resources (Recursos) y Add ARN (Añadir ARN) para bucket.
7. En el cuadro de diálogo Add ARN(s) (Añadir ARN), proporcione los detalles acerca de su recurso y elija Add (Añadir).

   Especifique el bucket de Amazon S3 al que se permitirá obtener acceso. Por ejemplo, si desea conceder a Aurora acceso al bucket de Amazon S3 llamado *`amzn-s3-demo-bucket`*, establezca el valor de Nombre de recurso de Amazon (ARN) en `arn:aws:s3:::`*`amzn-s3-demo-bucket`*.
8. Si se lista el recurso object (objeto), elija Add ARN (Añadir ARN) para object (objeto).
9. En el cuadro de diálogo Add ARN(s) (Añadir ARN), proporcione los detalles acerca de su recurso.

Para el bucket de Amazon S3, especifique el bucket de Amazon S3 al que se permitirá obtener acceso. Para el objeto, puede elegir Any (Cualquiera) para conceder permisos a cualquier objeto del bucket.

> **Note**
>
> Puede establecer en Nombre de recurso de Amazon (ARN) un valor de ARN más específico y que así Aurora solo tenga acceso a archivos o carpetas determinados de un bucket de Amazon S3. Para obtener más información acerca del modo de definir una política de acceso en Amazon S3, consulte Administración de permisos de acceso para los recursos de Amazon S3.

10. (Opcional) Elija Add ARN (Agregar ARN) para el bucket para agregar otro bucket de Amazon S3 a la política y repita los pasos anteriores para el bucket.

> **Note**
>
> Puede repetir esto para añadir las instrucciones de permisos de bucket correspondientes a la política para cada bucket de Amazon S3 al que deba obtener acceso Aurora. Opcionalmente, también puede conceder acceso a todos los buckets y objetos de Amazon S3.

11. Elija Review policy (Revisar política).
12. En Name (Nombre), escriba un nombre para la política de IAM, por ejemplo, `AllowAuroraToExampleBucket`. Utilizará este nombre al crear un rol de IAM y asociarlo al clúster de base de datos Aurora. También puede añadir una descripción opcional en Description (Descripción).
13. Elija Create Policy.
14. Realice los pasos que se indican en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.

Creación de una política de IAM para acceder a los recursos de AWS Lambda

Puede crear una política de IAM que conceda los permisos mínimos necesarios para que Aurora invoque una función de AWS Lambda en su nombre.

La política siguiente agrega los permisos que requiere Aurora para invocar una función de AWS Lambda en su nombre.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToExampleFunction",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:123456789012:function:example_function"
    }
  ]
}
```

Puede seguir los pasos indicados a continuación para crear una política de IAM que conceda los permisos mínimos necesarios para que Aurora invoque una función de AWS Lambda en su nombre. Para permitir que Aurora invoque todas las funciones de AWS Lambda puede omitir estos pasos y usar la política predefinida de `AWSLambdaRole` en lugar de crear la suya propia.

Para crear una política de IAM que permita invocar las funciones de AWS Lambda

1. Abra la consola de IAM.
2. En el panel de navegación, seleccione Policies.
3. Elija Create Policy.
4. En la pestaña Visual editor (Editor visual), seleccione Choose a service (Elegir un servicio) y, a continuación, Lambda.
5. Elija Expand all (Ampliar todo) en Actions (Acciones) y, a continuación, elija los permisos de AWS Lambda necesarios para la política de IAM.

   Asegúrese de que `InvokeFunction` está seleccionado. Es el permiso mínimo necesario para habilitar a Amazon Aurora invocar una función de AWS Lambda.
6. Elija Recursos y Add ARN (Añadir ARN) para función.

7. En el cuadro de diálogo Add ARN(s) (Añadir ARN), proporcione los detalles acerca de su recurso.

   Especifique la función de Lambda a la que se permitirá obtener acceso. Por ejemplo, si desea conceder a Aurora acceso a una función de Lambda llamada `example_function`, establezca como valor de ARN `arn:aws:lambda:::function:example_function`.

   Para obtener más información acerca del modo de definir una política de acceso para AWS Lambda, consulte Autenticación y control de acceso de AWS Lambda.

8. De forma opcional, elija Add additional permissions (Añadir permisos adicionales) para añadir otra función de AWS Lambda a la directiva y repita los pasos anteriores para la función.

   > **Note**
   >
   > Puede repetir esto para agregar las sentencias de permisos de función correspondientes a la política para cada función de AWS Lambda a la que deba acceder Aurora.

9. Elija Review policy (Revisar política).
10. En Nombre, escriba un nombre para la política de IAM, por ejemplo, `AllowAuroraToExampleFunction`. Utilizará este nombre al crear un rol de IAM y asociarlo al clúster de base de datos Aurora. También puede añadir una descripción opcional en Description (Descripción).
11. Elija Create Policy.
12. Realice los pasos que se indican en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.

Creación de una política de IAM para acceder a los recursos de CloudWatch Logs

Aurora puede acceder a CloudWatch Logs para exportar datos de registros de auditoría desde un clúster de base de datos Aurora. Sin embargo, primero debe crear una política de IAM que asigne los permisos de grupo de registros y flujo de registros que hacen posible que Aurora acceda a CloudWatch Logs.

La siguiente política agrega los permisos que requiere Aurora para acceder a Amazon CloudWatch Logs en su nombre y el número mínimo de permisos necesarios para crear grupos de registros y exportar datos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*"
        },
        {
            "Sid":
 "EnableCreationAndManagementOfRDSCloudwatchLogGroupsAndStreams",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutRetentionPolicy",
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*"
        }
    ]
}
```

Puede modificar los ARN de la política para restringir el acceso a una región y cuenta de AWS específicas.

Puede seguir los pasos indicados a continuación para crear una política de IAM que otorgue los permisos mínimos necesarios para que Aurora tenga acceso a CloudWatch Logs en su nombre. Para conceder acceso total de Aurora a CloudWatch Logs puede omitir estos pasos y usar la política de IAM predefinida `CloudWatchLogsFullAccess` en lugar de crear la suya propia. Para obtener más información, consulte Uso de políticas basadas en identidad (políticas de IAM) para CloudWatch Logs en la Guía del usuario de Amazon CloudWatch.

Para crear una política de IAM para dar acceso a los recursos de CloudWatch Logs

1. Abra la consola de IAM.
2. En el panel de navegación, seleccione Policies.
3. Elija Create Policy.
4. En la pestaña Visual editor (Editor visual), elija Choose a service (Elegir un servicio) y, a continuación, CloudWatch Logs.
5. Para Actions (Acciones), elija Expand all (Expandir todo) (a la derecha) y, a continuación, elija los permisos de Amazon CloudWatch Logs necesarios para la política de IAM.

   Compruebe que los permisos siguientes estén seleccionados:

   - `CreateLogGroup`
   - `CreateLogStream`
   - `DescribeLogStreams`
   - `GetLogEvents`
   - `PutLogEvents`
   - `PutRetentionPolicy`
6. Elija Resources (Recursos) y Add ARN (Añadir ARN) para log-group.
7. En el cuadro de diálogo Add ARN(s) (Añadir ARN), escriba los siguientes valores:

   - Región: una región de AWS o *
   - Account (Cuenta): un número de cuenta o *
   - Log Group Name (Nombre del grupo de registro – `/aws/rds/*`
8. En el cuadro de diálogo Add ARN(s) (Agregar ARN), elija Add (Agregar).
9. Elija Add ARN (Añadir ARN) para log-stream.
10. En el cuadro de diálogo Add ARN(s) (Añadir ARN), escriba los siguientes valores:

    - Región: una región de AWS o *
    - Account (Cuenta): un número de cuenta o *
    - Log Group Name (Nombre del grupo de registro – `/aws/rds/*`
    - Log Stream Name (Nombre del flujo de registro – *
11. En el cuadro de diálogo Add ARN(s) (Agregar ARN), elija Add (Agregar).
12. Elija Review policy (Revisar política).

13. En Nombre, escriba un nombre para la política de IAM, por ejemplo, `AmazonRDSCloudWatchLogs`. Utilizará este nombre al crear un rol de IAM y asociarlo al clúster de base de datos Aurora. También puede añadir una descripción opcional en Description (Descripción).
14. Elija Create Policy.
15. Realice los pasos que se indican en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.

Creación de una política de IAM para acceder a los recursos de AWS KMS

Aurora puede obtener acceso a las AWS KMS keys utilizadas para cifrar sus copias de seguridad de bases de datos. Sin embargo, primero debe crear una política de IAM que asigne los permisos que hacen posible que Aurora obtenga acceso a las claves de KMS.

La política siguiente añade los permisos que requiere Aurora para obtener acceso a las claves de KMS en su nombre.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToAccessKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-ID"
    }
  ]
}
```

Puede seguir los pasos indicados a continuación para crear una política de IAM que otorgue los permisos mínimos necesarios para que Aurora tenga acceso a las claves de KMS en su nombre.

Para crear una política de IAM para conceder acceso a sus claves de KMS

1. Abra la consola de IAM.
2. En el panel de navegación, seleccione Policies.
3. Elija Create Policy.
4. En la pestaña Visual editor (Editor visual), seleccione Choose a service (Elegir un servicio) y, a continuación, KMS.
5. En Actions (Acciones), elija Write (Escritura) y después elija Decrypt (Descifrar).
6. Elija Resources (Recursos) y después Add ARN (Añadir ARN).
7. En el cuadro de diálogo Add ARN(s) (Añadir ARN), escriba los siguientes valores:
   - Región: especifique la región de AWS, como `us-west-2`.
   - Cuenta: especifique el número de cuenta de usuario.
   - Nombre del flujo de registros: escriba el identificador de la clave de KMS.
8. En el cuadro de diálogo Add ARN(s) (Agregar ARN), elija Add (Agregar).
9. Elija Review policy (Revisar política).
10. En Nombre, escriba un nombre para la política de IAM, por ejemplo, `AmazonRDSKMSKey`. Utilizará este nombre al crear un rol de IAM y asociarlo al clúster de base de datos Aurora. También puede añadir una descripción opcional en Description (Descripción).
11. Elija Create Policy.
12. Realice los pasos que se indican en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.

Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS

Una vez creada una política de IAM para permitir a Aurora el acceso a recursos de AWS debe crear un rol de IAM y asociarle la política de IAM.

Para crear un rol de IAM que permita al clúster de Amazon RDS comunicarse con otros servicios de AWS en su nombre, siga estos pasos.

Crear un rol de IAM que permita a Amazon RDS el acceso a los servicios de AWS

1. Abra la consola de IAM.
2. Seleccione Roles en el panel de navegación.

3. Elija Crear rol.
4. En Servicio de AWS, elija RDS.
5. En Select your use case (Seleccionar su caso de uso), elija RDS: Add Role to Database (RDS: Añadir rol a base de datos).
6. Elija Siguiente.
7. En la página Permissions policies (Políticas de permisos), introduzca el nombre de su política en el campo Search (Buscar).
8. Cuando aparezca en la lista, seleccione la política que ha definido anteriormente siguiendo las instrucciones de alguna de las siguientes secciones:
   - Creación de una política de IAM para acceder a los recursos de Amazon S3
   - Creación de una política de IAM para acceder a los recursos de AWS Lambda
   - Creación de una política de IAM para acceder a los recursos de CloudWatch Logs
   - Creación de una política de IAM para acceder a los recursos de AWS KMS
9. Elija Siguiente.
10. En Role Name (Nombre del rol), escriba un nombre para el rol de IAM, por ejemplo, `RDSLoadFromS3`. También puede añadir una descripción opcional en Description (Descripción).
11. Elija Create Role (Crear rol).
12. Realice los pasos que se indican en Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.

Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL

Para permitir a los usuarios de una base de datos en un clúster de base de datos de Amazon Aurora acceder a otros servicios de AWS, debe asociar el rol de IAM que creó en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con ese clúster de base de datos. También puedehacer que AWS cree un nuevo rol de IAM asociando el servicio directamente.

> **Note**
>
> No se puede asociar un rol de IAM a un clúster de base de datos de Aurora Serverless v1. Para obtener más información, consulte Uso de Amazon Aurora Serverless v1.
> Se puede asociar un rol de IAM a un clúster de base de datos de Aurora Serverless v2.

Para asociar un rol de IAM con un clúster de base de datos es necesario hacer dos cosas:

1. Añadir el rol a la lista de roles asociados del clúster de base de datos con la consola de RDS, el comando add-role-to-db-cluster de la AWS CLI o la operación AddRoleToDBCluster de la API de RDS.

   Puede añadir un máximo de cinco roles de IAM por cada clúster de base de datos Aurora.

2. Establecer el ARN del rol de IAM asociado en el parámetro a nivel de clúster para el servicio de AWS de que se trate.

   En la tabla siguiente se indican los nombres de los parámetros a nivel de clúster para los roles de IAM empleados en el acceso a otros servicios de AWS.

Parámetro de nivel de clúster	Descripción	
`aws_default_lambda_role`	Se utiliza al invocar una función de Lambda desde el clúster de base de datos.	
`aws_default_logs_role`	Este parámetro ya no es necesario para exportar datos de registros desde el clúster de base de datos a Amazon CloudWatch Logs. Aurora MySQL utiliza ahora un rol vinculado al servicio para los permisos necesarios. Para obtener más información acerca de los roles vinculados a servicios, consulte Uso de roles vinculados a servicios de Amazon Aurora.	
`aws_default_s3_role`	Se utiliza al invocar la instrucción `LOAD DATA FROM S3`, `LOAD XML FROM S3` o `SELECT INTO OUTFILE S3` desde el clúster de base de datos. En la versión 2 de Aurora MySQL, el rol de IAM especificado en este parámetro se usa cuando no se especifica un rol de IAM para `aurora_load_from_s3_role` o	

Parámetro de nivel de clúster	Descripción	
	`aurora_select_into_s3_role` en la instrucción correspondiente. En la versión 3 de Aurora MySQL, siempre se utiliza el rol de IAM especificado para este parámetro.	
`aurora_load_from_s3_role`	Se utiliza al invocar la instrucción `LOAD DATA FROM S3` o `LOAD XML FROM S3` desde el clúster de base de datos. Si no se especifica un rol de IAM para este parámetro, se utiliza el rol de IAM especificado en el parámetro `aws_default_s3_role` . En la versión 3 de Aurora MySQL, este parámetro no está disponible.	
`aurora_select_into_s3_role`	Se utiliza al invocar la instrucción `SELECT INTO OUTFILE S3` desde el clúster de base de datos. Si no se especifica un rol de IAM para este parámetro, se utiliza el rol de IAM especificado en el parámetro `aws_default_s3_role` . En la versión 3 de Aurora MySQL, este parámetro no está disponible.	

Para asociar un rol de IAM que permita al clúster de Amazon RDS comunicarse con otros servicios de AWS en su nombre, siga estos pasos.

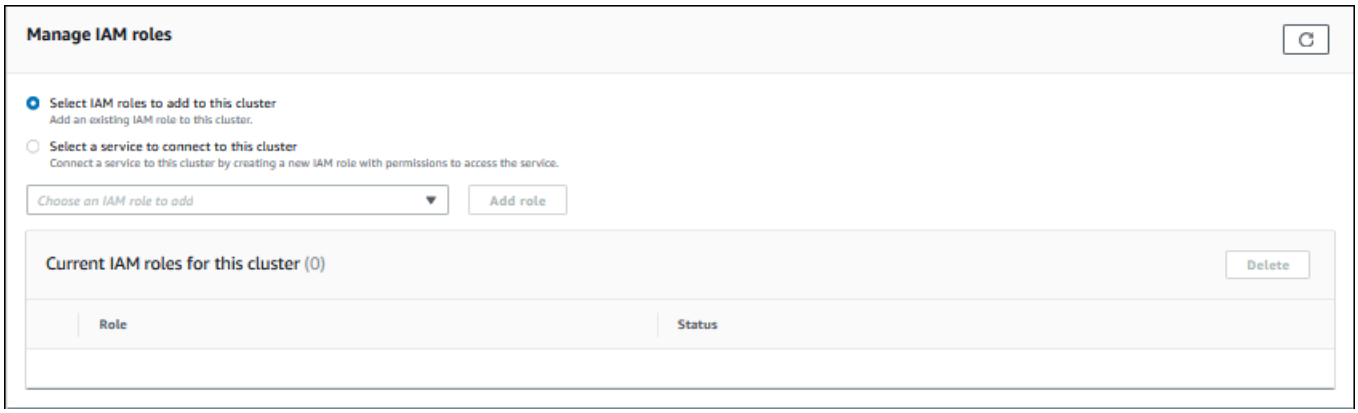
Consola

Para asociar un rol de IAM a un clúster de base de datos Aurora con la consola

1. Abra la consola de RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Databases (Bases de datos).

3. Elija el nombre del clúster de base de datos de Aurora al que desea asociar un rol de IAM para mostrar sus detalles.
4. En la pestaña Connectivity & security (Conectividad y seguridad), en la sección Manage IAM roles (Administrar roles de IAM), realice una de las siguientes acciones:

   - Select IAM roles to add to this cluster (Seleccionar roles de IAM que desea agregar a este clúster) (predeterminado)
   - Select a service to connect to this cluster (Seleccionar un servicio para conectarse a este clúster)



5. Para usar un rol de IAM existente, selecciónelo en el menú y, a continuación, seleccione Add role (Añadir rol).

   Si se agrega el rol correctamente, su estado se muestra como `Pending`, luego como `Available`.
6. Para conectar un servicio directamente:

   a. Elija Select a service to connect to this cluster (Seleccionar un servicio para conectarse a este clúster).

   b. Elija el servicio en el menú y, a continuación, seleccione Connect service (Conectar servicio).

   c. Para Connect cluster to ***`Service Name`*** (Conectar clúster a nombre de servicio), introduzca el Nombre de recurso de Amazon (ARN) que se utilizará para conectarse al servicio y, a continuación, seleccione Connect service (Conectar servicio).

   AWS crea un nuevo rol de IAM para conectarse al servicio. Su estado se muestra como `Pending`, luego como `Available`.

7. (Opcional) Para dejar de asociar un rol de IAM a un clúster de base de datos y retirarle el permiso correspondiente, elija el rol y luego Delete (Eliminar).

Establezca el parámetro de nivel de clúster para cada rol de IAM asociado.

1. En la consola de RDS, elija Parameter groups (Grupos de parámetros) en el panel de navegación.
2. Si ya está usando un grupo de parámetros de base de datos personalizado, puede seleccionarlo para usarlo en lugar de crear uno nuevo. Si utiliza el grupo de parámetros de clúster de base de datos predeterminado, cree un nuevo grupo de parámetros de clúster de base de datos como se describe en los pasos siguientes:

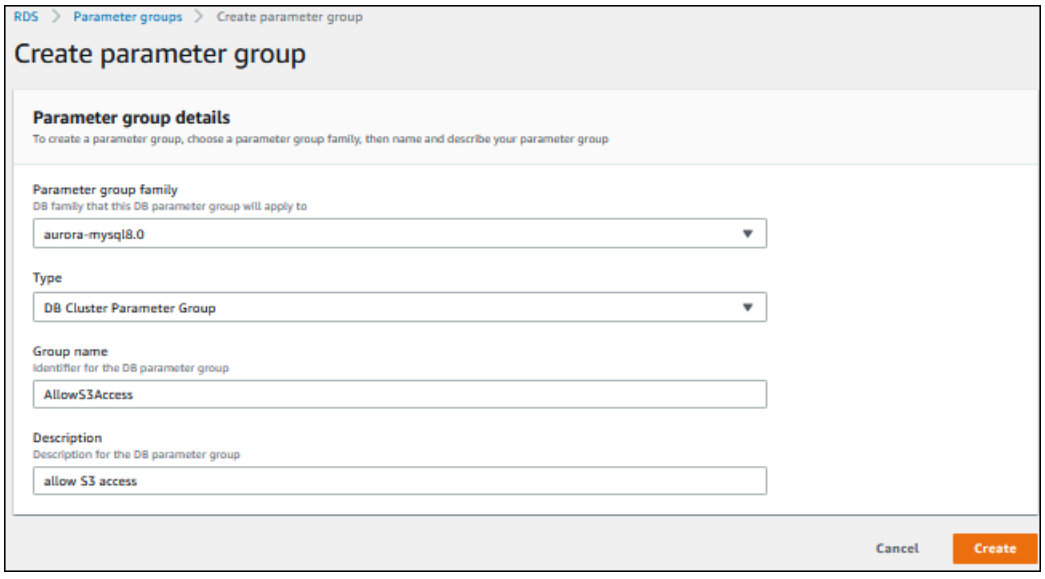
   a. Elija Create parameter group.

   b. En Familia de grupo de parámetros, elija `aurora-mysql8.0` para un clúster de base de datos compatible con Aurora MySQL 8.0 o `aurora-mysql5.7` para un clúster de base de datos compatible con Aurora MySQL 5.7.

   c. En Type (Tipo), elija DB Cluster Parameter Group (Grupo de parámetros de clúster de base de datos).

   d. Para Nombre de grupo, escriba el nombre del nuevo grupo de parámetros de clúster de base de datos.

   e. En Descripción, escriba una descripción del nuevo grupo de parámetros de clúster de base de datos.



   f. Seleccione Crear.

3. En la página Parameter groups (Grupos de parámetros), seleccione su grupo de parámetros de clúster de base de datos y elija Edit (Editar) en Parameter group actions (Acciones de grupos de parámetros).
4. Establezca los parámetros a nivel de clúster adecuados en los valores de ARN de rol de IAM correspondientes.

   Por ejemplo, puede establecer solo en el parámetro `aws_default_s3_role` el valor `arn:aws:iam::123456789012:role/AllowS3Access`.
5. Elija Guardar cambios.
6. Para cambiar el grupo de parámetros del clúster de base de datos para su clúster de base de datos, realice los siguientes pasos:

   a. Elija Databases (Bases de datos) y, a continuación, seleccione el clúster de base de datos de Aurora.

   b. Elija Modify.

   c. Desplácese hasta Database options (Opciones de base de datos) y establezca en DB cluster parameter group (Grupo de parámetros de clúster de base de datos) el grupo de parámetros de clúster de base de datos.

   d. Elija Continue.

   e. Verifique los cambios y elija Apply immediately (Aplicar inmediatamente).

   f. Elija Modify Cluster (Modificar clúster).

   g. Elija Databases (Bases de datos) y, a continuación, seleccione la instancia principal del clúster de base de datos.

   h. Para Actions (Acciones), elija Reboot (Reiniciar).

      Una vez reiniciada la instancia, el rol de IAM se asocia al clúster de base de datos.

      Para obtener más información acerca de los grupos de parámetros de clúster, consulte Parámetros de configuración de Aurora MySQL.

CLI

Para asociar un rol de IAM a un clúster de base de datos con la AWS CLI

1. Ejecute el comando `add-role-to-db-cluster` desde la AWS CLI para añadir los ARN de los roles de IAM al clúster de base de datos, como se muestra a continuación.

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraS3Role
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraLambdaRole
```

2. Si utiliza el grupo de parámetros de clúster de base de datos predeterminado, cree un nuevo grupo de parámetros de clúster de base de datos. Si ya está usando un grupo de parámetros de base de datos personalizado, puede usarlo en lugar de crear uno nuevo.

   Para crear un nuevo grupo de parámetros de clúster de base de datos, ejecute el comando `create-db-cluster-parameter-group` desde la AWS CLI, como se muestra a continuación.

   ```
   PROMPT> aws rds create-db-cluster-parameter-group  --db-cluster-parameter-group-
   name AllowAWSAccess \
        --db-parameter-group-family aurora5.7 --description "Allow access to Amazon S3
    and AWS Lambda"
   ```

   En el caso de un clúster de base de datos compatible con Aurora MySQL 5.7, especifique `aurora-mysql5.7` para `--db-parameter-group-family`. En el caso de un clúster de base de datos compatible con Aurora MySQL 8.0, especifique `aurora-mysql8.0` para `--db-parameter-group-family`.

3. Configure el parámetro o parámetros adecuados a nivel de clúster y los valores de ARN de rol de IAM correspondientes dentro del grupo de parámetros de clúster de base de datos, como se muestra a continuación.

   ```
   PROMPT> aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name
    AllowAWSAccess \
       --parameters
    "ParameterName=aws_default_s3_role,ParameterValue=arn:aws:iam::123456789012:role/
   AllowAuroraS3Role,method=pending-reboot" \
       --parameters
    "ParameterName=aws_default_lambda_role,ParameterValue=arn:aws:iam::123456789012:role/
   AllowAuroraLambdaRole,method=pending-reboot"
   ```

4. Modifique el clúster de base de datos de modo que use el nuevo grupo de parámetros de clúster de base de datos y, a continuación, reinicie el clúster, como se muestra a continuación.

```
PROMPT> aws rds modify-db-cluster --db-cluster-identifier my-cluster --db-cluster-
parameter-group-name AllowAWSAccess
PROMPT> aws rds reboot-db-instance --db-instance-identifier my-cluster-primary
```

Una vez reiniciada la instancia, los roles de IAM están asociados al clúster de base de datos.

Para obtener más información acerca de los grupos de parámetros de clúster, consulte Parámetros de configuración de Aurora MySQL.

Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS

Para utilizar otros servicios de AWS con Amazon Aurora, la configuración de red del clúster de base de datos de Aurora debe permitir conexiones salientes a los puntos de enlace de dichos servicios. Las siguientes operaciones requieren esta configuración de red.

- Invocar funciones de AWS Lambda. Para obtener más información sobre esta característica, consulte Invocación de una función de Lambda con una función nativa de Aurora MySQL.
- Acceso a archivos desde Amazon S3. Para obtener más información sobre esta característica, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3 y Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3.
- Acceso a los puntos de enlace de AWS KMS. Se requiere el acceso de AWS KMS para utilizar los flujos de actividad de la base de datos con Aurora MySQL. Para obtener más información sobre esta característica, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.
- Acceso a los puntos de conexión de IA de SageMaker. Para utilizar el machine learning de IA de SageMaker con Aurora MySQL se necesita acceso a IA de SageMaker. Para obtener más información sobre esta característica, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.

Cuando no puede conectar con un punto de enlace de servicio, Aurora devuelve el mensaje de error siguiente:

```
ERROR 1871 (HY000): S3 API returned error: Network Connection
```

```
ERROR 1873 (HY000): Lambda API returned error: Network Connection. Unable to connect to
 endpoint
```

```
ERROR 1815 (HY000): Internal error: Unable to initialize S3Stream
```

Para flujos de actividad de la base de datos que utilizan Aurora MySQL, el flujo de actividad deja de funcionar si el clúster de base de datos no puede acceder al punto de enlace de AWS KMS. Aurora notifica sobre este problema mediante eventos RDS.

Si encuentra estos mensajes mientras utiliza los servicios de AWS correspondientes, verifique si su clúster de base de datos de Aurora es público o privado. Si el clúster de base de datos de Aurora es privado, debe configurarlo para habilitar las conexiones.

Para que un clúster de base de datos de Aurora sea público debe estar marcado como accesible de forma pública. Si observa los detalles del clúster de base de datos en la Consola de administración de AWS, Publicly Accessible (Accesible públicamente) será Sí en tal caso. Además, el clúster de base de datos debe encontrarse en una subred pública de una Amazon VPC. Para obtener más información acerca de las instancias de bases de datos con acceso público, consulte Uso de una clúster de base de datos en una VPC. Para obtener más información acerca de las subredes de Amazon VPC públicas, consulte VPC y subredes.

Si el clúster de base de datos de Aurora no es de acceso público y se encuentra en una subred pública de una VPC, es privado. Es posible que tenga un clúster de base de datos privado y desee utilizar una de las características que requiere esta configuración de red. Si es así, configure el clúster para que se pueda conectar a direcciones de Internet a través de la traducción de direcciones de red (NAT). Como alternativa para Amazon S3, IA de Amazon SageMaker y AWS Lambda, también puede configurar la VPC para que tenga un punto de conexión de VPC para el otro servicio asociado a la tabla de enrutamiento del clúster de base de datos. Para ello, consulte Uso de una clúster de base de datos en una VPC. Para obtener más información acerca de la configuración de NAT en la VPC, consulte Gateways NAT. Para obtener más información acerca de la configuración de puntos de enlace de la VPC, consulte Puntos de enlace de la VPC. También puede crear un punto de conexión de puerta de enlace de S3 para acceder a su bucket de S3. Para obtener más información, consulte Puntos de conexión de puerta de enlace para Amazon S3.

Es posible que también tenga que abrir los puertos efímeros para sus listas de control de acceso (ACL) de red en las reglas de salida de su grupo de seguridad de la VPC. Para obtener más información sobre los puertos efímeros para las ACL de red, consulte Puertos efímeros en la Guía del usuario de Amazon Virtual Private Cloud.

Temas relacionados de

- Integración de Aurora con otros servicios de AWS
- Administración de un clúster de base de datos de Amazon Aurora

Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3

Puede utilizar la instrucción `LOAD DATA FROM S3` o `LOAD XML FROM S3` para cargar datos de archivos almacenados en un bucket de Amazon S3. En Aurora MySQL, los archivos se almacenan primero en el disco local y, a continuación, se exportan a la base de datos. Una vez finalizadas las importaciones a la base de datos, se eliminan los archivos locales.

> **Note**
>
> La carga de datos en una tabla desde archivos de texto no se admite en Aurora Serverless v1. Se admite para Aurora Serverless v2.

Contenido

- Otorgar acceso a Aurora a Amazon S3
- Concesión de privilegios para cargar datos en Amazon Aurora MySQL
- Especificación de una ruta (URI) a un bucket de Amazon S3
- LOAD DATA FROM S3
  - Sintaxis
  - Parámetros
  - Uso de un manifiesto para especificar los archivos de datos que se deben cargar
    - Comprobación de los archivos cargados mediante la tabla aurora_s3_load_history
  - Ejemplos
- LOAD XML FROM S3
  - Sintaxis
  - Parámetros

Otorgar acceso a Aurora a Amazon S3

Para poder cargar datos desde un bucket de Amazon S3, primero debe dar al clúster de base de datos Aurora MySQL permiso para que obtenga acceso a Amazon S3.

Para conceder a Aurora MySQL acceso a Amazon S3

1. Cree una política de AWS Identity and Access Management (IAM) que asigne los permisos de bucket y objeto que permiten que su clúster de base de datos de Aurora MySQL acceda a Amazon S3. Para obtener instrucciones, consulte Creación de una política de IAM para acceder a los recursos de Amazon S3.

   > **Note**
   >
   > En la versión 3.05 de Aurora MySQL y versiones posteriores, puede cargar objetos cifrados mediante AWS KMS keys cifrado por el cliente. Para ello, incluya el permiso `kms:Decrypt` en su política de IAM. Para obtener más información, consulte Creación de una política de IAM para acceder a los recursos de AWS KMS.
   > No necesita este permiso para cargar objetos cifrados mediante Claves administradas por AWS o claves administradas de Amazon S3 (SSE-S3).

2. Cree un rol de IAM y asocie la política de IAM que creó en Creación de una política de IAM para acceder a los recursos de Amazon S3 al nuevo rol de IAM. Para obtener instrucciones, consulte Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.
3. Asegúrese de que el clúster de base de datos utiliza un grupo de parámetros del clúster de base de datos.

   Para obtener más información acerca de cómo crear un grupo de parámetros del clúster de base de datos personalizado, consulte Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.
4. Para la versión 2 de Aurora MySQL, establezca el parámetro de clúster de base de datos `aurora_load_from_s3_role` o `aws_default_s3_role` en el nombre de recurso de Amazon (ARN) del nuevo rol de IAM. Si no se ha especificado un rol de IAM para `aurora_load_from_s3_role`, Aurora utilizará el rol de IAM especificado en el parámetro `aws_default_s3_role`.

   Para Aurora MySQL versión 3, use `aws_default_s3_role`.

Si el clúster es parte de una base de datos global de Aurora, configure este parámetro para cada clúster de Aurora en la base de datos global. Aunque solo el clúster principal de una base de datos global de Aurora puede cargar datos, se puede promover otro clúster por parte del mecanismo de conmutación por error y convertirse en clúster principal.

Para obtener más información acerca de los parámetros de clúster de base de datos, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos.

5. Para permitir el acceso a Aurora MySQL a los usuarios de base de datos un clúster de base de datos Amazon S3, asocie el rol que creó en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con el clúster. Para una base de datos global de Aurora, asocie el rol con cada clúster de Aurora en la base de datos global. Para obtener información acerca de cómo asociar un rol de IAM con un clúster de base de datos, consulte Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.
6. Configure el clúster de base de datos Aurora MySQL para permitir conexiones salientes hacia Amazon S3. Para obtener instrucciones, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

   Si el clúster de base de datos de no es de acceso público y se encuentra en una subred pública de una VPC, es privado. Puede crear un punto de conexión de puerta de enlace de S3 para acceder a su bucket de S3. Para obtener más información, consulte Puntos de conexión de puerta de enlace para Amazon S3.

   Para una base de datos global de Aurora, habilite las conexiones de salida para cada clúster de Aurora en la base de datos global.

Concesión de privilegios para cargar datos en Amazon Aurora MySQL

El usuario de la base de datos que utiliza la instrucción `LOAD DATA FROM S3` o `LOAD XML FROM S3` debe tener un rol específico o el privilegio para poder hacerlo. En Aurora MySQL versión 3, usted otorga el rol de `AWS_LOAD_S3_ACCESS`. En la versión 2 de Aurora MySQL, usted otorga el privilegio `LOAD FROM S3`. El usuario administrativo de un clúster de base de datos tiene el rol o privilegio adecuados de forma predeterminada. Para conceder el privilegio a otro usuario, puede usar una de las instrucciones siguientes.

Utilice la siguiente instrucción para Aurora MySQL versión 3:

```
GRANT AWS_LOAD_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Cuando utiliza la técnica de rol en Aurora MySQL versión 3, también puede activar el rol mediante la instrucción `SET ROLE role_name` o `SET ROLE ALL`. Si no está familiarizado con el sistema de roles MySQL 8.0, puede obtener más información en Modelo de privilegios basado en roles. Para obtener más información, consulte Using roles en el Manual de referencia de MySQL.
> Esto solo se aplica a la sesión activa actual. Cuando se vuelva a conectar, debe ejecutar la instrucción `SET ROLE` de nuevo para conceder privilegios. Para obtener más información, consulte la instrucción SET ROLE en el Manual de referencia de MySQL.
> Puede utilizar el parámetro de clúster de base de datos `activate_all_roles_on_login` para activar automáticamente todos los roles cuando un usuario se conecta a una instancia de base de datos. Cuando se establece este parámetro, por lo general no tiene que llamar a la instrucción `SET ROLE` para activar un rol. Para obtener más información, consulte activate_all_roles_on_login en el Manual de referencia de MySQL.
> Sin embargo, debe llamar a `SET ROLE ALL` de forma explícita al principio de un procedimiento almacenado para activar el rol cuando un usuario diferente llame al procedimiento almacenado.

Utilice la siguiente instrucción para la versión 2 de Aurora MySQL:

```
GRANT LOAD FROM S3 ON *.* TO 'user'@'domain-or-ip-address'
```

El rol `AWS_LOAD_S3_ACCESS` y el privilegio `LOAD FROM S3` son específicos de Amazon Aurora y no están disponibles en las bases de datos de MySQL externas ni en las instancias de base de datos de RDS para MySQL. Si ha configurado replicación entre un clúster de base de datos de Aurora como origen y una base de datos MySQL como cliente, la instrucción `GRANT` para el rol o privilegio hará que la replicación se detenga con un error. Puede pasar por alto el error para continuar la replicación. Para pasar por alto el error en una instancia de RDS para MySQL, utilice el procedimiento mysql_rds_skip_repl_error. Para omitir el error en una base de datos MySQL externa, utilice la variable de sistema slave_skip_errors (Aurora MySQL versión 2) o la variable de sistema replica_skip_errors (Aurora MySQL versión 3).

> **Note**
>
> El usuario de la base de datos debe contar con privilegios de `INSERT` para la base de datos en la que carga los datos.

Especificación de una ruta (URI) a un bucket de Amazon S3

La sintaxis para especificar la ruta de acceso (URI) a los archivos almacenados en un bucket de Amazon S3 es la siguiente.

```
s3-region://amzn-s3-demo-bucket/file-name-or-prefix
```

La ruta incluye los siguientes valores:

- `region` (opcional): la región de AWS que contiene el bucket de Amazon S3 desde el que se va a realizar la carga. Este valor es opcional. Si no se especifica el valor `region`, Aurora carga el archivo de Amazon S3 desde la misma región en la que se encuentra el clúster de base de datos.
- `bucket-name`: el nombre del bucket de Amazon S3 que contiene los datos que se van a cargar. Pueden usarse prefijos de objeto que identifiquen una ruta de carpeta virtual.
- `file-name-or-prefix`: el nombre del archivo de texto o el archivo XML de Amazon S3, o un prefijo que identifica el archivo o los archivos de texto o XML que se van a cargar. También puede especificar un archivo de manifiesto que identifique el archivo o los archivos de texto que se van a cargar. Para obtener más información acerca de cómo utilizar un archivo de manifiesto para cargar archivos de texto desde Amazon S3, consulte Uso de un manifiesto para especificar los archivos de datos que se deben cargar.

Copia del URI para los archivos de un bucket de S3

1. Inicie sesión Consola de administración de AWS Management Console y abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.
2. En el panel de navegación, elija Buckets y, a continuación, elija el bucket cuyo URI desea copiar.
3. Seleccione el prefijo o el archivo que desee cargar desde S3.
4. Elija Copiar URI de S3.

LOAD DATA FROM S3

Puede utilizar la instrucción `LOAD DATA FROM S3` para cargar datos desde archivos de texto con cualquier formato que sea compatible con la instrucción LOAD DATA INFILE de MySQL, como los datos de texto delimitados por comas. No se admiten los archivos comprimidos.

> **Note**
>
> Asegúrese de que su clúster de base de datos de Aurora MySQL permita conexiones salientes a S3. Para obtener más información, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

Sintaxis

```
LOAD DATA [FROM] S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

> **Note**
>
> En la versión 3.05 de Aurora MySQL y versiones posteriores, la palabra clave `FROM` es opcional.

Parámetros

La instrucción `LOAD DATA FROM S3` utiliza los siguientes parámetros obligatorios y opcionales. Puede encontrar más información acerca de algunos de estos parámetros en LOAD DATA Statement (Instrucción LOAD DATA) en la documentación de MySQL.

FILE | PREFIX | MANIFEST

Identifica si se deben cargar los datos de un solo archivo, de todos los archivos que coincidan con un prefijo determinado o de todos los archivos del manifiesto especificado. `FILE` es el valor predeterminado.

S3-URI

Especifica el URI del archivo de texto o de manifiesto que se debe cargar, o el prefijo de Amazon S3 que se debe utilizar. Especifique el URI utilizando la sintaxis descrita en Especificación de una ruta (URI) a un bucket de Amazon S3.

REPLACE | IGNORE

Determina la acción que se debe realizar si una fila de entrada tiene los mismos valores de clave única que una fila existente en la tabla de la base de datos.

- Especifique `REPLACE` si desea que la fila de entrada sustituya la fila existente en la tabla.
- Especifique `IGNORE` si desea descartar la fila de entrada.

INTO TABLE

Identifica el nombre de la tabla de la base de datos en la que se deben cargar las filas de entrada.

PARTITION

Requiere que todas las filas de entrada se inserten en las particiones identificadas por la lista especificada que contiene los nombres de las particiones separados por comas. Si no se puede insertar una fila de entrada en una de las particiones especificadas, la instrucción falla y se devuelve un error.

CHARACTER SET

Identifica el conjunto de caracteres de los datos del archivo de entrada.

FIELDS | COLUMNS

Identifica cómo están delimitados los campos o las columnas del archivo de entrada. De forma predeterminada, los campos están delimitados por tabuladores.

LINES

Identifica cómo están delimitadas las líneas del archivo de entrada. De forma predeterminada, las líneas están delimitadas por un carácter de nueva línea (`'\n'`).

IGNORE ***number*** LINES | ROWS

Especifica que se deben omitir cierto número de líneas o filas al principio del archivo de entrada. Por ejemplo, puede utilizar `IGNORE 1 LINES` para omitir una línea de encabezado inicial que contiene los nombres de las columnas o `IGNORE 2 ROWS` para omitir las dos primeras filas de datos del archivo de entrada. Si también utiliza `PREFIX`, `IGNORE` omite cierto número de líneas o filas al principio del primer archivo de entrada.

col_name_or_user_var, ...

Especifica una lista separada por comas de uno o varios nombres de columna o variables de usuario que identifican las columnas que se deben cargar por nombre. El nombre de una variable de usuario utilizada para este propósito debe coincidir con el nombre de un elemento del archivo de texto, con el prefijo @. Puede utilizar variables de usuario para almacenar los valores de los campos correspondientes para utilizarlos posteriormente.

Por ejemplo, la siguiente instrucción carga la primera columna del archivo de entrada en la primera columna de `table1`, y establece el valor de la columna `table_column2` de `table1` en el valor de entrada de la segunda columna, dividido por 100.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, @var1)
    SET table_column2 = @var1/100;
```

SET

Especifica una lista separada por comas que contiene operaciones de asignación que asignan valores no incluidos en el archivo de entrada a las columnas de la tabla.

Por ejemplo, la siguiente instrucción asigna a las dos primeras columnas de `table1` los valores de las dos primeras columnas del archivo de entrada y, a continuación, asigna la fecha y hora actuales a `column3` de `table1`.

```
LOAD DATA FROM S3  's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
```

```
    (column1, column2)
    SET column3 = CURRENT_TIMESTAMP;
```

Es posible utilizar subconsultas en el lado derecho de las asignaciones `SET`. Para una subconsulta que devuelve un valor que se va a asignar a una columna, solo se puede utilizar una subconsulta escalar. Además, no puede utilizar una subconsulta para seleccionar datos de la tabla que se está cargando.

No se puede utilizar la palabra clave `LOCAL` de la instrucción `LOAD DATA FROM S3` para cargar datos de un bucket de Amazon S3.

Uso de un manifiesto para especificar los archivos de datos que se deben cargar

Puede utilizar la instrucción `LOAD DATA FROM S3` con la palabra clave `MANIFEST` para especificar un archivo de manifiesto con formato JSON que enumera los archivos de texto que se cargarán en una tabla del clúster de base de datos.

El siguiente esquema JSON describe el formato y el contenido de un archivo de manifiesto.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "Aurora_LoadFromS3_Manifest",
    "properties": {
        "entries": {
            "additionalItems": false,
            "id": "/properties/entries",
            "items": {
                "additionalProperties": false,
                "id": "/properties/entries/items",
                "properties": {
                    "mandatory": {
                        "default": "false",
                        "id": "/properties/entries/items/properties/mandatory",
                        "type": "boolean"
                    },
                    "url": {
                        "id": "/properties/entries/items/properties/url",
                        "maxLength": 1024,
                        "minLength": 1,
                        "type": "string"
```

```
                    }
                },
                "required": [
                    "url"
                ],
                "type": "object"
            },
            "type": "array",
            "uniqueItems": true
        }
    },
    "required": [
        "entries"
    ],
    "type": "object"
}
```

Cada `url` del manifiesto debe especificar una URL con el nombre del bucket y la ruta de objeto completa del archivo, no solo un prefijo. Puede usar un manifiesto para cargar archivos de diferentes buckets o regiones, o archivos que no comparten el mismo prefijo. Si no se especifica una región en la URL, se utiliza la región del clúster de base de datos de destino de Aurora. En el siguiente ejemplo se muestra un archivo de manifiesto que carga cuatro archivos desde distintos buckets.

```
{
  "entries": [
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":true
    },
    {
      "url":"s3-us-west-2://aurora-bucket-usw2/2013-10-05-customerdata",
      "mandatory":true
    },
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":false
    },
    {
      "url":"s3://aurora-bucket/2013-10-05-customerdata"
    }
  ]
}
```

La marca opcional `mandatory` especifica si `LOAD DATA FROM S3` debe devolver un error en caso de que no se encuentre el archivo. De forma predeterminada, la marca `mandatory` tiene el valor `false`. Sea cual sea el valor de `mandatory`, `LOAD DATA FROM S3` finaliza si no se encuentra ningún archivo.

Los archivos de manifiesto pueden tener cualquier extensión. En el siguiente ejemplo, se ejecuta la instrucción `LOAD DATA FROM S3` con el manifiesto del ejemplo anterior, que se denomina **`customer.manifest`**.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/customer.manifest'
    INTO TABLE CUSTOMER
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL);
```

Una vez finalizada la instrucción, se escribe una entrada en la tabla `aurora_s3_load_history` para cada archivo que se ha cargado correctamente.

Comprobación de los archivos cargados mediante la tabla aurora_s3_load_history

Cada vez que se ejecuta correctamente la instrucción `LOAD DATA FROM S3`, se actualiza la tabla `aurora_s3_load_history` del esquema `mysql` con una entrada para cada archivo que se ha cargado.

Después de ejecutar la instrucción `LOAD DATA FROM S3`, puede comprobar qué archivos se han cargado consultando la tabla `aurora_s3_load_history`. Para ver los archivos que se cargaron durante una iteración de la instrucción, utilice la cláusula `WHERE` para filtrar los registros utilizando el URI de Amazon S3 del archivo de manifiesto utilizado en la instrucción. Si ha utilizado el mismo archivo de manifiesto anteriormente, filtre los resultados utilizando el campo `timestamp`.

```
select * from mysql.aurora_s3_load_history where load_prefix = 'S3_URI';
```

En la siguiente tabla se describen los campos de la tabla `aurora_s3_load_history`.

Campo	Descripción
`load_prefix`	El URI que se especificó en la instrucción load. Este URI puede mapearse a cualquiera de los elementos siguientes:

Campo	Descripción
	• Un solo archivo de datos para una instrucción `LOAD DATA FROM S3 FILE` • Un prefijo de Amazon S3 mapeado a varios archivos de datos para una instrucción `LOAD DATA FROM S3 PREFIX` • Un único archivo de manifiesto que contiene los nombres de los archivos que se van a cargar para una instrucción `LOAD DATA FROM S3 MANIFEST`
`file_name`	El nombre de un archivo que se cargó en Aurora desde Amazon S3 utilizando el URI identificado en el campo `load_prefix` .
`version_number`	El número de versión del archivo identificado por el campo `file_name` que se cargó, si el bucket de Amazon S3 tiene un número de versión.
`bytes_loaded`	El tamaño del archivo cargado, en bytes.
`load_timestamp`	La fecha y hora en que finalizó la instrucción `LOAD DATA FROM S3`.

Ejemplos

La siguiente instrucción carga datos desde un bucket de Amazon S3 que se encuentra en la misma región que el clúster de base de datos Aurora. La instrucción lee los datos delimitados por comas del archivo `customerdata.txt`, que se encuentra en el bucket de Amazon S3 *`amzn-s3-demo-bucket`*, y carga los datos en la tabla `store-schema.customer-table`.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/customerdata.csv'
    INTO TABLE store-schema.customer-table
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, ADDRESS, EMAIL, PHONE);
```

La siguiente instrucción carga datos desde un bucket de Amazon S3 que se encuentra en una región que no coincide con la del clúster de base de datos Aurora. La instrucción lee los datos delimitados por comas de todos los archivos que coinciden con el prefijo de objeto `employee-data` en el bucket

de Amazon S3 *amzn-s3-demo-bucket*, en la región us-west-2 y carga los datos en la tabla employees.

```
LOAD DATA FROM S3 PREFIX 's3-us-west-2://amzn-s3-demo-bucket/employee_data'
    INTO TABLE employees
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL, SALARY);
```

La siguiente instrucción carga los datos de los archivos especificados en un archivo de manifiesto JSON denominado q1_sales.json en la tabla sales.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://amzn-s3-demo-bucket1/q1_sales.json'
    INTO TABLE sales
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (MONTH, STORE, GROSS, NET);
```

LOAD XML FROM S3

Puede utilizar la instrucción LOAD XML FROM S3 para cargar datos de archivos XML almacenados en un bucket de Amazon S3 con uno de los tres formatos XML siguientes:

- Con los nombres de las columnas como atributos de un elemento <row>. El valor del atributo identifica el contenido del campo de la tabla.

  ```
  <row column1="value1" column2="value2" .../>
  ```

- Con los nombres de las columnas como elementos secundarios de un elemento <row>. El valor del elemento secundario identifica el contenido del campo de la tabla.

  ```
  <row>
    <column1>value1</column1>
    <column2>value2</column2>
  </row>
  ```

- Con los nombres de las columnas en el atributo name de los elementos <field> de un elemento <row>. El valor del elemento <field> identifica el contenido del campo de la tabla.

  ```
  <row>
    <field name='column1'>value1</field>
  ```

```
  <field name='column2'>value2</field>
</row>
```

Sintaxis

```
LOAD XML FROM S3 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<element-name>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Parámetros

La instrucción `LOAD XML FROM S3` utiliza los siguientes parámetros obligatorios y opcionales. Puede encontrar más información acerca de algunos de estos parámetros en LOAD XML Statement (Instrucción LOAD XML) en la documentación de MySQL.

FILE | PREFIX

Identifica si se deben cargar los datos de un solo archivo o de todos los archivos que coincidan con un prefijo determinado. `FILE` es el valor predeterminado.

REPLACE | IGNORE

Determina la acción que se debe realizar si una fila de entrada tiene los mismos valores de clave única que una fila existente en la tabla de la base de datos.

- Especifique `REPLACE` si desea que la fila de entrada sustituya la fila existente en la tabla.
- Especifique `IGNORE` si desea para descartar la fila de entrada. `IGNORE` es el valor predeterminado.

INTO TABLE

Identifica el nombre de la tabla de la base de datos en la que se deben cargar las filas de entrada.

PARTITION

Requiere que todas las filas de entrada se inserten en las particiones identificadas por la lista especificada que contiene los nombres de las particiones separados por comas. Si no se puede

insertar una fila de entrada en una de las particiones especificadas, la instrucción falla y se devuelve un error.

CHARACTER SET

Identifica el conjunto de caracteres de los datos del archivo de entrada.

ROWS IDENTIFIED BY

Establece el nombre del elemento que identifica una fila del archivo de entrada. El valor predeterminado es `<row>`.

IGNORE ***number*** LINES | ROWS

Especifica que se deben omitir cierto número de líneas o filas al principio del archivo de entrada. Por ejemplo, puede utilizar `IGNORE 1 LINES` para omitir la primera línea del archivo de texto o `IGNORE 2 ROWS` para omitir las dos primeras filas de datos del archivo XML de entrada.

field_name_or_user_var, ...

Especifica una lista separada por comas de uno o varios elementos XML o variables de usuario que identifican los elementos que se deben cargar por nombre. El nombre de una variable de usuario utilizada para este propósito debe coincidir con el nombre de un elemento del archivo XML, con el prefijo @. Puede utilizar variables de usuario para almacenar los valores de los campos correspondientes para utilizarlos posteriormente.

Por ejemplo, la siguiente instrucción carga la primera columna del archivo de entrada en la primera columna de `table1`, y establece el valor de la columna `table_column2` de `table1` en el valor de entrada de la segunda columna, dividido por 100.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, @var1)
   SET table_column2 = @var1/100;
```

SET

Especifica una lista separada por comas que contiene operaciones de asignación que asignan valores no incluidos en el archivo de entrada a las columnas de la tabla.

Por ejemplo, la siguiente instrucción asigna a las dos primeras columnas de `table1` los valores de las dos primeras columnas del archivo de entrada y, a continuación, asigna la fecha y hora actuales a `column3` de `table1`.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, column2)
   SET column3 = CURRENT_TIMESTAMP;
```

Es posible utilizar subconsultas en el lado derecho de las asignaciones `SET`. Para una subconsulta que devuelve un valor que se va a asignar a una columna, solo se puede utilizar una subconsulta escalar. Además, no puede utilizar una subconsulta para seleccionar datos de la tabla que se está cargando.

Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3

Puede usar la instrucción `SELECT INTO OUTFILE S3` para consultar datos de un clúster de base de datos de Amazon Aurora MySQL y guardarlos directamente en archivos de texto almacenados en un bucket de Amazon S3. En Aurora MySQL, los archivos se almacenan primero en el disco local y, a continuación, se exportan a S3. Una vez finalizadas las exportaciones, se eliminan los archivos locales.

Puede cifrar el bucket de Amazon S3 mediante una clave administrada de Amazon S3 (SSE-S3) o AWS KMS key (SSE-KMS: Clave administrada de AWS o clave administrada por el cliente).

La instrucción `LOAD DATA FROM S3` puede usar archivos creados mediante la instrucción `SELECT INTO OUTFILE S3` para cargar datos en un clúster de base de datos de Aurora. Para obtener más información, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.

> **Note**
>
> Esta característica no se admite para clústeres de base de datos de Aurora Serverless v1. Es compatible con clústeres de bases de datos de Aurora Serverless v2.
> También puede guardar datos de clúster de base de datos y datos de instantáneas de clúster de base de datos en Amazon S3 mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS. Para obtener más información, consulte Exportación de datos del clúster de base de datos a Amazon S3 y Exportación de datos de instantánea del clúster de bases de datos a Amazon S3.

Contenido

- Otorgar acceso a Aurora MySQL a Amazon S3
- Concesión de privilegios para guardar datos en Aurora MySQL
- Especificación de una ruta a un bucket de Amazon S3
- Creación de un manifiesto con una lista de los archivos de datos
- SELECT INTO OUTFILE S3
  - Sintaxis
  - Parámetros
  - Consideraciones
  - Ejemplos

Otorgar acceso a Aurora MySQL a Amazon S3

Para poder guardar datos en un bucket de Amazon S3, primero debe dar permiso al clúster de base de datos Aurora MySQL para que tenga acceso a Amazon S3.

Para conceder a Aurora MySQL acceso a Amazon S3

1. Cree una política de AWS Identity and Access Management (IAM) que asigne los permisos de bucket y objeto que permiten que su clúster de base de datos de Aurora MySQL acceda a Amazon S3. Para obtener instrucciones, consulte Creación de una política de IAM para acceder a los recursos de Amazon S3.

   > **Note**
   >
   > En Aurora MySQL versión 3.05 y versiones posteriores, puede cifrar objetos mediante claves administradas por el cliente AWS KMS. Para ello, incluya el permiso `kms:GenerateDataKey` en su política de IAM. Para obtener más información, consulte Creación de una política de IAM para acceder a los recursos de AWS KMS.
   > No necesita este permiso para cifrar objetos mediante Claves administradas por AWS o claves administradas de Amazon S3 (SSE-S3).

2. Cree un rol de IAM y asocie la política de IAM que creó en Creación de una política de IAM para acceder a los recursos de Amazon S3 al nuevo rol de IAM. Para obtener instrucciones, consulte Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.

3. Para la versión 2 de Aurora MySQL, establezca el parámetro de clúster de base de datos `aurora_select_into_s3_role` o `aws_default_s3_role` en el nombre de recurso de Amazon (ARN) del nuevo rol de IAM. Si no se ha especificado un rol de IAM para `aurora_select_into_s3_role`, Aurora utilizará el rol de IAM especificado en el parámetro `aws_default_s3_role`.

   Para Aurora MySQL versión 3, use `aws_default_s3_role`.

   Si el clúster es parte de una base de datos global de Aurora, configure este parámetro para cada clúster de Aurora en la base de datos global.

   Para obtener más información acerca de los parámetros de clúster de base de datos, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos.
4. Para permitir el acceso a Aurora MySQL a los usuarios de base de datos un clúster de base de datos Amazon S3, asocie el rol que creó en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con el clúster.

   Para una base de datos global de Aurora, asocie el rol con cada clúster de Aurora en la base de datos global.

   Para obtener información acerca de cómo asociar un rol de IAM con un clúster de base de datos, consulte Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.
5. Configure el clúster de base de datos Aurora MySQL para permitir conexiones salientes hacia Amazon S3. Para obtener instrucciones, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

   Para una base de datos global de Aurora, habilite las conexiones de salida para cada clúster de Aurora en la base de datos global.

Concesión de privilegios para guardar datos en Aurora MySQL

El usuario de la base de datos que utilice la instrucción `SELECT INTO OUTFILE S3` debe tener un rol o privilegio específicos. En Aurora MySQL versión 3, usted otorga el rol de `AWS_SELECT_S3_ACCESS`. En la versión 2 de Aurora MySQL, usted otorga el privilegio `SELECT INTO S3`. El usuario administrativo de un clúster de base de datos tiene el rol o privilegio adecuados de forma predeterminada. Para conceder el privilegio a otro usuario, puede usar una de las instrucciones siguientes.

Utilice la siguiente instrucción para Aurora MySQL versión 3:

```
GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Cuando utiliza la técnica de rol en Aurora MySQL versión 3, también puede activar el rol mediante la instrucción `SET ROLE role_name` o `SET ROLE ALL`. Si no está familiarizado con el sistema de roles MySQL 8.0, puede obtener más información en Modelo de privilegios basado en roles. Para obtener más información, consulte Using roles en el Manual de referencia de MySQL.
>
> Esto solo se aplica a la sesión activa actual. Cuando se vuelva a conectar, debe ejecutar la instrucción `SET ROLE` de nuevo para conceder privilegios. Para obtener más información, consulte la instrucción SET ROLE en el Manual de referencia de MySQL.
>
> Puede utilizar el parámetro de clúster de base de datos `activate_all_roles_on_login` para activar automáticamente todos los roles cuando un usuario se conecta a una instancia de base de datos. Cuando se establece este parámetro, por lo general no tiene que llamar a la instrucción `SET ROLE` para activar un rol. Para obtener más información, consulte activate_all_roles_on_login en el Manual de referencia de MySQL.
>
> Sin embargo, debe llamar a `SET ROLE ALL` de forma explícita al principio de un procedimiento almacenado para activar el rol cuando un usuario diferente llame al procedimiento almacenado.

Utilice la siguiente instrucción para la versión 2 de Aurora MySQL:

```
GRANT SELECT INTO S3 ON *.* TO 'user'@'domain-or-ip-address'
```

El rol `AWS_SELECT_S3_ACCESS` y el privilegio `SELECT INTO S3` son específicos de Amazon Aurora MySQL y no están disponibles en las bases de datos de MySQL ni en las instancias de base de datos de RDS for MySQL. Si ha configurado replicación entre un clúster de base de datos Aurora MySQL como origen y una base de datos MySQL como cliente, la instrucción `GRANT` para el rol o privilegio hará que la replicación se detenga con un error. Puede pasar por alto el error para continuar la replicación. Para pasar por alto el error en una instancia de base de datos de RDS for MySQL, utilice el procedimiento mysql_rds_skip_repl_error. Para omitir el error en una base de datos MySQL externa, utilice la variable de sistema slave_skip_errors (Aurora MySQL versión 2) o la variable de sistema replica_skip_errors (Aurora MySQL versión 3).

Especificación de una ruta a un bucket de Amazon S3

La sintaxis para especificar una ruta donde almacenar los datos y los archivos de manifiesto en un bucket de Amazon S3 es similar a la empleada en la instrucción `LOAD DATA FROM S3 PREFIX`, como se indica a continuación:

```
s3-region://bucket-name/file-prefix
```

La ruta incluye los siguientes valores:

- `region` (opcional): la región de AWS que contiene el bucket de Amazon S3 para guardar los datos. Este valor es opcional. Si no especifica el valor `region`, Aurora guarda los archivos en Amazon S3 en la misma región en la que se encuentra el clúster de base de datos.
- `bucket-name`: nombre del bucket de Amazon S3 donde se guardan los datos. Pueden usarse prefijos de objeto que identifiquen una ruta de carpeta virtual.
- `file-prefix`: prefijo de objeto de Amazon S3 que identifica los archivos que se guardan en Amazon S3.

Los archivos de datos creados con la instrucción `SELECT INTO OUTFILE S3` usan la ruta siguiente, donde *`00000`* representa un número entero de cinco dígitos basado en cero.

```
s3-region://bucket-name/file-prefix.part_00000
```

Por ejemplo, supongamos que la instrucción `SELECT INTO OUTFILE S3` especifica `s3-us-west-2://bucket/prefix` como la ruta donde almacenar los archivos de datos y que crea tres archivos. El bucket de Amazon S3 especificado contendrá los archivos de datos siguientes:

- s3-us-west-2://bucket/prefix.part_00000
- s3-us-west-2://bucket/prefix.part_00001
- s3-us-west-2://bucket/prefix.part_00002

Creación de un manifiesto con una lista de los archivos de datos

Puede utilizar la instrucción `SELECT INTO OUTFILE S3` con la opción `MANIFEST ON` para crear un archivo de manifiesto con formato JSON que enumera los archivos de texto creados por la instrucción. La instrucción `LOAD DATA FROM S3` puede usar el archivo de manifiesto para volver

a cargar los archivos de datos en un clúster de base de datos Aurora MySQL. Para obtener más información acerca de cómo utilizar un archivo de manifiesto para cargar archivos de texto desde Amazon S3 en un clúster de base de datos Aurora MySQL, consulte Uso de un manifiesto para especificar los archivos de datos que se deben cargar.

Los archivos de datos incluidos en el manifiesto creado con la instrucción `SELECT INTO OUTFILE S3` se enumeran en el orden en que la instrucción los ha creado. Por ejemplo, supongamos que la instrucción `SELECT INTO OUTFILE S3` especificó `s3-us-west-2://bucket/prefix` como la ruta donde almacenar los archivos de datos y que ha creado tres archivos de datos y un archivo de manifiesto. El bucket de Amazon S3 especificado contendrá un archivo de manifiesto llamado `s3-us-west-2://bucket/prefix.manifest` que contiene la información siguiente:

```
{
  "entries": [
    {
      "url":"s3-us-west-2://bucket/prefix.part_00000"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00001"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00002"
    }
  ]
}
```

SELECT INTO OUTFILE S3

Con la instrucción `SELECT INTO OUTFILE S3` puede consultar datos de un clúster de base de datos y guardarlos directamente en archivos de texto delimitado almacenados en un bucket de Amazon S3.

No se admiten archivos comprimidos. Los archivos cifrados son compatibles a partir de Aurora MySQL versión 2.09.0.

Sintaxis

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
```

```
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
        [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
         [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
    [export_options]
    [MANIFEST {ON | OFF}]
    [OVERWRITE {ON | OFF}]
    [ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['cmk_id']}]

export_options:
    [FORMAT {CSV|TEXT} [HEADER]]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
]
```

Parámetros

La instrucción `SELECT INTO OUTFILE S3` utiliza los siguientes parámetros obligatorios y opcionales que son específicos de Aurora.

s3-uri

Especifica el URI del prefijo de Amazon S3 utilizado. Utilice la sintaxis descrita en Especificación de una ruta a un bucket de Amazon S3.

FORMAT {CSV|TEXT} [HEADER]

Si lo desea, guarda los datos en formato CSV.

La opción `TEXT` es la predeterminada y produce el formato de exportación MySQL existente.

La opción `CSV` produce valores de datos separados por comas. El formato CSV sigue la especificación de RFC-4180. Si especifique la palabra clave opcional `HEADER`, el archivo de salida contiene una línea de encabezado. Las etiquetas de la línea de encabezado se corresponden con los nombres de las columnas de la instrucción `SELECT`. Puede usar los archivos CSV para el entrenamiento de modelos de datos a fin de utilizarlos con los servicios de ML de AWS. Para obtener más información acerca del uso de datos exportados de Aurora con servicios de ML de AWS, consulte Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado).

MANIFEST {ON | OFF}

Indica si se crea o no un archivo de manifiesto en Amazon S3. El archivo de manifiesto es un archivo en notación de objetos JavaScript (JSON) que puede usarse para cargar datos en un clúster de base de datos Aurora con la instrucción `LOAD DATA FROM S3 MANIFEST`. Para obtener más información acerca de `LOAD DATA FROM S3 MANIFEST`, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.

Si se especifica `MANIFEST ON` en la consulta, se creará el archivo de manifiesto en Amazon S3 después de haber creado y cargado todos los archivos de datos. El archivo de manifiesto se crea en la ruta siguiente:

```
s3-region://bucket-name/file-prefix.manifest
```

Para obtener más información acerca del formato del contenido del archivo de manifiesto, consulte Creación de un manifiesto con una lista de los archivos de datos.

OVERWRITE {ON | OFF}

Indica si los archivos existentes en el bucket de Amazon S3 especificado se sobrescriben. Si se especifica `OVERWRITE ON`, se sobrescribirán los archivos existentes con el prefijo indicado en el URI especificado en `s3-uri`. En caso contrario, se produce un error.

ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['*cmk_id*']}

Indica si se usa el cifrado del servidor con claves administradas de Amazon S3 (SSE-S3) o AWS KMS keys (SSE-KMS, incluidas Claves administradas por AWS y claves administradas por el cliente). La configuración de `SSE_S3` y `SSE_KMS` está disponible en la versión 3.05 de Aurora MySQL y versiones posteriores.

También puede utilizar la variable de sesión `aurora_select_into_s3_encryption_default` en lugar de la cláusula `ENCRYPTION`, como se muestra en el siguiente ejemplo. Utilice la cláusula SQL o la variable de sesión, pero no ambas.

```
set session set session aurora_select_into_s3_encryption_default={ON | OFF | SSE_S3
 | SSE_KMS};
```

La configuración de `SSE_S3` y `SSE_KMS` está disponible en la versión 3.05 de Aurora MySQL y versiones posteriores.

Cuando se establece `aurora_select_into_s3_encryption_default` en el siguiente valor:

- `OFF`: se sigue la política de cifrado predeterminada del bucket de S3. El valor predeterminado de `aurora_select_into_s3_encryption_default` es `OFF`.
- `ON` o:`SSE_S3` el objeto S3 se cifra mediante claves administradas de Amazon S3 (SSE-S3).
- `SSE_KMS`: el objeto S3 se cifra mediante una AWS KMS key.

  En este caso, también se incluye la variable de sesión `aurora_s3_default_cmk_id`, por ejemplo:

  ```
  set session aurora_select_into_s3_encryption_default={SSE_KMS};
  set session aurora_s3_default_cmk_id={NULL | 'cmk_id'};
  ```

  - Cuando `aurora_s3_default_cmk_id` es `NULL`, el objeto S3 se cifra mediante una Clave administrada de AWS.
  - Cuando `aurora_s3_default_cmk_id` es `cmk_id` de una cadena que no está vacía, el objeto S3 se cifra mediante una clave gestionada por el cliente.

    El valor de `cmk_id` no puede ser una cadena vacía.

Si se utiliza el comando `SELECT INTO OUTFILE S3`, Aurora determina el cifrado de la siguiente manera:

- Si la cláusula `ENCRYPTION` está presente en el comando SQL, Aurora se basa únicamente en el valor de `ENCRYPTION` y no usa una variable de sesión.
- Si la cláusula `ENCRYPTION` no está presente, Aurora se basa en el valor de la variable de sesión.

Para obtener más información, consulte Uso del cifrado del servidor con claves administradas por Amazon S3 (SSE-S3) y Uso del cifrado del lado del servidor con claves AWS KMS (SSE-KMS) en la Guía del usuario de Amazon Simple Storage Service.

Encontrará más información sobre otros parámetros en SELECT statement (Instrucción SELECT) y LOAD DATA statement (Instrucción LOAD DATA) en la documentación de MySQL.

Consideraciones

El número de archivos escritos en el bucket de Amazon S3 depende de la cantidad de datos que seleccione la instrucción `SELECT INTO OUTFILE S3` y del umbral de tamaño de archivo de Aurora MySQL. Por defecto, el umbral de tamaño de archivo es 6 gigabytes (GB). Si el volumen de los datos seleccionados por la instrucción es inferior al umbral de tamaño de archivo, se creará un solo archivo. En caso contrario, se crearán varios. Las siguientes son otras consideraciones sobre los archivos que crea esta instrucción:

- Aurora MySQL garantiza que las filas de los archivos de datos no se dividan entre archivos. Si hay varios archivos, normalmente el tamaño de cada uno, salvo el último, será cercano al umbral de tamaño de archivo. Sin embargo, en ocasiones, mantenerse por debajo del umbral de tamaño de archivo implicaría dividir una fila entre dos archivos. En tal caso, Aurora MySQL crea un archivo de datos que mantiene la fila completa, pero que puede tener un tamaño superior al umbral.
- Dado que cada instrucción `SELECT` en Aurora MySQL ejecuta como transacción atómica, una instrucción `SELECT INTO OUTFILE S3` que selecciona un conjunto de datos grande puede ejecutarse durante un tiempo considerable. Si la instrucción devuelve un error por cualquier motivo, puede ser necesario comenzar desde el principio y ejecutarla de nuevo. Sin embargo, si la instrucción falla, los archivos ya cargados en Amazon S3 permanecerán en el bucket de Amazon S3 especificado. Entonces puede utilizar otra instrucción para cargar los datos faltantes en lugar de comenzar de nuevo desde el principio.
- Si el volumen de datos que se seleccionan es grande (más de 25 GB), es recomendable usar varias instrucciones `SELECT INTO OUTFILE S3` para guardar los datos en Amazon S3. Cada instrucción debe seleccionar una parte distinta de los datos que se deben guardar y también debe especificar un valor `file_prefix` distinto en el parámetro `s3-uri` al guardar los archivos de datos. La partición de los datos que se van a seleccionar con varias instrucciones facilita la recuperación de un error en una instrucción. Si se produce un error para una instrucción, solo se debe volver a seleccionar una parte de los datos y cargarlos en Amazon S3. El uso de múltiples instrucciones también ayuda a evitar una transacción única prolongada, lo que puede mejorar el desempeño.

- Si se ejecutan varias instrucciones `SELECT INTO OUTFILE S3` en paralelo con el mismo valor de `file_prefix` en el parámetro `s3-uri` para seleccionar datos y cargarlos en Amazon S3, el comportamiento resultante no está definido.
- Aurora MySQL no carga en Amazon S3 los metadatos, como el esquema de tablas y los metadatos de archivos.
- En algunos casos puede ser necesario volver a ejecutar una consulta `SELECT INTO OUTFILE S3`, por ejemplo para recuperarse de un error. En estos casos, deberá eliminar los archivos de datos que haya en el bucket de Amazon S3 con el prefijo especificado en `s3-uri` o bien deberá especificar `OVERWRITE ON` en la consulta `SELECT INTO OUTFILE S3`.

La instrucción `SELECT INTO OUTFILE S3` devuelve el número de error y la respuesta habituales de MySQL en caso de éxito o error de la operación. Si no tiene acceso al número de error y la respuesta de MySQL, la forma más sencilla de determinar cuándo ha terminado la operación es especificar `MANIFEST ON` en la instrucción. El archivo de manifiesto es el último archivo que escribe la instrucción. En otras palabras, si existe un archivo de manifiesto, la instrucción se ha completado.

Actualmente no existe un modo de monitorear el progreso de la instrucción `SELECT INTO OUTFILE S3` mientras se ejecuta. Sin embargo, supongamos que va a cargar una gran cantidad de datos de Aurora MySQL en Amazon S3 con esta instrucción y que conoce el volumen de los datos seleccionados. En tal caso puede estimar el progreso observando la creación de los archivos de datos en Amazon S3.

Para ello partimos del hecho de que se crea un archivo de datos en el bucket de Amazon S3 aproximadamente por cada 6 GB de datos seleccionados por la instrucción. Dividiendo el volumen de los datos seleccionados entre 6 GB se obtiene el número estimado de archivos de datos que se crean. Puede estimar el progreso de la instrucción monitoreando el número de archivos cargados en Amazon S3 mientras se ejecuta la instrucción.

Ejemplos

La instrucción siguiente selecciona todos los datos de la tabla `employees` y los guarda en un bucket de Amazon S3 situado en una región distinta de la del clúster de base de datos Aurora MySQL. La instrucción crea archivos de datos en los que cada campo termina con un carácter coma (`,`) y cada fila termina con un carácter de nueva línea (`\n`). La instrucción devuelve un error si en el bucket de Amazon S3 especificado ya existen archivos con el prefijo especificado en `sample_employee_data`.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

La instrucción siguiente selecciona todos los datos de la tabla `employees` y los guarda en un bucket de Amazon S3 situado en la misma región que el clúster de base de datos Aurora MySQL. La instrucción crea archivos de datos en los que cada campo termina con un carácter coma (`,`) y cada fila termina con un carácter de nueva línea (\n), y también crea un archivo de manifiesto. La instrucción devuelve un error si en el bucket de Amazon S3 especificado ya existen archivos con el prefijo especificado en `sample_employee_data`.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON;
```

La instrucción siguiente selecciona todos los datos de la tabla `employees` y los guarda en un bucket de Amazon S3 situado en una región distinta de la del clúster de base de datos Aurora. La instrucción crea archivos de datos en los que cada campo termina con un carácter coma (`,`) y cada fila termina con un carácter de nueva línea (\n). La instrucción sobrescribe los archivos que puedan existir en el bucket de `sample_employee_data` con el prefijo especificado en Amazon S3.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    OVERWRITE ON;
```

La instrucción siguiente selecciona todos los datos de la tabla `employees` y los guarda en un bucket de Amazon S3 situado en la misma región que el clúster de base de datos Aurora MySQL. La instrucción crea archivos de datos en los que cada campo termina con un carácter coma (`,`) y cada fila termina con un carácter de nueva línea (\n), y también crea un archivo de manifiesto. La instrucción sobrescribe los archivos que puedan existir en el bucket de `sample_employee_data` con el prefijo especificado en Amazon S3.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
```

```
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON
    OVERWRITE ON;
```

Invocación de una función de Lambda desde un clúster de base de datos de Amazon Aurora MySQL

Puede invocar una función de AWS Lambda desde un clúster de base de datos de Amazon Aurora Edición compatible con MySQL con una función nativa `lambda_sync` o `lambda_async`. Antes de invocar una función Lambda desde Aurora MySQL, el clúster de base de datos Aurora debe tener acceso a Lambda. Para obtener más información sobre cómo conceder acceso a Aurora MySQL, consulte Otorgar acceso a Aurora a Lambda. Para obtener información sobre las funciones almacenadas de `lambda_sync` y `lambda_async`, consulte Invocación de una función de Lambda con una función nativa de Aurora MySQL.

También puede llamar a una función AWS Lambda mediante un procedimiento almacenado. Sin embargo, el uso de un procedimiento almacenado está obsoleto. Recomendamos encarecidamente el uso de una función nativa de Aurora MySQL si utiliza una de las siguientes versiones de Aurora MySQL:

- Versión 2 de Aurora MySQL para clústeres compatibles con MySQL 5.7.
- Versión 3.01 y posterior de Aurora MySQL, para clústeres compatibles con MySQL 8.0. En la versión 3 de Aurora MySQL, el procedimiento almacenado no está disponible.

Para obtener información sobre cómo darle a Aurora acceso a Lambda y sobre la invocación de una función de Lambda, consulte los siguientes temas.

Temas

- Otorgar acceso a Aurora a Lambda
- Invocación de una función de Lambda con una función nativa de Aurora MySQL
- Invocación de una función de Lambda con un procedimiento almacenado de Aurora MySQL (obsoleto)

Otorgar acceso a Aurora a Lambda

Para poder invocar funciones Lambda desde un clúster de base de datos de Aurora MySQL, primero debe dar al clúster permiso para que obtenga acceso a Lambda.

Para conceder a Aurora MySQL acceso a Lambda

1. Cree una política de AWS Identity and Access Management (IAM) que asigne los permisos que permiten que su clúster de base de datos de Aurora MySQL invoque las funciones de Lambda. Para obtener instrucciones, consulte Creación de una política de IAM para acceder a los recursos de AWS Lambda.
2. Cree un rol de IAM y asocie la política de IAM que creó en Creación de una política de IAM para acceder a los recursos de AWS Lambda al nuevo rol de IAM. Para obtener instrucciones, consulte Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS.
3. Configure el parámetro de clúster de base de datos `aws_default_lambda_role` con el nombre de recurso de Amazon (ARN) del nuevo rol de IAM.

   Si el clúster es parte de una base de datos global de Aurora, aplique la misma configuración a cada clúster de Aurora en la base de datos global.

   Para obtener más información acerca de los parámetros de clúster de base de datos, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos.
4. Para permitir la invocación de funciones Aurora MySQL a los usuarios de base de datos un clúster de base de datos Lambda, asocie el rol que creó en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con el clúster. Para obtener información acerca de cómo asociar un rol de IAM con un clúster de base de datos, consulte Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.

   Si el clúster es parte de una base de datos global de Aurora, asocie el rol con cada clúster de Aurora en la base de datos global.
5. Configure el clúster de base de datos Aurora MySQL para permitir conexiones salientes hacia Lambda. Para obtener instrucciones, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

   Si el clúster es parte de una base de datos global de Aurora, habilite las conexiones de salida para cada clúster de Aurora en la base de datos global.

Invocación de una función de Lambda con una función nativa de Aurora MySQL

> **Note**
>
> Puede llamar a las funciones nativas `lambda_sync` y `lambda_async` cuando utilice la versión 2 de Aurora MySQL o la versión 3.01 y posteriores de Aurora MySQL. Para obtener más información acerca de las versiones de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

Para invocar una función de AWS Lambda desde un clúster de base de datos de Aurora MySQL, llame a las funciones nativas de `lambda_sync` y`lambda_async`. Este método puede ser útil cuando se desea integrar una base de datos que se ejecuta en Aurora MySQL con otros servicios de AWS. Por ejemplo, es posible que desee enviar una notificación mediante Amazon Simple Notification Service (Amazon SNS) siempre que se inserte una fila en una tabla específica de una base de datos.

Contenido

- Trabajo con funciones nativas para invocar una función de Lambda
  - Concesión de roles en Aurora MySQL versión 3
  - Concesión de privilegios en la versión 2 de Aurora MySQL
  - Sintaxis de la función lambda_sync
  - Parámetros de la función lambda_sync
  - Ejemplo de la función lambda_sync
  - Sintaxis de la función lambda_async
  - Parámetros de la función lambda_async
  - Ejemplo de la función lambda_async
  - Invocación de una función de Lambda en un desencadenador

Trabajo con funciones nativas para invocar una función de Lambda

Las funciones `lambda_sync` y `lambda_async` son funciones nativas integradas que invocan una función de Lambda de forma síncrona o asíncrona. Cuando necesite conocer el resultado de la función de Lambda antes de pasar a otra acción, utilice la función síncrona `lambda_sync`. Cuando

no necesite conocer el resultado de la función de Lambda antes de pasar a otra acción, use la función asíncrona `lambda_async`.

Concesión de roles en Aurora MySQL versión 3

En la versión 3 de Aurora MySQL, al usuario que invoca una función nativa se le debe conceder el rol de `AWS_LAMBDA_ACCESS`. Para conceder este rol a un usuario, conéctese a la instancia de base de datos como usuario administrativo y ejecute la siguiente instrucción.

```
GRANT AWS_LAMBDA_ACCESS TO user@domain-or-ip-address
```

Puede revocar este rol ejecutando la siguiente instrucción.

```
REVOKE AWS_LAMBDA_ACCESS FROM user@domain-or-ip-address
```

> **Tip**
>
> Cuando utiliza la técnica de rol en Aurora MySQL versión 3, también puede activar el rol mediante la instrucción `SET ROLE role_name` o `SET ROLE ALL`. Si no está familiarizado con el sistema de roles MySQL 8.0, puede obtener más información en Modelo de privilegios basado en roles. Para obtener más información, consulte Using roles en el Manual de referencia de MySQL.
>
> Esto solo se aplica a la sesión activa actual. Cuando se vuelva a conectar, debe ejecutar la instrucción `SET ROLE` de nuevo para conceder privilegios. Para obtener más información, consulte la instrucción SET ROLE en el Manual de referencia de MySQL.
>
> Puede utilizar el parámetro de clúster de base de datos `activate_all_roles_on_login` para activar automáticamente todos los roles cuando un usuario se conecta a una instancia de base de datos. Cuando se establece este parámetro, por lo general no tiene que llamar a la instrucción `SET ROLE` para activar un rol. Para obtener más información, consulte activate_all_roles_on_login en el Manual de referencia de MySQL.
>
> Sin embargo, debe llamar a `SET ROLE ALL` de forma explícita al principio de un procedimiento almacenado para activar el rol cuando un usuario diferente llame al procedimiento almacenado.

Si recibe un error como el siguiente al intentar invocar una función de Lambda, ejecute una instrucción `SET ROLE`.

```
SQL Error [1227] [42000]: Access denied; you need (at least one of) the Invoke Lambda
 privilege(s) for this operation
```

Asegúrese de conceder el rol al usuario correcto, tal como se muestra en las entradas de la tabla `mysql.users`. Puede que haya varios usuarios con el mismo nombre, pero en hosts diferentes. Según la aplicación o el host que invoque la función `lambda_sync`, MySQL selecciona el usuario con la mejor coincidencia según las entradas de las columnas `host`.

Concesión de privilegios en la versión 2 de Aurora MySQL

En la versión 2 de Aurora MySQL, al usuario que invoca una función nativa se le debe conceder el privilegio de `INVOKE LAMBDA`. Para conceder este privilegio a un usuario, conéctese a la instancia de base de datos como usuario administrativo y ejecute la siguiente instrucción.

```
GRANT INVOKE LAMBDA ON *.* TO user@domain-or-ip-address
```

Puede revocar este privilegio ejecutando la siguiente instrucción.

```
REVOKE INVOKE LAMBDA ON *.* FROM user@domain-or-ip-address
```

Sintaxis de la función lambda_sync

Puede invocar la función `lambda_sync` de forma síncrona con el tipo de invocación `RequestResponse`. La función devuelve el resultado de la invocación de Lambda en una carga JSON. La función presenta la siguiente sintaxis.

```
lambda_sync (
  lambda_function_ARN,
  JSON_payload
)
```

Parámetros de la función lambda_sync

La función `lambda_sync` tiene los siguientes parámetros.

lambda_function_ARN

El Nombre de recurso de Amazon (ARN) de la función Lambda que se va a invocar.

JSON_payload

La carga de la función de Lambda invocada, en formato JSON.

> **Note**
>
> Aurora MySQL versión 3 admite las funciones de análisis JSON de MySQL 8.0. Sin embargo, la versión 2 de Aurora MySQL no incluye esas funciones. El análisis de JSON no es necesario si una función de Lambda devuelve un valor atómico, como un número o una cadena.

Ejemplo de la función lambda_sync

La siguiente consulta basada en `lambda_sync` invoca la función de Lambda `BasicTestLambda` de forma síncrona mediante la función ARN. La carga de la función es `{"operation": "ping"}`.

```
SELECT lambda_sync(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Sintaxis de la función lambda_async

Puede invocar la función `lambda_async` de forma asíncrona con el tipo de invocación `Event`. La función devuelve el resultado de la invocación de Lambda en una carga JSON. La función presenta la siguiente sintaxis.

```
lambda_async (
  lambda_function_ARN,
  JSON_payload
)
```

Parámetros de la función lambda_async

La función `lambda_async` tiene los siguientes parámetros.

lambda_function_ARN

El Nombre de recurso de Amazon (ARN) de la función Lambda que se va a invocar.

JSON_payload

La carga de la función de Lambda invocada, en formato JSON.

> **Note**
>
> Aurora MySQL versión 3 admite las funciones de análisis JSON de MySQL 8.0. Sin embargo, la versión 2 de Aurora MySQL no incluye esas funciones. El análisis de JSON no es necesario si una función de Lambda devuelve un valor atómico, como un número o una cadena.

Ejemplo de la función lambda_async

La siguiente consulta basada en `lambda_async` invoca la función de Lambda `BasicTestLambda` de forma asíncrona mediante la función ARN. La carga de la función es `{"operation": "ping"}`.

```
SELECT lambda_async(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Invocación de una función de Lambda en un desencadenador

Puede usar disparadores para llamar Lambda en instrucciones de modificación de datos. El siguiente ejemplo usa la función `lambda_async` nativa y almacena el resultado en una variable.

```
mysql>SET @result=0;
mysql>DELIMITER //
mysql>CREATE TRIGGER myFirstTrigger
      AFTER INSERT
          ON Test_trigger FOR EACH ROW
      BEGIN
      SELECT lambda_async(
          'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
          '{"operation": "ping"}')
          INTO @result;
      END; //
mysql>DELIMITER ;
```

> **Note**
>
> Los desencadenadores no se ejecutan una vez por instrucción SQL, sino una vez por fila modificada, de una en una. Cuando se ejecuta un desencadenador, el proceso es sincrónico. La instrucción de modificación de datos sólo devuelve cuando se completa el desencadenador.
> Tenga cuidado al invocar una función de AWS Lambda desde desencadenadores en tablas que experimentan un alto tráfico de escritura. Los desencadenadores `INSERT`, `UPDATE` y `DELETE` se activan por fila. Una gran carga de trabajo de escritura en una tabla con desencadenadores `INSERT`, `UPDATE` o `DELETE` genera un gran número de llamadas a la función AWS Lambda.

Invocación de una función de Lambda con un procedimiento almacenado de Aurora MySQL (obsoleto)

Para invocar una función de AWS Lambda desde un clúster de base de datos de Aurora MySQL, llame al procedimiento `mysql.lambda_async`. Este método puede ser útil cuando se desea integrar una base de datos que se ejecuta en Aurora MySQL con otros servicios de AWS. Por ejemplo, es posible que desee enviar una notificación mediante Amazon Simple Notification Service (Amazon SNS) siempre que se inserte una fila en una tabla específica de una base de datos.

Contenido

- Consideraciones sobre la versión de Aurora MySQL
- Uso del procedimiento mysql.lambda_async para invocar una función de Lambda (obsoleto)
  - Sintaxis
  - Parámetros
  - Ejemplos

Consideraciones sobre la versión de Aurora MySQL

A partir de la versión 2 de Aurora MySQL, puede utilizar el método de función nativa en lugar de estos procedimientos almacenados para invocar una función de Lambda. Para obtener más información acerca de las funciones nativas, consulte Trabajo con funciones nativas para invocar una función de Lambda.

En la versión 2 de Aurora MySQL, el procedimiento almacenado `mysql.lambda_async` ya no está disponible. Recomendamos que, en su lugar, utilice funciones de Lambda nativas.

En Aurora MySQL versión 3, el procedimiento almacenado no está disponible.

Uso del procedimiento mysql.lambda_async para invocar una función de Lambda (obsoleto)

El procedimiento `mysql.lambda_async` es un procedimiento almacenado integrado que invoca una función Lambda de forma asíncrona. Para utilizar este procedimiento, el usuario de la base de datos debe tener privilegios `EXECUTE` para el procedimiento almacenado `mysql.lambda_async`.

Sintaxis

El procedimiento `mysql.lambda_async` tiene la siguiente sintaxis.

```
CALL mysql.lambda_async (
  lambda_function_ARN,
  lambda_function_input
)
```

Parámetros

El procedimiento `mysql.lambda_async` tiene los siguientes parámetros.

lambda_function_ARN

El Nombre de recurso de Amazon (ARN) de la función Lambda que se va a invocar.

lambda_function_input

La cadena de entrada, en formato JSON, para la función Lambda invocada.

Ejemplos

Como práctica recomendada, es conveniente que encapsule las llamadas al procedimiento `mysql.lambda_async` en un procedimiento almacenado que se pueda invocar desde distintos orígenes, como los disparadores o el código del cliente. Esto puede ayudar a evitar los problemas de "discrepancia de impedancia" y hacer que resulte más sencillo llamar a las funciones Lambda.

> **Note**
>
> Tenga cuidado al invocar una función de AWS Lambda desde desencadenadores en tablas que experimentan un alto tráfico de escritura. Los desencadenadores `INSERT`, `UPDATE`

y DELETE se activan por fila. Una gran carga de trabajo de escritura en una tabla con desencadenadores INSERT, UPDATE o DELETE genera un gran número de llamadas a la función AWS Lambda.
Aunque las llamadas al procedimiento mysql.lambda_async son asíncronas, los disparadores son síncronos. Una instrucción que da como resultado un gran número de activaciones de disparadores no espera a que finalice la llamada a la función de AWS Lambda, pero espera a que finalicen los disparadores antes de devolver el control al cliente.

Example Ejemplo: Invocar una función de AWS Lambda para enviar correo electrónico

En el siguiente ejemplo se crea un procedimiento almacenado que puede llamarse desde el código de una base de datos para enviar un mensaje de correo electrónico utilizando una función Lambda.

AWS Lambda Función

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },

        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Procedimiento almacenado

```
DROP PROCEDURE IF EXISTS SES_send_email;
DELIMITER ;;
  CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                  IN email_to VARCHAR(255),
                                  IN subject VARCHAR(255),
                                  IN body TEXT) LANGUAGE SQL
  BEGIN
    CALL mysql.lambda_async(
         'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
         CONCAT('{"email_to" : "', email_to,
             '", "email_from" : "', email_from,
             '", "email_subject" : "', subject,
             '", "email_body" : "', body, '"}')
     );
  END
  ;;
DELIMITER ;
```

Llamada al procedimiento almacenado para invocar la función de AWS Lambda

```
mysql> call SES_send_email('example_from@amazon.com', 'example_to@amazon.com', 'Email
 subject', 'Email content');
```

Example Ejemplo: Invocar una función de AWS Lambda para publicar un evento procedente de un desencadenador

En el siguiente ejemplo se crea un procedimiento almacenado que publica un evento mediante Amazon SNS. El código llama al procedimiento desde un disparador cuando se añade una fila a una tabla.

AWS Lambda Función

```
import boto3

sns = boto3.client('sns')

def SNS_publish_message(event, context):

    return sns.publish(
```

```
        TopicArn='arn:aws:sns:us-west-2:123456789012:Sample_Topic',
        Message=event['message'],
        Subject=event['subject'],
        MessageStructure='string'
    )
```

Procedimiento almacenado

```
DROP PROCEDURE IF EXISTS SNS_Publish_Message;
DELIMITER ;;
CREATE PROCEDURE SNS_Publish_Message (IN subject VARCHAR(255),
                                      IN message TEXT) LANGUAGE SQL
BEGIN
  CALL mysql.lambda_async('arn:aws:lambda:us-
west-2:123456789012:function:SNS_publish_message',
     CONCAT('{ "subject" : "', subject,
            '", "message" : "', message, '" }')
     );
END
;;
DELIMITER ;
```

Tabla

```
CREATE TABLE 'Customer_Feedback' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'customer_name' varchar(255) NOT NULL,
  'customer_feedback' varchar(1024) NOT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Desencadenador

```
DELIMITER ;;
CREATE TRIGGER TR_Customer_Feedback_AI
  AFTER INSERT ON Customer_Feedback
  FOR EACH ROW
BEGIN
  SELECT CONCAT('New customer feedback from ', NEW.customer_name),
 NEW.customer_feedback INTO @subject, @feedback;
  CALL SNS_Publish_Message(@subject, @feedback);
END
```

```
;;
DELIMITER ;
```

Inserción de una fila en la tabla para desencadenar la notificación

```
mysql> insert into Customer_Feedback (customer_name, customer_feedback) VALUES ('Sample
 Customer', 'Good job guys!');
```

Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs

Puede configurar su clúster de base de datos Aurora MySQL para publicar datos de registros generales, lentos, de auditoría y de errores en un grupo de registros en Amazon CloudWatch Logs. Con CloudWatch Logs, puede realizar análisis en tiempo real de los datos de registro y utilizar CloudWatch para crear alarmas y ver métricas. Puede utilizar CloudWatch Logs para almacenar los registros de registros en almacenamiento de larga duración.

Para publicar logs en CloudWatch Logs, se deben habilitar los logs respectivos. Los logs de errores están habilitados de forma predeterminada, pero debe habilitar explícitamente los demás tipos de logs. Para obtener información acerca de habilitar registros en MySQL, consulte Selecting General Query and Slow Query Log Output Destinations en la documentación de MySQL. Para obtener más información acerca de cómo habilitar los registros de auditoría de Aurora MySQL, consulte Habilitar la auditoría avanzada.

> **Note**
>
> - Si la exportación de datos de log está deshabilitada, Aurora no elimina los grupos de logs o flujos de logs. Si la exportación de datos de log está deshabilitada, los datos de log de existentes seguirán disponibles en CloudWatch Logs, en función de la retención de logs y seguirá incurriendo en gastos por los datos de log de auditoría almacenados. Puede eliminar los flujos y los grupos de registros en la consola de CloudWatch Logs, la AWS CLI o la API de CloudWatch Logs.
> - Una forma alternativa de publicar los registros de auditoría en CloudWatch Logs consiste en habilitar la auditoría avanzada, configurar un grupo de parámetros de clúster de base de datos personalizado y establecer el parámetro `server_audit_logs_upload` como 1. El valor predeterminado para el parámetro del clúster de base de datos `server_audit_logs_upload` es `0`. Para obtener información sobre cómo habilitar la

auditoría avanzada, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.

Si usa este método alternativo, debe tener un rol de IAM para obtener acceso a CloudWatch Logs y establecer el parámetro de nivel de clúster `aws_default_logs_role` en el ARN para esta función. Para obtener información acerca de la creación del rol de , consulte Configuración de roles de IAM para acceder a servicios de AWS. Sin embargo, si tiene el rol vinculado alservicio `AWSServiceRoleForRDS`, proporciona acceso a CloudWatch Logs y anula cualquier función definida por el usuario. Para obtener información acerca de los roles vinculados al servicio para Amazon RDS, consulte Uso de roles vinculados a servicios de Amazon Aurora.

- Si no desea exportar logs de auditoría a CloudWatch Logs, asegúrese de que todos los métodos de exportación de logs de auditoría estén deshabilitados. Estos métodos son la Consola de administración de AWS, la AWS CLI, la API de RDS y el parámetro `server_audit_logs_upload`.
- El procedimiento es ligeramente distinto para clústeres de bases de datos de Aurora Serverless v1 que para clústeres de bases de datos con instancias aprovisionadas o instancias de bases de datos de Aurora Serverless v2. Los clústeres de Aurora Serverless v1 cargan automáticamente todos los tipos de registros que habilita a través de los parámetros de configuración.

  Por lo tanto, activa o desactiva la carga de registros para clústeres de bases de datos de Aurora Serverless v1 mediante la activación y desactivación de diferentes tipos de registros en el grupo de parámetros de clústeres de base de datos. No modifique la configuración del propio clúster a través de la Consola de administración de AWS, la AWS CLI o la API de RDS. Para obtener información sobre cómo activar y desactivar los registros de MySQL para clústeres de Aurora Serverless v1, consulte Grupos de parámetros de Aurora Serverless v1.

Consola

Puede publicar registros de Aurora MySQL para clústeres aprovisionados en CloudWatch Logs con la consola.

Para publicar logs de Aurora MySQL desde la consola

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos de Aurora MySQL para el que desee publicar los datos de registro.
4. Elija Modify.
5. En la sección Logs exports (Exportaciones de registros), elija los registros que desea comenzar a publicar en CloudWatch Logs.
6. Elija Continue (Continuar), seguido de Modify DB Cluster (Modificar clúster de base de datos) en la página de resumen.

AWS CLI

Puede publicar registros de Aurora MySQL para clústeres aprovisionados con la CLI de AWS. Para ello, ejecute el comando modify-db-cluster de la AWS CLI con las siguientes opciones:

- `--db-cluster-identifier`: identificador de clúster de base de datos.
- `--cloudwatch-logs-export-configuration`: el ajuste de configuración para los tipos de registros que habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

También puede publicar registros de Aurora MySQL; para ello, ejecute uno de los siguientes comandos de la AWS CLI:

- create-db-cluster
- restore-db-clúster-from-s3
- restore-db-clúster-from-snapshot
- restore-db-clúster-to-point-in-time

Ejecute uno de estos comandos de la AWS CLI con las siguientes opciones:

- `--db-cluster-identifier`: identificador de clúster de base de datos.
- `--engine`: el motor de base de datos.
- `--enable-cloudwatch-logs-exports`: el ajuste de configuración para los tipos de registros que habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

Podrían ser necesarias otras opciones en función del comando de la AWS CLI que se ejecute.

Example

El siguiente comando modifica un clúster de base de datos Aurora MySQL existente para publicar archivos de registro en CloudWatch Logs.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Example

El siguiente comando crea un clúster de base de datos Aurora MySQL para publicar archivos de registro en CloudWatch Logs.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine aurora \
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

Para Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine aurora ^
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

API de RDS

Puede publicar registros de Aurora MySQL para clústeres aprovisionados con la API de RDS. Para ello, ejecute la operación ModifyDBCluster con las siguientes opciones:

- `DBClusterIdentifier`: identificador de clúster de base de datos.
- `CloudwatchLogsExportConfiguration`: el ajuste de configuración para los tipos de registros que habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

También puede publicar logs de Aurora MySQL con la API de RDS ejecutando una de las siguientes operaciones de API de RDS:

- CreateDBCluster
- RestoreDBclústerFromS3
- RestoreDBclústerFromSnapshot
- RestoreDBclústerToPointInTime

Ejecute la operación de la API de RDS con los siguientes parámetros:

- `DBClusterIdentifier`: identificador de clúster de base de datos.
- `Engine`: el motor de base de datos.
- `EnableCloudwatchLogsExports`: el ajuste de configuración para los tipos de registros que habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

Podrían ser necesarios otros parámetros en función del comando de la AWS CLI que ejecute.

Monitoreo de eventos de registro en Amazon CloudWatch

Después de habilitar los eventos de registro de Aurora MySQL, puede monitorizar los eventos en Amazon CloudWatch Logs. Se crea un nuevo grupo de registros automáticamente para el clúster de base de datos Aurora con el siguiente prefijo, donde *`cluster-name`* representa el nombre del clúster de base de datos y *`log_type`* representa el tipo de registro.

```
/aws/rds/cluster/cluster-name/log_type
```

Por ejemplo, si configura la función de exportación para que incluya el log de consultas lentas para un clúster de base de datos denominado `mydbcluster`, los datos de consultas lentas se almacenan en el grupo de logs `/aws/rds/cluster/mydbcluster/slowquery`.

Todos los eventos de todas las instancias de base de datos en un clúster de base de datos se envían a un grupo de logs utilizando flujos de registros distintos. El comportamiento depende de cuál de las siguientes condiciones se da:

- Existe un grupo de registros con el nombre especificado.

  Aurora utiliza el grupo de registros existente para exportar los datos de registros para el clúster. Puede utilizar la configuración automática, como AWS CloudFormation, para crear grupos de logs con periodos de retención de registros predefinidos, filtros de métricas y acceso de cliente.
- No existe un grupo de registros con el nombre especificado.

  Cuando se detecta una entrada de registro coincidente en el archivo de registros de la instancia, Aurora MySQL crea automáticamente un nuevo grupo de registros en CloudWatch Logs. El grupo de registros utiliza el período de retención de registros predeterminado de Never Expire (Nunca caduca).

  Use la consola de CloudWatch Logs, la AWS CLI o la API de CloudWatch Logs para modificar el periodo de retención de registros. Para obtener más información sobre cómo cambiar los periodos de retención de registro en CloudWatch Logs, consulte Cambiar la retención de datos de registro en CloudWatch Logs.

Use la consola de CloudWatch Logs, la AWS CLI o la API de CloudWatch Logs para buscar información en los eventos de registro para un clúster de base de datos. Para obtener más información sobre la búsqueda y el filtrado de datos de registro, consulte Búsqueda y filtrado de datos de registros.

Modo lab de Amazon Aurora MySQL

> **Important**
>
> El modo lab se ha ingresado en la versión 2 de Aurora MySQL para habilitar la optimización de DDL rápida, lo que ha mejorado la eficiencia de determinadas operaciones de DDL. En la versión 3 de Aurora MySQL, se ha eliminado el modo lab y DDL rápida se ha sustituido por la característica de MySQL 8.0 denominada DDL instantáneo.

El modo lab de Aurora se utiliza para habilitar características de Aurora que están disponibles en la actual versión de base de datos Aurora, pero que no están habilitadas de manera predeterminada. Aunque no se recomienda el uso de características del modo lab de Aurora en los clústeres de bases de datos de producción, puede utilizar el modo lab de Aurora para habilitar estas características para los clústeres de base de datos sus entornos de desarrollo y prueba. Para obtener más información sobre las características de Aurora que están disponibles cuando está habilitado el modo lab de Aurora, consulte Características del modo lab de Aurora.

El parámetro `aurora_lab_mode` es un parámetro en el nivel de la instancia que se encuentra en el grupo de parámetros predeterminado. El parámetro está establecido en `0` (deshabilitado) en el grupo de parámetros predeterminado. Para habilitar el modo lab de Aurora, cree un grupo de parámetros personalizado, establezca el parámetro `aurora_lab_mode` en `1` (habilitado) en el grupo de parámetros personalizado y modifique una o más instancias de base de datos en su clúster de Aurora para que utilice el grupo de parámetros personalizado. A continuación, conecte el punto de enlace de instancia adecuado para probar las características del modo lab. Para obtener información acerca de cómo modificar un grupo de parámetros de base de datos, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora. Para obtener información acerca de los grupos de parámetros y Amazon Aurora, consulte Parámetros de configuración de Aurora MySQL.

Características del modo lab de Aurora

La siguiente característica de Aurora está actualmente disponible cuando se habilita el modo de laboratorio de Aurora. Debe habilitar el modo lab de Aurora para poder utilizar estas características.

DDL rápida

Esta característica le permite ejecutar una operación `ALTER TABLE tbl_name ADD COLUMN col_name column_definition` casi al instante. La operación se completa sin que sea necesario copiar la tabla y sin que haya un impacto material en otras instrucciones DML. Como no consume almacenamiento temporal para una copia de la tabla, las instrucciones de DDL resultan prácticas incluso para tablas grandes en clases de instancias pequeñas.

El DDL rápido solo se admite actualmente para añadir columnas que se puedan anular, sin un valor predeterminado, al final de una tabla. Para obtener más información sobre cómo usar esta característica, consulte Modificación de las tablas de Amazon Aurora con DDL rápido.

Prácticas recomendadas con Amazon Aurora MySQL

En este tema se proporciona información acerca de las prácticas recomendadas y las opciones para usar o migrar datos en un clúster de base de datos de Amazon Aurora MySQL. La información de este tema resume y reitera algunas de las directrices y procedimientos que puede encontrar en Administración de un clúster de base de datos de Amazon Aurora.

Contenido

- Determinar a qué instancia de base de datos está conectado
- Prácticas recomendadas de escalado y rendimiento de Aurora MySQL
  - Utilización de clases de instancia T para el desarrollo y la prueba
  - Optimización de las consultas de combinación indexadas de Aurora MySQL con la captura previa de claves asíncronas
    - Habilitación de la captura previa de clave asíncrona
    - Optimización de consultas para la captura previa de clave asíncrona
  - Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash
    - Habilitación de las combinaciones hash
    - Optimización de consultas para combinaciones hash
  - Uso de Amazon Aurora para escalar las lecturas de una base de datos de MySQL
  - Optimización de las operaciones de marca temporal
  - Errores de desbordamiento de ID de índice virtual
- Prácticas recomendadas para obtener una elevada disponibilidad de Aurora MySQL
  - Uso de Amazon Aurora para la recuperación de desastres con sus bases de datos de MySQL
  - Migración de MySQL a Amazon Aurora MySQL con un tiempo de inactividad reducido
  - Prevención del rendimiento lento, el reinicio automático y la conmutación por error de las instancias de base de datos Aurora MySQL
- Recomendaciones para características de MySQL en Aurora MySQL
  - Uso de la replicación de varios subprocesos en Aurora MySQL
  - Invocación de funciones de AWS Lambda mediante funciones de MySQL nativas
  - Evitar las transacciones de XA con Amazon Aurora MySQL
  - Mantenimiento de las claves externas activadas durante las instrucciones DML
  - Configuración de la frecuencia de vaciado del búfer de registro

- Minimización y solución de problemas de los interbloqueos de Aurora MySQL
  - Minimización de interbloqueos de InnoDB
  - Supervisión de interbloqueos de InnoDB
- Evaluación de la utilización de instancias de base de datos para Aurora MySQL con métricas de Amazon CloudWatch

Determinar a qué instancia de base de datos está conectado

Puede determinar a qué instancia de base de datos un clúster de base de datos Aurora MySQL está conectado comprobando la variable global `innodb_read_only`, como se muestra en el siguiente ejemplo.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

La variable `innodb_read_only` se establece en `0N` si está conectado a una instancia de base de datos de lector. Este ajuste estará en `0FF` si está conectado a una instancia de base de datos de escritor, como, por ejemplo, la instancia principal de un clúster aprovisionado.

Este método puede resultar útil si se desea añadir lógica al código de la aplicación para equilibrar la carga de trabajo o para garantizar que una operación de escritura utilice la conexión correcta.

Prácticas recomendadas de escalado y rendimiento de Aurora MySQL

Aplique las prácticas recomendadas que se describen a continuación para mejorar el rendimiento y la escalabilidad de los clústeres de Aurora MySQL.

Temas

- Utilización de clases de instancia T para el desarrollo y la prueba
- Optimización de las consultas de combinación indexadas de Aurora MySQL con la captura previa de claves asíncronas
- Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash
- Uso de Amazon Aurora para escalar las lecturas de una base de datos de MySQL
- Optimización de las operaciones de marca temporal
- Errores de desbordamiento de ID de índice virtual

Utilización de clases de instancia T para el desarrollo y la prueba

Las instancias de Amazon Aurora MySQL que utilizan las clases de instancia de base de datos `db.t2`, `db.t3` o `db.t4g` son más adecuadas para las aplicaciones que no admiten una carga de trabajo elevada durante un periodo de tiempo largo. Las instancias T se han diseñado para ofrecer un desempeño de referencia moderado y la capacidad de poder ampliarlo a un nivel considerablemente superior si así lo exige la carga de trabajo. Están pensadas para las cargas de trabajo que no utilizan toda la CPU con frecuencia o de forma continua, pero que de vez en cuando necesitan ampliar sus procesos. Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Instancias de rendimiento ampliable.

Si el clúster de Aurora tiene más de 40 TB, no utilice las clases de instancia T. Cuando la base de datos tiene un gran volumen de datos, la sobrecarga de memoria para administrar objetos de esquema puede exceder la capacidad de la instancia T.

No habilite el esquema de rendimiento de MySQL en instancias T de Amazon Aurora MySQL. Si el esquema de desempeño está habilitado, la instancia T podría quedarse sin memoria.

> **Tip**
>
> Si la base de datos a veces está inactiva pero en otras ocasiones tiene una carga de trabajo sustancial, puede utilizar Aurora Serverless v2 como alternativa a las instancias T. Con Aurora Serverless v2, define un rango de capacidad y Aurora escala automáticamente la base de datos vertical y horizontalmente según la carga de trabajo actual. Para obtener más información sobre el uso, consulte Uso de Aurora Serverless v2. Para conocer las versiones del motor de base de datos que puede utilizar con Aurora Serverless v2, consulte Requisitos y limitaciones para Aurora Serverless v2.

Cuando utilice una instancia T como una instancia de base de datos en un clúster de base de datos de Aurora MySQL, recomendamos lo siguiente:

- Utilice la misma clase de instancia de base de datos para todas las instancias del clúster de base de datos. Por ejemplo, si utiliza `db.t2.medium` para su instancia de escritor, le recomendamos que utilice `db.t2.medium` también para las instancias de lector.

- No ajuste ninguna configuración relacionada con la memoria, como, por ejemplo, `innodb_buffer_pool_size`. Aurora utiliza un conjunto de valores predeterminados muy ajustados para búferes de memoria en las instancias T. Estos valores predeterminados especiales son necesarios para que Aurora se pueda ejecutar en instancias con restricción de memoria. Si cambia cualquier configuración relacionada con la memoria en una instancia T, es mucho más probable que encuentre condiciones de falta de memoria, incluso si el cambio está destinado a aumentar el tamaño del búfer.
- Monitorice el saldo de crédito de su CPU (`CPUCreditBalance`) para asegurarse de que está en un nivel sostenible. Es decir, que los créditos de la CPU se están acumulando a la misma velocidad a la que se usan.

  Cuando se agotan los créditos de CPU correspondientes a una instancia, se percibe una disminución inmediata en la CPU disponible y un incremento en la latencia de lectura y escritura de la instancia. Esta situación provoca una grave reducción del desempeño general de la instancia.

  Si el saldo de crédito de la CPU no está en un nivel sostenible, es recomendable que modifique la instancia de base de datos para que use una de las clases de instancia de base de datos R disponibles (escalado del cálculo).

  Para obtener más información acerca de las métricas de monitorización, consulte Consulta de métricas en la consola de Amazon RDS.
- Supervise el retardo de réplica (`AuroraReplicaLag`) entre la instancia de escritor y las instancias de lector.

  Si una instancia de lector se queda sin créditos de CPU antes de la instancia de escritor, el retraso resultante puede provocar que la instancia de lector se reinicie con frecuencia. Este resultado es habitual cuando una aplicación mantiene una carga elevada de operaciones de lectura distribuidas entre las instancias de lector al mismo tiempo que la instancia de escritor tiene una carga mínima de operaciones de escritura.

  Si ve un aumento sostenido del retardo de las réplicas, asegúrese de que no se está agotando el saldo de crédito de la CPU para las instancias de lector en su clúster de base de datos.

  Si el saldo de crédito de la CPU no está en un nivel sostenible, es recomendable que modifique la instancia de base de datos para que use una de las clases de instancia de base de datos R disponibles (escalado del cálculo).
- Mantenga el número de inserciones por transacción por debajo de 1 millón para los clústeres de base de datos que tengan habilitado el registro binario.

Si el grupo de parámetros de su clúster de base de datos tiene el parámetro `binlog_format` definido en un valor distinto de `OFF`, dicho clúster de base de datos puede experimentar condiciones de falta de memoria si recibe transacciones que contengan más de 1 millón de filas para insertar. Puede monitorizar la métrica de la memoria que se puede liberar (`FreeableMemory`) para determinar si su clúster de base de datos se está quedando sin memoria. Luego puede comprobar la métrica de operaciones de escritura (`VolumeWriteIOPS`) para ver si la instancia de escritura recibe una carga pesada de operaciones de escritura. En ese caso, es recomendable que actualice su aplicación para limitar el número de inserciones de una transacción a menos de 1 millón o que modifique la instancia para que utilice una de las clases de instancia de base de datos R compatibles (escalado del cálculo).

Optimización de las consultas de combinación indexadas de Aurora MySQL con la captura previa de claves asíncronas

Amazon MySQL puede usar la característica de captura previa de claves asíncronas (AKP, por sus siglas en inglés) para mejorar el desempeño de las consultas que unen las tablas a través de los índices. Esta característica mejora el rendimiento al prever las filas necesarias para ejecutar consultas en las que una consulta JOIN requiere el uso del algoritmo de combinación Batched Key Access (BKA) y las características de optimización Multi-Range Read (MRR). Para obtener más información acerca de BKA y MRR, consulte Block Nested-Loop and Batched Key Access Joins y Multi-Range Read Optimization en la documentación de MySQL.

Para sacar máximo partido de la característica AKP, una consulta debe utilizar BKA y MRR. Por lo general, dicho tipo de consulta se produce cuando la cláusula JOIN de una consulta utiliza un índice secundario, pero requiere también algunas columnas del índice primario. Por ejemplo, puede usar la AKP cuando una cláusula JOIN represente a un equijoin en los valores de índice entre una tabla exterior pequeña y una interior grande, y el índice sea sumamente selectivo en la tabla grande. AKP trabaja conjuntamente con BKA y MRR para llevar a cabo una búsqueda del índice secundario al primario durante la evaluación de la cláusula JOIN. AKP identifica las filas necesarias para ejecutar la consulta durante la evaluación de la cláusula JOIN. Seguidamente, utiliza un subproceso en segundo plano para cargar de manera asíncrona las páginas que contienen dichas filas en la memoria antes de ejecutar la consulta.

AKP está disponible para la versión 2.10 y posteriores y la versión 3 de Aurora MySQL. Para obtener más información acerca de las versiones de Aurora MySQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.

Habilitación de la captura previa de clave asíncrona

Para habilitar la característica de AKP, establezca la configuración `aurora_use_key_prefetch`, una variable de MySQL Server, en `on`. De forma predeterminada, este valor se establece en `on`. No obstante, AKP no podrá habilitarse hasta que no active el algoritmo de combinación BKA y deshabilite la característica MRR basada en el costo. Para ello, debe ajustar los valores siguientes de `optimizer_switch`, una variable de MySQL Server:

- Establece `batched_key_access` en `on`. Este valor controla el uso del algoritmo de combinación BKA. De forma predeterminada, este valor se establece en `off`.
- Establece `mrr_cost_based` en `off`. Este valor controla el uso de la característica MRR basada en el costo. De forma predeterminada, este valor se establece en `on`.

En la actualidad, puede configurar estos valores únicamente en el nivel de sesión. El siguiente ejemplo ilustra cómo configurar estos valores a fin de habilitar AKP para la sesión actual ejecutando las instrucciones SET.

```
mysql> set @@session.aurora_use_key_prefetch=on;
mysql> set @@session.optimizer_switch='batched_key_access=on,mrr_cost_based=off';
```

Del mismo modo, puede usar las instrucciones SET para deshabilitar AKP y el algoritmo de combinación BKA, y volver a habilitar la característica MRR basada en el costo para la sesión actual, como se muestra en el ejemplo siguiente.

```
mysql> set @@session.aurora_use_key_prefetch=off;
mysql> set @@session.optimizer_switch='batched_key_access=off,mrr_cost_based=on';
```

Para obtener más información acerca de los cambios del optimizador batched_key_access y mrr_cost_based, consulte Switchable Optimizations en la documentación de MySQL.

Optimización de consultas para la captura previa de clave asíncrona

Puede confirmar si una consulta puede beneficiarse de la característica AKP. Para ello, utilice la instrucción `EXPLAIN` con el fin de perfilar la consulta antes de ejecutarla. La instrucción `EXPLAIN` ofrece información acerca del plan de ejecución que se va a utilizar para una consulta especificada

En el resultado de la instrucción `EXPLAIN`, la columna `Extra` describe información adicional que se incluye con el plan de ejecución. Si la característica AKP se aplica a una tabla que se ha utilizado en la consulta, esta tabla incluye uno de los siguientes valores:

- `Using Key Prefetching`
- `Using join buffer (Batched Key Access with Key Prefetching)`

En el siguiente ejemplo se muestra el uso de `EXPLAIN` para ver el plan de ejecución de una consulta que puede beneficiarse de AKP.

```
mysql> explain select sql_no_cache
    ->     ps_partkey,
    ->     sum(ps_supplycost * ps_availqty) as value
    -> from
    ->     partsupp,
    ->     supplier,
    ->     nation
    -> where
    ->     ps_suppkey = s_suppkey
    ->     and s_nationkey = n_nationkey
    ->     and n_name = 'ETHIOPIA'
    -> group by
    ->     ps_partkey having
    ->         sum(ps_supplycost * ps_availqty) > (
    ->             select
    ->                 sum(ps_supplycost * ps_availqty) * 0.0000003333
    ->             from
    ->                 partsupp,
    ->                 supplier,
    ->                 nation
    ->             where
    ->                 ps_suppkey = s_suppkey
    ->                 and s_nationkey = n_nationkey
    ->                 and n_name = 'ETHIOPIA'
    ->         )
    -> order by
    ->     value desc;
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
```

```
| id | select_type | table    | type | possible_keys         | key           | key_len
 | ref                              | rows | filtered | Extra
                              |
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
|  1 | PRIMARY     | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where; Using temporary;
 Using filesort                |
|  1 | PRIMARY     | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  1 | PRIMARY     | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
|  2 | SUBQUERY    | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where
                              |
|  2 | SUBQUERY    | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  2 | SUBQUERY    | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
6 rows in set, 1 warning (0.00 sec)
```

Para obtener más información acerca del formato de salida `EXPLAIN`, consulte el tema sobre el Formato de salida de EXPLAIN ampliado en la documentación de MySQL.

Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash

Si necesita unir una gran cantidad de datos mediante equijoin, una combinación hash puede mejorar el desempeño de las consultas. Puede habilitar las combinaciones hash para Aurora MySQL.

Una columna de combinación hash puede ser cualquier expresión compleja. En una columna de combinación hash puede realizar comparaciones entre distintos tipos de datos de las formas siguientes:

- Puede comparar cualquier cosa en la categoría de tipos de datos numéricos precisos, como `int`, `bigint`, `numeric` y `bit`.
- Puede comparar cualquier cosa en la categoría de tipos de datos numéricos aproximados, como `float` y `double`.
- Puede comparar elementos en distintos tipos de cadena que tengan el mismo conjunto de caracteres y la misma intercalación.
- Puede comparar elementos con tipos de datos de marca de fecha y hora si los tipos son los mismos.

> **Note**
>
> No se pueden comparar tipos de datos de categorías distintas.

Las siguientes restricciones se aplican a las combinaciones hash de Aurora MySQL:

- Las uniones exteriores izquierda y derecha no son compatibles con la versión 2 de Aurora MySQL, pero sí con la versión 3.
- No se admiten las semicombinaciones, como las subconsultas, a no ser que las subconsultas se materialicen primero.
- No se admiten las actualizaciones o supresiones de varias tablas.

  > **Note**
  >
  > Se admiten las actualizaciones o supresiones de una sola tabla.

- Las columnas de tipos de datos especiales y BLOB no pueden ser columnas de unión en una combinación hash.

Habilitación de las combinaciones hash

Para habilitar combinaciones hash:

- Aurora MySQL versión 2: establezca el parámetro de base de datos o el parámetro del clúster de base de datos `aurora_disable_hash_join` en `0`. Si `aurora_disable_hash_join` está desactivado, se establece el valor de `optimizer_switch` en `hash_join=on`.

- Aurora MySQL versión 3: establezca el parámetro del servidor MySQL `optimizer_switch` en `block_nested_loop=on`.

Las combinaciones hash están activadas de forma predeterminada en la versión 3 de Aurora MySQL y están desactivadas de forma predeterminada en la versión 2 de Aurora MySQL. En el siguiente ejemplo, se ilustra cómo habilitar las combinaciones hash para la versión 3 de Aurora MySQL. Puede emitir la declaración `select @@optimizer_switch` primero para ver qué otras configuraciones están presentes en la cadena del parámetro `SET`. La actualización de un ajuste en el parámetro `optimizer_switch` no borra ni modifica las demás configuraciones.

```
mysql> SET optimizer_switch='block_nested_loop=on';
```

> **Note**
>
> Para Aurora MySQL, versión 3, se admite la combinación hash en todas las versiones secundarias y se activa de forma predeterminada.
> Para la versión 2 de Aurora MySQL, se admiten combinaciones hash en todas las versiones secundarias. En Aurora MySQL versión 2, la característica de combinación hash siempre está controlada por el valor `aurora_disable_hash_join`.

Con esta configuración, el optimizador elige usar una combinación hash basada en el costo, las características de la consulta y la disponibilidad de los recursos. Si la estimación del costo es incorrecta, puede forzar al optimizador a elegir una combinación hash. Para ello, establezca `hash_join_cost_based`, una variable de MySQL Server, en `off`. En el siguiente ejemplo se ilustra cómo forzar al optimizador a elegir una combinación hash.

```
mysql> SET optimizer_switch='hash_join_cost_based=off';
```

> **Note**
>
> Esta configuración anula las decisiones del optimizador basado en costos. Aunque esta configuración puede ser útil para las pruebas y el desarrollo, le recomendamos que no la utilice en producción.

Optimización de consultas para combinaciones hash

Para averiguar si una consulta puede beneficiarse de usar una combinación hash, utilice antes la instrucción `EXPLAIN` para perfilar la consulta. La instrucción `EXPLAIN` ofrece información acerca del plan de ejecución que se va a utilizar para una consulta especificada

En el resultado de la instrucción `EXPLAIN`, la columna `Extra` describe información adicional que se incluye con el plan de ejecución. Si se aplica una combinación hash a las tablas utilizadas en la consulta, esta columna incluye valores similares a los siguientes:

- `Using where; Using join buffer (Hash Join Outer table` *`table1_name`*`)`
- `Using where; Using join buffer (Hash Join Inner table` *`table2_name`*`)`

En el siguiente ejemplo se muestra el uso de EXPLAIN para ver el plan de ejecución de una consulta hash join (con combinación hash).

```
mysql> explain SELECT sql_no_cache * FROM hj_small, hj_big, hj_big2
    ->     WHERE hj_small.col1 = hj_big.col1 and hj_big.col1=hj_big2.col1 ORDER BY 1;
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows |
 Extra                                                          |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
|  1 | SIMPLE      | hj_small | ALL  | NULL          | NULL | NULL    | NULL |    6 |
 Using temporary; Using filesort                                |
|  1 | SIMPLE      | hj_big   | ALL  | NULL          | NULL | NULL    | NULL |   10 |
 Using where; Using join buffer (Hash Join Outer table hj_big)  |
|  1 | SIMPLE      | hj_big2  | ALL  | NULL          | NULL | NULL    | NULL |   15 |
 Using where; Using join buffer (Hash Join Inner table hj_big2) |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
3 rows in set (0.04 sec)
```

En la salida, la `Hash Join Inner table` es la tabla utilizada para crear la tabla hash y la `Hash Join Outer table` es la tabla usada para sondear la tabla hash.

Para obtener más información acerca del formato de salida ampliado `EXPLAIN`, consulte Extended EXPLAIN Output Format (Formato de salida ampliado de EXPLAIN) en la documentación de producto de MySQL.

En Aurora MySQL 2.08 y versiones posteriores, puede usar sugerencias SQL para influir en si una consulta usa combinación hash o no, y las tablas que usar para los lados de compilación y sondeo de la combinación. Para obtener más información, consulte Sugerencias de Aurora MySQL.

Uso de Amazon Aurora para escalar las lecturas de una base de datos de MySQL

Puede usar Amazon Aurora con su instancia de base de datos MySQL para aprovechar las capacidades de escalado de lectura de Amazon Aurora y ampliar la carga de trabajo de lectura para su instancia de base de datos MySQL. Si desea usar Aurora para leer la escala de su instancia de base de datos de MySQL, cree un clúster de base de datos de Aurora MySQL y haga que sea una réplica de lectura de su instancia de base de datos de MySQL. A continuación, conéctese al clúster Aurora MySQL para procesar las consultas de lectura. La base de datos de origen puede ser una instancia de base de datos de RDS for MySQL o una base de datos de MySQL que se ejecute fuera de Amazon RDS. Para obtener más información, consulte Escalado de lecturas para su base de datos MySQL con Amazon Aurora.

Optimización de las operaciones de marca temporal

Cuando el valor de la variable de sistema `time_zone` se establece en `SYSTEM`, cada llamada a una función de MySQL que requiera un cálculo de zona horaria realiza una llamada a la biblioteca del sistema. Al ejecutar instrucciones SQL que devuelvan o modifiquen dichos valores de `TIMESTAMP` con una alta concurrencia, es posible que aumente la latencia, la contención de bloqueos y el uso de la CPU. Para obtener más información, consulte time_zone en la documentación de MySQL.

Para evitar este comportamiento, se recomienda cambiar el valor del parámetro `time_zone` del clúster de base de datos a `UTC`. Para obtener más información, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Si bien el parámetro `time_zone` es dinámico (no requiere el reinicio del servidor de base de datos), el nuevo valor solo se usará para las conexiones nuevas. Para asegurarse de que todas las conexiones se actualicen y utilicen el nuevo valor de `time_zone`, se recomienda que recicle las conexiones de la aplicación después de actualizar el parámetro del clúster de base de datos.

Errores de desbordamiento de ID de índice virtual

Aurora MySQL limita los valores de los ID de índice virtuales a 8 bits para evitar un problema provocado por el formato de deshacer de MySQL. Si un índice supera el límite de ID de índice virtual, es posible que el clúster no esté disponible. Cuando un índice se acerca al límite de ID de índice virtual o cuando intenta crear un índice por encima del límite de ID de índice virtual, es posible que

RDS arroje un código de error 63955 o un código de advertencia 63955. Para solucionar un error de límite de ID de índice virtual, le recomendamos que vuelva a crear la base de datos con un volcado lógico y una restauración.

Para obtener más información sobre el volcado lógico y la restauración de Amazon Aurora MySQL, consulte Migrar bases de datos muy grandes a Amazon Aurora MySQL mediante MyDumper y MyLoader. Para obtener más información sobre cómo acceder a los registros de errores en Amazon Aurora, consulte Supervisión de archivos de registro de Amazon Aurora.

Prácticas recomendadas para obtener una elevada disponibilidad de Aurora MySQL

Aplique las prácticas recomendadas que se describen a continuación para mejorar la disponibilidad de los clústeres de Aurora MySQL.

Temas

- Uso de Amazon Aurora para la recuperación de desastres con sus bases de datos de MySQL
- Migración de MySQL a Amazon Aurora MySQL con un tiempo de inactividad reducido
- Prevención del rendimiento lento, el reinicio automático y la conmutación por error de las instancias de base de datos Aurora MySQL

Uso de Amazon Aurora para la recuperación de desastres con sus bases de datos de MySQL

Puede usar Amazon Aurora con su instancia de base de datos MySQL para crear una copia de sseguridad fuera del sitio para la recuperación de desastres. Para usar Aurora para la recuperación de desastres de su instancia de base de datos de MySQL, cree un clúster de base de datos de Amazon Aurora y haga que sea una réplica de lectura de su instancia de base de datos de MySQL. Esto es válido para una instancia de base de datos de RDS for MySQL o una base de datos de MySQL que se ejecute fuera de Amazon RDS.

> **Important**
>
> Cuando configure la replicación entre una instancia de base de datos MySQL y un clúster de base de datos Amazon Aurora MySQL, debe monitorizar la replicación para asegurarse de que sigue estando en un estado correcto y repararla si es necesario.

Para obtener instrucciones acerca de cómo crear un clúster de base de datos de Amazon Aurora MySQL y convertirlo en una réplica de lectura de su instancia de base de datos de MySQL, siga el procedimiento que se describe en Uso de Amazon Aurora para escalar las lecturas de una base de datos de MySQL.

Para obtener más información acerca de los modelos de recuperación de desastres, consulte How to choose the best disaster recovery option for your Amazon Aurora MySQL cluster.

Migración de MySQL a Amazon Aurora MySQL con un tiempo de inactividad reducido

Al importar datos desde una base de datos de MySQL que admita una aplicación en directo a un clúster de base de datos de Amazon Aurora MySQL, puede que desee reducir la cantidad de tiempo que se interrumpe el servicio mientras se produce la migración. Para ello, puede usar el procedimiento documentado en Importación de datos a una instancia de base de datos de Amazon RDS para MySQL con tiempo de inactividad reducido que figura en la Guía del usuario de Amazon Relational Database Service. Este procedimiento puede resultar especialmente útil al trabajar con una base de datos de gran tamaño. Puede usar dicho procedimiento para reducir el costo de la importación, ya que minimiza la cantidad de datos que se transfieren a por la red AWS.

El procedimiento muestra los pasos necesarios para transferir una copia de los datos de la base de datos a una instancia de Amazon EC2 e importar los datos en una nueva instancia de base de datos de RDS for MySQL. Dado que Amazon Aurora es compatible con MySQL, puede usar un clúster de base de datos Amazon Aurora para la instancia de base de datos MySQL en Amazon RDS de destino.

Prevención del rendimiento lento, el reinicio automático y la conmutación por error de las instancias de base de datos Aurora MySQL

Si ejecuta una carga de trabajo pesada o cargas de trabajo que superan los recursos asignados a su instancia de base de datos, puede agotar los recursos en los que ejecuta la aplicación y la base de datos de Aurora. Para obtener métricas de su instancia de base de datos, como el uso de la CPU, el uso de la memoria y el número de conexiones de base de datos utilizadas, puede consultar las métricas proporcionadas por Amazon CloudWatch, Performance Insights y Enhanced Monitoring. Para obtener más información acerca de la monitorización de las métricas de las instancias de base de datos, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Si su carga de trabajo agota los recursos que utiliza, su instancia de base de datos podría ralentizarse, reiniciarse o incluso realizar una conmutación por error a otra instancia de base

de datos. Para evitarlo, supervise la utilización de los recursos, examine la carga de trabajo que se ejecuta en la instancia de base de datos y realice las optimizaciones necesarias. Si las optimizaciones no mejoran las métricas de la instancia ni mitigan el agotamiento de los recursos, considere la posibilidad de ampliar la instancia de base de datos antes de alcanzar sus límites. Para obtener más información sobre las clases de instancias de base de datos disponibles y sus especificaciones, consulte Clases de instancia de base de datos de Amazon Aurora.

Recomendaciones para características de MySQL en Aurora MySQL

A continuación se describen las funciones que están disponibles en Aurora MySQL para ofrecer compatibilidad con MySQL. Sin embargo, tienen problemas de rendimiento, escalabilidad, estabilidad o compatibilidad en el entorno de Aurora. Por lo tanto, le recomendamos que siga determinadas directrices en el uso de estas características. Por ejemplo, le recomendamos que no utilice determinadas funciones para las implementaciones de Aurora de producción.

Temas

- Uso de la replicación de varios subprocesos en Aurora MySQL
- Invocación de funciones de AWS Lambda mediante funciones de MySQL nativas
- Evitar las transacciones de XA con Amazon Aurora MySQL
- Mantenimiento de las claves externas activadas durante las instrucciones DML
- Configuración de la frecuencia de vaciado del búfer de registro
- Minimización y solución de problemas de los interbloqueos de Aurora MySQL

Uso de la replicación de varios subprocesos en Aurora MySQL

Con la replicación de registros binarios de múltiples procesos, un subproceso SQL lee los eventos del registro de retransmisión y los pone en cola para que se apliquen los subprocesos de trabajo de SQL. Los subprocesos de trabajo SQL se administran mediante un subproceso coordinador. Los eventos de registros binarios se aplican en paralelo cuando es posible.

La replicación de varios subprocesos se admite en la versión 3 de Aurora MySQL y la versión 2.12.1 de Aurora MySQL y versiones posteriores.

Para versiones de Aurora MySQL anteriores a 3.04, Aurora utiliza de forma predeterminada la replicación de subproceso único cuando se utiliza un clúster de base de datos de Aurora MySQL como réplica de lectura para la replicación de registros binarios.

Las versiones anteriores a la versión 2 de Aurora MySQL heredó de MySQL Community Edition varios problemas relativos a la replicación con varios subprocesos. Para esas versiones, no es recomendable usar la replicación con varios subprocesos en entornos de producción.

Si utiliza la replicación de varios subprocesos, le recomendamos que realice pruebas exhaustivas.

Para obtener más información acerca del uso de la replicación en Amazon Aurora, consulte Replicación con Amazon Aurora. Para obtener más información sobre la replicación de varios subprocesos en Aurora MySQL, consulte Replicación de registros binarios de varios subprocesos.

Invocación de funciones de AWS Lambda mediante funciones de MySQL nativas

Recomendamos utilizar las funciones nativas de MySQL `lambda_sync` y `lambda_async` para invocar las funciones de Lambda.

Si utiliza el procedimiento obsoleto `mysql.lambda_async`, es recomendable integrar las llamadas al procedimiento `mysql.lambda_async` en un procedimiento almacenado. Puede llamar a este procedimiento almacenado desde distintos orígenes, como los disparadores o el código cliente. Este método puede ayudar a evitar los problemas de discrepancia de la impedancia y facilitar a los programadores de bases de datos la invocación de las funciones de Lambda.

Para obtener más información acerca de la invocación de funciones de Lambda desde Amazon Aurora, consulte Invocación de una función de Lambda desde un clúster de base de datos de Amazon Aurora MySQL.

Evitar las transacciones de XA con Amazon Aurora MySQL

No es recomendable usar transacciones de eXtended Architecture (XA) con Aurora MySQL, ya que pueden provocar tiempos de recuperación prolongados si la transacción de XA se encuentra en el estado `PREPARED`. Si debe usar transacciones de XA con Aurora MySQL, siga estas prácticas recomendadas:

- No deje ninguna transacción de XA abierta en el estado `PREPARED`.
- Mantenga las transacciones de XA tan pequeñas como sea posible.

Para obtener más información acerca del uso de transacciones de XA con MySQL, consulte XA Transactions en la documentación de MySQL.

Mantenimiento de las claves externas activadas durante las instrucciones DML

Le recomendamos encarecidamente que no ejecute instrucciones en lenguaje de definición de datos (DDL) cuando la variable `foreign_key_checks` está definida en `0` (apagada).

Si tiene que insertar o actualizar filas que requieran una infracción transitoria de claves externas, siga estos pasos:

1. Establezca `foreign_key_checks` en `0`.
2. Realice sus cambios de lenguaje de manipulación de datos (DML).
3. Asegúrese de que los cambios completados no infrinjan ninguna restricción de claves externas.
4. Establezca `foreign_key_checks` en `1` (activado).

Además, siga estas otras prácticas recomendadas para restricciones de clave externa:

- Asegúrese de que sus aplicaciones cliente no establezcan la variable `foreign_key_checks` en `0` como parte de la variable `init_connect`.
- Si una restauración a partir de una copia de seguridad lógica `mysqldump` deja de funcionar o está incompleta, asegúrese de que `foreign_key_checks` esté definido en `1` antes de iniciar alguna otra operación en la misma sesión. Una copia de seguridad lógica establece `foreign_key_checks` en `0` cuando se inicia.

Configuración de la frecuencia de vaciado del búfer de registro

En MySQL Community Edition, para que las transacciones sean duraderas, el búfer de registro de InnoDB debe vaciarse en un almacenamiento duradero. Utilice el parámetro `innodb_flush_log_at_trx_commit` para configurar la frecuencia con la que se vacía el búfer de registro en un disco.

Al establecer el parámetro `innodb_flush_log_at_trx_commit` en el valor predeterminado de 1, el búfer de registro se vacía en cada confirmación de transacción. Esta configuración ayuda a mantener la base de datos compatible con ACID. Le recomendamos que utilice el ajuste predeterminado de 1.

Cambiar `innodb_flush_log_at_trx_commit` a un valor no predeterminado puede ayudar a reducir la latencia del lenguaje de manipulación de datos (DML), pero se sacrifica la durabilidad de los registros. Esta falta de durabilidad hace que la base de datos ACID no sea compatible.

Recomendamos que sus bases de datos cumplan con ACID para evitar el riesgo de pérdida de datos si se reinicia el servidor. Para obtener más información sobre este parámetro, consulte innodb_flush_log_at_trx_commit.

En Aurora MySQL, el procesamiento de un registro de rehacer se descarga a la capa de almacenamiento, por lo que no se descargan los archivos de registro en la instancia de base de datos. Cuando se emite una escritura, los registros de rehacer se envían desde la instancia de base de datos del escritor directamente al volumen del clúster de Aurora. Las únicas escrituras que cruzan la red son los registros de rehacer. Nunca se escribe ninguna página desde el nivel de base de datos.

De manera predeterminada, cada subproceso que confirma una transacción espera la confirmación del volumen del clúster de Aurora. Esta confirmación indica que este registro y todos los registros anteriores de rehacer están escritos y han alcanzado el quórum. Si se conservan los registros y se alcanza el quórum, la transacción es duradera, ya sea mediante confirmación automática o explícita. Para obtener más información sobre la arquitectura de almacenamiento de Aurora, consulte Amazon Aurora storage demystified.

Aurora MySQL no vacía los registros en los archivos de datos como lo hace MySQL Community Edition. Sin embargo, puede usar el parámetro `innodb_flush_log_at_trx_commit` para relajar las restricciones de durabilidad al escribir registros de rehacer en el volumen del clúster de Aurora.

Para Aurora MySQL, versión 2.

- `innodb_flush_log_at_trx_commit` = 0 o 2: la base de datos no espera a que se confirme que los registros redo se han escrito en el volumen del clúster de Aurora.
- `innodb_flush_log_at_trx_commit` = 1: la base de datos espera a que se confirme que los registros redo se han escrito en el volumen del clúster de Aurora.

Para Aurora MySQL, versión 3:

- `innodb_flush_log_at_trx_commit` = 0: la base de datos no espera a que se confirme que los registros redo se han escrito en el volumen del clúster de Aurora.
- `innodb_flush_log_at_trx_commit` = 1 o 2: la base de datos espera a que se confirme que los registros redo se han escrito en el volumen del clúster de Aurora.

Por lo tanto, para obtener el mismo comportamiento no predeterminado en Aurora MySQL versión 3 que con el valor establecido en 0 o 2 de Aurora MySQL versión 2, establezca el parámetro en 0.

Si bien estas configuraciones pueden reducir la latencia de DML para el cliente, también pueden provocar la pérdida de datos en caso de una conmutación por error o un reinicio. Por lo tanto, le recomendamos que mantenga el parámetro `innodb_flush_log_at_trx_commit` establecido en el valor predeterminado de 1.

Si bien la pérdida de datos puede ocurrir tanto en MySQL Community Edition como en Aurora MySQL, el comportamiento difiere en cada base de datos debido a sus diferentes arquitecturas. Estas diferencias arquitectónicas pueden provocar diversos grados de pérdida de datos. Para asegurarse de que su base de datos es compatible con ACID, establezca siempre `innodb_flush_log_at_trx_commit` en 1.

> **Note**
>
> En la versión 3 de Aurora MySQL, antes de poder cambiar `innodb_flush_log_at_trx_commit` a un valor distinto de 1, primero debe cambiar el valor de `innodb_trx_commit_allow_data_loss` a 1. Al hacerlo, acepta el riesgo de una posible pérdida de datos.

Minimización y solución de problemas de los interbloqueos de Aurora MySQL

Los usuarios que ejecutan cargas de trabajo que sufren regularmente infracciones de las restricciones en índices secundarios únicos o claves externas, al modificar registros de la misma página de datos de forma simultánea, podrían experimentar un aumento de los interbloqueos y los tiempos de espera de bloqueo. Estos interbloqueos y tiempos de espera se deben a una corrección de errores de MySQL Community Edition.

Esta corrección se incluye en las versiones 5.7.26 y posteriores de MySQL Community Edition, y se ha introducido también en las versiones 2.10.3 y posteriores de Aurora MySQL. La corrección es necesaria para aplicar la serialización mediante la implementación de un bloqueo adicional para este tipo de operaciones del lenguaje de manipulación de datos (DML) en los cambios realizados en los registros de una tabla de InnoDB. Este problema se descubrió como parte de una investigación sobre los problemas de interbloqueos introducidos en una corrección de errores anterior de MySQL Community Edition.

La corrección cambió el manejo interno de la reversión parcial de una actualización de tuplas (filas) en el motor de almacenamiento de InnoDB. Las operaciones que generan infracciones de restricciones en claves externas o índices secundarios únicos provocan una reversión parcial. Esto

incluye, entre otras, instrucciones `INSERT...ON DUPLICATE KEY UPDATE`, `REPLACE INTO`, y `INSERT IGNORE` simultáneas (upserts).

En este contexto, la reversión parcial no se refiere a la reversión de las transacciones a nivel de la aplicación, sino a la reversión interna de InnoDB de los cambios en un índice en clúster cuando se detecta una infracción de una restricción. Supongamos que se encuentra un valor de clave duplicado durante una operación upsert.

En una operación de inserción normal, InnoDB crea de forma atómica entradas de índice secundario y en clúster para cada índice. Si InnoDB detecta un valor duplicado en un índice secundario único durante una operación upsert, la entrada insertada en el índice en clúster debe revertirse (reversión parcial) y, a continuación, debe aplicarse la actualización a la fila duplicada existente. Durante este paso de reversión parcial interna, InnoDB debe bloquear cada registro que se detecte como parte de la operación. La solución garantiza la serialización de las transacciones al introducir un bloqueo adicional después de la reversión parcial.

Minimización de interbloqueos de InnoDB

Puede adoptar los siguientes métodos para reducir la frecuencia de los interbloqueos en la instancia de base de datos. Encontrará más ejemplos en la documentación de MySQL.

1. Para reducir las probabilidades de que se produzcan interbloqueos, confirme las transacciones inmediatamente después de realizar un conjunto de cambios relacionados. Para ello, puede dividir las transacciones de gran tamaño (actualizaciones de varias filas entre confirmaciones) en transacciones más pequeñas. Si va a insertar filas por lotes, intente reducir el tamaño de las inserciones por lotes, especialmente cuando utilice las operaciones upsert mencionadas anteriormente.

   Para reducir el número de posibles reversiones parciales, puede probar algunos de los siguientes métodos:

   a. Sustituya las operaciones de inserción por lotes por inserciones de filas de una en una. Esto puede reducir el tiempo durante el que las transacciones mantienen bloqueos que pueden generar conflictos.

   b. En lugar de usar `REPLACE INTO`, reescriba la instrucción SQL como una transacción de múltiples instrucciones, como la siguiente:

```
BEGIN;
DELETE conflicting rows;
INSERT new rows;
```

```
COMMIT;
```

c. En lugar de usar `INSERT...ON DUPLICATE KEY UPDATE`, reescriba la instrucción SQL como una transacción de múltiples instrucciones, como la siguiente:

```
BEGIN;
SELECT rows that conflict on secondary indexes;
UPDATE conflicting rows;
INSERT new rows;
COMMIT;
```

2. Evite las transacciones de larga duración, activas o inactivas, que podrían mantener bloqueos. Esto incluye sesiones interactivas de clientes MySQL que podrían estar abiertas durante un periodo de tiempo prolongado con una transacción no confirmada. Al optimizar los tamaños de las transacciones o los tamaños de lotes, el resultado puede variar en función de varios factores, como la simultaneidad, el número de duplicados y la estructura de la tabla. Cualquier cambio debe implementarse y probarse en función de su carga de trabajo.
3. En algunas situaciones, pueden producirse interbloqueos cuando dos transacciones intentan acceder a los mismos conjuntos de datos, ya sea en una o varias tablas, en órdenes diferentes. Para evitarlo, puede modificar las transacciones para que accedan a los datos en el mismo orden, serializando así el acceso. Por ejemplo, cree una cola de transacciones que deben completarse. Este método puede ayudar a evitar interbloqueos cuando se realizan varias transacciones al mismo tiempo.
4. Añadir índices cuidadosamente seleccionados a las tablas puede mejorar la selectividad y reducir la necesidad de acceder a las filas, lo que disminuye los bloqueos.
5. Si se encuentra con un bloqueo de espacio, puede modificar el nivel de aislamiento de las transacciones por `READ COMMITTED` para la sesión o la transacción para evitarlo. Para obtener más información sobre los niveles de aislamiento de InnoDB y sus comportamientos, consulte Transaction isolation levels (Niveles de aislamiento de las transacciones) en la documentación de MySQL.

> **Note**
>
> Aunque puede tomar medidas para reducir la posibilidad de que se produzcan interbloqueos, son un comportamiento normal de la base de datos y pueden producirse a pesar de todo. Las aplicaciones deben tener la lógica necesaria para gestionar los interbloqueos cuando se encuentren. Por ejemplo, implemente lógica de reintento e interrupción en la aplicación.

Es mejor solucionar la causa principal del problema, pero si se produce un interbloqueo, la aplicación tiene la opción de esperar y volver a intentarlo.

Supervisión de interbloqueos de InnoDB

Pueden producirse interbloqueos en MySQL cuando las transacciones de la aplicación intentan aceptar bloqueos a nivel de tabla y fila de una forma que da lugar a una espera circular. Un interbloqueo ocasional de InnoDB no tiene por qué ser necesariamente un problema, ya que el motor de almacenamiento de InnoDB detecta la condición inmediatamente y anula una de las transacciones automáticamente. Si se encuentra con interbloqueos con frecuencia, le recomendamos que revise y modifique la aplicación para reducir los problemas de rendimiento y evitar los interbloqueos. Cuando la detección de interbloqueos está activada (opción predeterminada), InnoDB detecta automáticamente los interbloqueos de transacciones y revierte una transacción o transacciones para salir del interbloqueo. InnoDB intenta seleccionar transacciones pequeñas para revertirlas, donde el tamaño de una transacción viene determinado por el número de filas insertadas, actualizadas o eliminadas.

- Instrucción `SHOW ENGINE`: la instrucción `SHOW ENGINE INNODB STATUS \G` contiene detalles del último interloqueo encontrado en la base de datos desde el último reinicio.
- Registro de errores de MySQL: si se encuentra interbloqueos frecuentes en los que la salida de la instrucción `SHOW ENGINE` no es adecuada, puede activar el parámetro del clúster de base de datos innodb_print_all_deadlocks.

  Cuando este parámetro está activado, la información sobre todos los interbloqueos en las transacciones de los usuarios de InnoDB se registra en el registro de errores de Aurora MySQL.
- Métricas de Amazon CloudWatch: también le recomendamos que supervise de forma proactiva los interbloqueos mediante la métrica de CloudWatch `Deadlocks`. Para obtener más información, consulte Métricas de nivel de instancia para Amazon Aurora.
- Registros de Amazon CloudWatch: con Registros de CloudWatch, puede ver métricas, analizar datos de registro y crear alarmas en tiempo real. Para obtener más información, consulte Monitor errors in Amazon Aurora MySQL and Amazon RDS for MySQL using Amazon CloudWatch and send notifications using Amazon SNS (Supervisión de errores en Amazon Aurora MySQL y Amazon RDS para MySQL mediante Amazon CloudWatch y envío de notificaciones mediante Amazon SNS).

Si utiliza Registros de CloudWatch con la opción `innodb_print_all_deadlocks` activada, puede configurar alarmas para que le notifiquen cuando el número de interbloqueos supere un umbral determinado. Para definir un umbral, le recomendamos que observe las tendencias y utilice un valor basado en su carga de trabajo normal.

- Información sobre rendimiento: al utilizar Información sobre rendimiento, puede supervisar las métricas `innodb_deadlocks` y `innodb_lock_wait_timeout`. Para obtener más información sobre estas métricas, consulte Contadores no nativos para Aurora MySQL.

Evaluación de la utilización de instancias de base de datos para Aurora MySQL con métricas de Amazon CloudWatch

Puede usar las métricas de CloudWatch para supervisar el rendimiento de su instancia de base de datos y determinar si su clase de instancia de base de datos proporciona recursos suficientes para sus aplicaciones. Para obtener más información acerca de los límites de las clases de instancia de base de datos, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora. Busque las especificaciones de la clase de instancia de base de datos para encontrar el rendimiento de la red.

Si el uso de la instancia de base de datos está cerca del límite de las clases de instancia, es posible que el rendimiento comience a disminuir. Las métricas de CloudWatch pueden confirmar esta situación para que pueda planificar el escalado vertical manual a una clase de instancia más grande.

Combine los siguientes valores de métricas de CloudWatch para averiguar si se acerca al límite de clases de instancia:

- NetworkThroughput: rendimiento de red que reciben y transmiten los clientes para cada instancia en el clúster de base de datos de Aurora. Este valor de rendimiento no incluye el tráfico de red entre las instancias del clúster de bases de datos y el volumen de clúster.
- StorageNetworkThroughput: rendimiento de red que recibe el subsistema de almacenamiento de Aurora y que cada instancia del clúster de bases de datos de Aurora envía al subsistema de almacenamiento de Aurora.

Sume la métrica NetworkThroughput a StorageNetworkThroughput para determinar el rendimiento de red recibido y enviado al subsistema de almacenamiento de Aurora por cada instancia del clúster de base de datos de Aurora. El límite de clases de instancia para su instancia debe ser mayor que la suma de estas dos métricas combinadas.

Puede utilizar las siguientes métricas para revisar detalles adicionales del tráfico de red de las aplicaciones cliente al enviar y recibir:

- NetworkReceiveThroughput: rendimiento de la red recibido de los clientes por cada instancia de base de datos del clúster de base de datos de Aurora MySQL. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de y el volumen de clúster.
- NetworkTransmitThroughput: rendimiento de red enviado a los clientes por cada instancia del clúster de bases de datos de Aurora. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de y el volumen de clúster.
- StorageNetworkReceiveThroughput: rendimiento de red recibido del subsistema de almacenamiento de Aurora por cada instancia del clúster de bases de datos.
- StorageNetworkTransmitThroughput: rendimiento de red enviado al subsistema de almacenamiento de Aurora por cada instancia del clúster de bases de datos.

Sume todas estas métricas para evaluar cuál es el uso de su red con respecto al límite de las clases de instancia de base de datos. El límite de las clases de instancias debe ser mayor que la suma de estas métricas combinadas.

Los límites de la red y el uso de la CPU para el almacenamiento están directamente relacionados. Cuando aumenta el rendimiento de la red, también aumenta el uso de la CPU. La supervisión del uso de la CPU y la red proporciona información sobre cómo y por qué se agotan los recursos.

Para ayudar a minimizar el uso de la red, puede plantearse lo siguiente:

- Utilizar una clase de instancia de base de datos mayor.
- Dividir las solicitudes de escritura en lotes para reducir las transacciones generales.
- Redirigir la carga de trabajo de solo lectura a una instancia de solo lectura.
- Eliminar los índices no utilizados.

Solución de problemas de rendimiento de las bases de datos Amazon Aurora MySQL

Este tema se centra en algunos problemas comunes de rendimiento de la base de datos Aurora MySQL y en cómo solucionar o recopilar información para solucionarlos rápidamente. Dividimos el rendimiento de las bases de datos en dos categorías:

- Rendimiento del servidor: todo el servidor de base de datos funciona más lento.
- Rendimiento de las consultas: una o más consultas tardan más en ejecutarse.

Opciones de monitoreo de AWS

Le recomendamos que utilice las siguientes opciones de monitoreo de AWS para solucionar el problema:

- Amazon CloudWatch: Amazon CloudWatch monitorea los recursos de AWS y las aplicaciones que ejecuta en AWS en tiempo real. Puede utilizar CloudWatch para recopilar y hacer un seguimiento de métricas, que son las variables que puede medir en los recursos y aplicaciones. Para obtener más información, consulte ¿Qué es Amazon CloudWatch?.

  Puede ver todas las métricas del sistema y la información de los procesos de sus instancias de base de datos en la Consola de administración de AWS. Puede configurar su clúster de base de datos Aurora MySQL para publicar datos de registros generales, lentos, de auditoría y de errores en un grupo de registros en Amazon CloudWatch Logs. Aquí puede ver las tendencias, mantener registros en caso de que un host se vea afectado y crear una base de referencia del rendimiento “normal” con el fin de identificar fácilmente las anomalías o los cambios. Para obtener más información, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.
- Monitoreo mejorado: para habilitar métricas adicionales de Amazon CloudWatch para una base de datos Aurora MySQL, active el monitoreo mejorado. Al crear o modificar un clúster de base de datos de Aurora, seleccione Habilitar la monitorización mejorada. Esto permite a Aurora publicar métricas de rendimiento en CloudWatch. Algunas de las métricas clave disponibles incluyen el uso de la CPU, las conexiones a las bases de datos, el uso del almacenamiento y la latencia de las consultas. Estas métricas pueden ayudarle a identificar los cuellos de botella de rendimiento.

  La cantidad de información transferida para una instancia de base de datos es directamente proporcional a la granularidad definida en el monitoreo mejorado. Un intervalo de monitorización

más corto deriva en informes más frecuentes de métricas del SO y aumenta el costo de la monitorización. Para administrar los costos, establezca diferentes granularidades para diferentes instancias en sus Cuentas de AWS. La granularidad predeterminada al crear una instancia es de 60 segundos. Para obtener más información, consulte Costo de la monitorización mejorada.

- Información de rendimiento: puede ver todas las métricas de las llamadas de la base de datos. Esto incluye los bloqueos de la base de datos, las esperas y el número de filas procesadas, todo lo cual puede utilizar para solucionar problemas. Al crear o modificar un clúster de base de datos de Aurora, seleccione Activar Información de rendimiento. De forma predeterminada, Información de rendimiento tiene un período de retención de datos de 7 días, pero se puede personalizar para analizar las tendencias de rendimiento a largo plazo. Si desea una retención superior a 7 días, debe actualizar el nivel de pago. Para obtener más información, consulte Precio de Información de rendimiento. Puede configurar el período de retención de datos para cada instancia de base de datos de Aurora por separado. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

Motivos más comunes de los problemas de rendimiento de bases de datos Aurora MySQL

Puede utilizar los siguientes pasos para solucionar problemas de rendimiento en la base de datos Aurora MySQL. Indicamos los pasos en el orden lógico de investigación, pero no tienen por qué ser lineales. Al realizar un descubrimiento, podría ser necesario saltar a diferentes pasos, por lo que se permiten muchas rutas de investigación.

1. Carga de trabajo: comprenda la carga de trabajo de su base de datos.
2. Registro: revise todos los registros de la base de datos.
3. Conexiones a bases de datos: asegúrese de que las conexiones entre las aplicaciones y la base de datos sean fiables.
4. Rendimiento de consultas: examine sus planes de ejecución de consultas para ver si han cambiado. Los cambios en el código pueden provocar cambios en los planes.

Solución de problemas de carga de trabajo de bases de datos Aurora MySQL

La carga de trabajo de la base de datos se puede ver como lecturas y escrituras. Si comprende cuál es la carga de trabajo “normal” de una base de datos, podrá ajustar las consultas y el servidor de base de datos para adaptarlos a la demanda a medida que esta vaya cambiando. Existen varios motivos por los que el rendimiento puede cambiar, por lo que el primer paso es entender qué ha cambiado.

- ¿Se ha realizado una actualización de una versión principal o secundaria?

  Una actualización de la versión principal incluye cambios en el código del motor, especialmente en el optimizador, que pueden cambiar el plan de ejecución de las consultas. Al actualizar las versiones de la base de datos, especialmente las versiones principales, es muy importante analizar la carga de trabajo de la base de datos y ajustarla en consecuencia. Este ajuste puede implicar optimizar y reescribir las consultas o agregar y actualizar la configuración de los parámetros, en función de los resultados de las pruebas. Entender qué es lo que está causando ese efecto le permitirá empezar a centrarse en esa área específica.

  Para obtener más información, consulte What is new in MySQL 8.0 y Server and status variables and options added, deprecated, or removed in MySQL 8.0 en la documentación de MySQL y Comparación entre Aurora MySQL versión 2 y Aurora MySQL versión 3.
- ¿Se ha producido un aumento en el procesamiento de datos (número de filas)?
- ¿Hay más consultas ejecutándose simultáneamente?
- ¿Hay cambios en el esquema o en la base de datos?
- ¿Se han producido errores o correcciones en el código?

Contenido

- Métricas de host de la instancia
  - Uso de la CPU
  - Uso de memoria
  - Network throughput
- Métricas de bases de datos
- Solución de problemas de uso de memoria de bases de datos Aurora MySQL
  - Ejemplo 1: uso elevado y continuo de memoria

  - Ejemplo 2: picos de memoria de transitorios
  - Ejemplo 3: la memoria liberable cae continuamente y no se recupera
- Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL

Métricas de host de la instancia

Supervise las métricas del host de la instancia, como la actividad de la CPU, la memoria y la red, para saber si se ha producido un cambio en la carga de trabajo. Existen dos conceptos principales para entender los cambios en la carga de trabajo:

- Utilización: el uso de un dispositivo, como la CPU o el disco. Puede basarse en el tiempo o en la capacidad.
  - Basado en el tiempo: la cantidad de tiempo que un recurso está ocupado durante un período de observación determinado.
  - Basado en la capacidad: la cantidad de rendimiento que puede ofrecer un sistema o componente, expresado como porcentaje de su capacidad.
- Saturación: hasta qué punto se requiere más trabajo de un recurso del que puede procesar. Cuando el uso basado en la capacidad alcanza el 100 %, el trabajo adicional no se puede procesar y debe ponerse en cola.

Uso de la CPU

Puede utilizar las siguientes herramientas para identificar el uso y la saturación de la CPU:

- CloudWatch proporciona la métrica `CPUUtilization`. Si llega al 100 %, la instancia está saturada. Sin embargo, las métricas de CloudWatch tienen un promedio de más de 1 minuto y carecen de granularidad.

  Para obtener más información sobre las métricas de CloudWatch, consulte Métricas de nivel de instancia para Amazon Aurora.
- El monitoreo mejorado proporciona métricas que devuelve el comando del sistema operativo `top`. Muestra los promedios de carga y los siguientes estados de la CPU, con una granularidad de 1 segundo:
  - `Idle (%)` = tiempo de inactividad
  - `IRQ (%)` = interrupciones de software
  - `Nice (%)` = tiempo nice para los procesos con una prioridad niced.

- `Steal (%)` = tiempo dedicado a atender a otros inquilinos (relacionado con la virtualización)
- `System (%)` = tiempo del sistema
- `User (%)` = tiempo de usuario
- `Wait (%)` = espera de E/S

Para obtener más información acerca de las métricas de Supervisión mejorada, consulte Métricas del sistema operativo para Aurora.

Uso de memoria

Si al sistema le falta memoria y el consumo de recursos está a punto de saturarse, debería haber un alto grado de errores de análisis de páginas, paginación, intercambio y falta de memoria.

Puede utilizar las siguientes herramientas para identificar el uso y la saturación de la memoria:

CloudWatch proporciona la métrica `FreeableMemory`, que muestra la cantidad de memoria que se puede recuperar vaciando algunas de las cachés del sistema operativo y la memoria libre actual.

Para obtener más información sobre las métricas de CloudWatch, consulte Métricas de nivel de instancia para Amazon Aurora.

El monitoreo mejorado proporciona las siguientes métricas que pueden ayudarle a identificar los problemas de uso de la memoria:

- `Buffers (KB)`: la cantidad de memoria utilizada para almacenar en búfer solicitudes de E/S antes de escribir en el dispositivo de almacenamiento, en kilobytes.
- `Cached (KB)`: la cantidad de memoria utilizada para almacenar en la caché las E/S basadas en el sistema de archivos.
- `Free (KB)`: la cantidad de memoria no asignada, en kilobytes.
- `Swap`: en caché, gratis y total.

Por ejemplo, si ve que la instancia de base de datos utiliza memoria `Swap`, la cantidad total de memoria para su carga de trabajo es mayor de la que la instancia tiene disponible actualmente. Le recomendamos aumentar el tamaño de la instancia de base de datos o ajustar la carga de trabajo para utilizar menos memoria.

Para obtener más información acerca de las métricas de Supervisión mejorada, consulte Métricas del sistema operativo para Aurora.

Para obtener información más detallada sobre el uso de Performance Schema y el esquema de `sys` para determinar qué conexiones y componentes utilizan memoria, consulte Solución de problemas de uso de memoria de bases de datos Aurora MySQL.

Network throughput

CloudWatch proporciona las siguientes métricas para el rendimiento total de la red, todas con un promedio de más de 1 minuto:

- `NetworkReceiveThroughput`: la cantidad de rendimiento de red que recibe de los clientes cada instancia en el clúster de base de datos de Aurora.
- `NetworkTransmitThroughput`: el rendimiento de red que envía a los clientes cada instancia del clúster de base de datos de Aurora.
- `NetworkThroughput`: el rendimiento de red que recibe de los clientes y transmite a ellos cada instancia en el clúster de base de datos de Aurora.
- `StorageNetworkReceiveThroughput`: el rendimiento de red que se recibe del subsistema de almacenamiento de Aurora por cada instancia del clúster de base de datos.
- `StorageNetworkTransmitThroughput`: el rendimiento de red que se envía al subsistema de almacenamiento de Aurora por cada instancia en el clúster de base de datos de Aurora.
- `StorageNetworkThroughput`: el rendimiento de red que se recibe del subsistema de almacenamiento de Aurora y se envía a este por cada instancia en el clúster de base de datos de Aurora.

Para obtener más información sobre las métricas de CloudWatch, consulte Métricas de nivel de instancia para Amazon Aurora.

El monitoreo mejorado proporciona los gráficos de `network` recibidos (RX) y transmitidos (TX), con una granularidad de hasta 1 segundo.

Para obtener más información acerca de las métricas de Supervisión mejorada, consulte Métricas del sistema operativo para Aurora.

Métricas de bases de datos

Examine las siguientes métricas de CloudWatch para ver si hay cambios en la carga de trabajo:

- `BlockedTransactions`: número medio de transacciones de la base de datos que se bloquean cada segundo.

- `BufferCacheHitRatio`: porcentaje de solicitudes que se responden desde la caché de búfer.
- `CommitThroughput`: número medio de operaciones de confirmación por segundo.
- `DatabaseConnections`: número de conexiones de red de cliente a la instancia de base de datos.
- `Deadlocks`: número medio de interbloqueos en la base de datos por segundo.
- `DMLThroughput`: número medio de inserciones, actualizaciones y eliminaciones por segundo.
- `ResultSetCacheHitRatio`: porcentaje de solicitudes que se responden desde la caché de consultas.
- `RollbackSegmentHistoryListLength`: registros redo que registran transacciones confirmadas con registros marcados para su eliminación.
- `RowLockTime`: tiempo total dedicado a adquirir bloqueos de fila para tablas de InnoDB.
- `SelectThroughput`: número medio de consultas de selección por segundo.

Para obtener más información sobre las métricas de CloudWatch, consulte Métricas de nivel de instancia para Amazon Aurora.

Tenga en cuenta las siguientes preguntas al examinar la carga de trabajo:

1. ¿Se han producido cambios recientes en la clase de la instancia de base de datos, por ejemplo, se ha reducido el tamaño de la instancia de 8xlarge a 4xlarge o se ha cambiado de db.r5 a db.r6?
2. ¿Puede crear un clon y reproducir el problema o solo ocurre en esa instancia?
3. ¿Se agotan los recursos del servidor? ¿La CPU o la memoria sufren un gran agotamiento? En caso afirmativo, esto podría significar que se requiere hardware adicional.
4. ¿Una o más consultas están tardando más tiempo?
5. ¿Los cambios se deben a una actualización, especialmente a una actualización de una versión principal? En caso afirmativo, compare las métricas previas y posteriores a la actualización.
6. ¿Hay cambios en la cantidad de instancias de base de datos de lector?
7. ¿Ha habilitado el registro general, de auditoría o binario? Para obtener más información, consulte Registro de bases de datos Aurora MySQL.
8. ¿Ha habilitado, deshabilitado o cambiado el uso de la replicación de registros binarios (binlog)?
9. ¿Hay transacciones de larga duración que contengan un gran número de bloqueos de filas? Examine la longitud de la lista de historial (HLL) de InnoDB para ver si hay indicios de transacciones de larga duración.

Para obtener más información, consulte La longitud de la lista de historial de InnoDB ha aumentado de forma significativa y la entrada del blog Why is my SELECT query running slowly on my Amazon Aurora MySQL DB clúster?

a. Si una HLL de gran tamaño se debe a una transacción de escritura, eso significa que los registros UNDO se están acumulando (no se limpian con regularidad). En una transacción de escritura de gran tamaño, esta acumulación puede aumentar rápidamente. En MySQL, UNDO se almacena en el espacio de tabla SYSTEM. El espacio de tabla SYSTEM no se puede reducir. El registro UNDO puede hacer que el espacio de tabla SYSTEM aumente varios GB o incluso TB. Tras la purga, libere el espacio asignado realizando una copia de seguridad lógica (volcado) de los datos y, a continuación, importe el volcado a una nueva instancia de base de datos.

b. Si una HLL de gran tamaño se debe a una transacción de lectura (consulta de larga duración), eso puede significar que la consulta utiliza una gran cantidad de espacio temporal. Reinicie para liberar el espacio temporal. Examine las métricas de la base de datos de Información de rendimiento para ver si se ha producido algún cambio en la sección Temp, por ejemplo, `created_tmp_tables`. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

10. ¿Se pueden dividir las transacciones de larga duración en otras más pequeñas que modifiquen menos filas?

11. ¿Se han producido cambios en las transacciones bloqueadas o han aumentado los interbloqueos? Examine las métricas de base de datos de Información de rendimiento para detectar cualquier cambio en las variables de estado en la sección Locks, como `innodb_row_lock_time`, `innodb_row_lock_waits` y `innodb_dead_locks`. Use intervalos de 1 o 5 minutos.

12. ¿Hay un aumento de los eventos de espera? Examine los eventos de espera y los tipos de espera de Información de rendimiento a intervalos de 1 o 5 minutos. Analice los principales eventos de espera y compruebe si están relacionados con los cambios en la carga de trabajo o con la contención de la base de datos. Por ejemplo, `buf_pool mutex` indica una contención del grupo de búferes. Para obtener más información, consulte Ajuste de Aurora MySQL con eventos de espera.

Solución de problemas de uso de memoria de bases de datos Aurora MySQL

Si bien CloudWatch, Supervisión mejorada e Información de rendimiento proporcionan una visión general óptima del uso de la memoria en el sistema operativo, por ejemplo, la cantidad de memoria que utiliza el proceso de la base de datos, no permiten desglosar qué conexiones o componentes del motor podrían estar causando este uso de memoria.

Para solucionar este problema, puede utilizar Performance Schema y el esquema de `sys`. En Aurora MySQL versión 3, la instrumentación de memoria se habilita de forma predeterminada cuando se habilita Performance Schema. En Aurora MySQL versión 2, solo se habilita por defecto la instrumentación de memoria para el uso de memoria de Performance Schema. Para obtener información sobre las tablas disponibles en Performance Schema para realizar un seguimiento del uso de la memoria y habilitar la instrumentación de memoria de Performance Schema, consulte la tablas de resumen de memoria en la documentación de MySQL. Para obtener más información sobre el uso de Performance Schema con Información de rendimiento, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

Si bien se encuentra disponible información detallada en Performance Schema para realizar un seguimiento del uso actual de la memoria, el esquema sys de MySQL tiene vistas en la parte superior de las tablas de Performance Schema que puede utilizar para identificar rápidamente dónde se utiliza la memoria.

En el esquema `sys`, están disponibles las siguientes vistas para realizar un seguimiento del uso de la memoria por conexión, componente y consulta.

Visualización	Descripción
memory_by_host_by_current_bytes	Proporciona información sobre el uso de la memoria del motor por el host. Esto puede resultar útil para identificar qué servidores de aplicaciones o hosts de clientes consumen memoria.
memory_by_thread_by_current_bytes	Proporciona información sobre el uso de la memoria del motor por ID de subproceso. El ID de subproceso en MySQL puede ser una conexión de cliente o un subproceso en segundo plano. Puede asignar ID de subprocesos a ID de conexión de MySQL mediante la vista sys.processlist o la tabla performance_schema.threads.
memory_by_user_by_current_bytes	Proporciona información sobre el uso de la memoria del motor por usuario. Esto puede

Visualización	Descripción
	resultar útil para identificar qué cuentas de usuario o clientes consumen memoria.
memory_global_by_current_bytes	Proporciona información sobre el uso de la memoria del motor por componente del motor. Esto puede resultar útil para identificar el uso de memoria global por parte de búferes o componentes del motor. Por ejemplo, es posible que vea el evento `memory/innodb/buf_buf_pool` del conjunto de búferes de InnoDB o el evento `memory/sql/Prepared_statement::main_mem_root` de las instrucciones preparadas.
memory_global_total	Proporciona una descripción general del uso total de memoria del que se ha hecho un seguimiento en el motor de base de datos.

En Aurora MySQL versión 3.05 y versiones posteriores, también puede realizar un seguimiento del uso de memoria máximo mediante resumen de instrucciones en las tablas de resumen de instrucciones de Performance Schema. Las tablas de resumen de instrucciones contienen resúmenes normalizados de instrucciones y estadísticas agregadas sobre su ejecución. La columna `MAX_TOTAL_MEMORY` puede ayudarle a identificar la memoria máxima utilizada por el resumen de consultas desde la última vez que se restablecieron las estadísticas o desde que se reinició la instancia de la base de datos. Esto puede resultar útil para identificar consultas específicas que podrían estar consumiendo mucha memoria.

> **Note**
>
> Performance Schema y el esquema `sys` muestran el uso actual de memoria en el servidor y los niveles máximos de memoria consumida por conexión y componente del motor. Como Performance Schema se mantiene en la memoria, la información se restablece cuando se reinicia la instancia de base de datos. Para mantener un historial a lo largo del tiempo, le

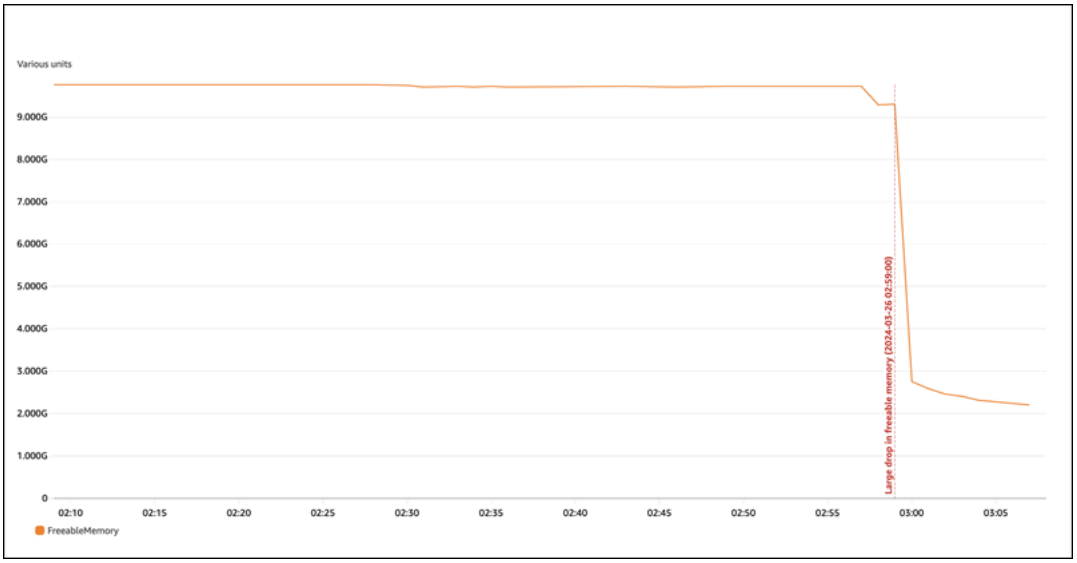
recomendamos que configure la recuperación y el almacenamiento de estos datos fuera de Performance Schema.

Temas

- Ejemplo 1: uso elevado y continuo de memoria
- Ejemplo 2: picos de memoria de transitorios
- Ejemplo 3: la memoria liberable cae continuamente y no se recupera

Ejemplo 1: uso elevado y continuo de memoria

Si observamos globalmente `FreeableMemory` en CloudWatch, podemos ver que el uso de memoria ha aumentado en gran medida a las 02:59 UTC del 26 de marzo de 2024.



Esto no nos muestra el panorama completo. Para determinar qué componente está consumiendo más memoria, puede iniciar sesión en la base de datos y ver `sys.memory_global_by_current_bytes`. Esta tabla contiene una lista de eventos de memoria de los que hace seguimiento MySQL, junto con información sobre la asignación de memoria por evento. Cada evento de seguimiento de memoria comienza con `memory/%`, seguido de otra información sobre el componente o la característica del motor al que está asociado el evento.

Por ejemplo, `memory/performance_schema/%` es para eventos de memoria relacionados con Performance Schema, `memory/innodb/%` es para InnoDB, etc. Para obtener información sobre las convenciones de nomenclatura de los eventos, consulte Performance Schema Instrument Naming Conventions en la documentación de MySQL.

A partir de la siguiente consulta, podemos encontrar al probable culpable en función de `current_alloc`, pero también podemos ver muchos eventos de `memory/performance_schema/%`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+--------------+--------------+------------------+-----------+-----------
+----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-------------------------------------------------------------------------------
+--------------+--------------+------------------+-----------+-----------
+----------------+
| memory/sql/Prepared_statement::main_mem_root                                  |
 512817 | 4.91 GiB      | 10.04 KiB         |     512823 | 4.91 GiB   | 10.04 KiB
 |
| memory/performance_schema/prepared_statements_instances                       |
   252 | 488.25 MiB    | 1.94 MiB          |        252 | 488.25 MiB | 1.94 MiB       |
| memory/innodb/hash0hash                                                       |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error            |
  1028 | 52.27 MiB     | 52.06 KiB         |       1028 | 52.27 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     4 | 47.25 MiB     | 11.81 MiB         |          4 | 47.25 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/performance_schema/memory_summary_by_thread_by_event_name              |
     4 | 31.64 MiB     | 7.91 MiB          |          4 | 31.64 MiB  | 7.91 MiB       |
| memory/innodb/memory                                                          |
 15227 | 27.44 MiB     | 1.85 KiB          |      20619 | 33.33 MiB  | 1.66 KiB       |
| memory/sql/String::value                                                      |
 74411 | 21.85 MiB     |  307 bytes        |      76867 | 25.54 MiB  |  348 bytes     |
| memory/sql/TABLE                                                              |
  8381 | 21.03 MiB     | 2.57 KiB          |       8381 | 21.03 MiB  | 2.57 KiB       |
+-------------------------------------------------------------------------------
+--------------+--------------+------------------+-----------+-----------
+----------------+
10 rows in set (0.02 sec)
```

Mencionamos anteriormente que Performance Schema se almacena en la memoria, lo que significa que también se realiza un seguimiento de ello en la instrumentación de memoria de `performance_schema`.

> **Note**
>
> Si observa que Performance Schema utiliza mucha memoria y quiere limitar su uso, puede ajustar los parámetros de la base de datos en función de sus necesidades. Para obtener más información, consulte The Performance Schema memory-allocation model en la documentación de MySQL.

Para facilitar la lectura, puede volver a ejecutar la misma consulta pero excluir los eventos de Performance Schema. En el resultado se observa lo siguiente:

- El elemento que consume más memoria es `memory/sql/Prepared_statement::main_mem_root`.
- La columna `current_alloc` nos indica que MySQL tiene 4,91 GiB actualmente asignados a este evento.
- `high_alloc column` nos indica que 4,91 GiB es el nivel máximo de `current_alloc` desde que se restablecieron las estadísticas por última vez o desde que se reinició el servidor. Esto significa que `memory/sql/Prepared_statement::main_mem_root` está en su valor más alto.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name NOT LIKE
 'memory/performance_schema/%' LIMIT 10;

+-------------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                      | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-------------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root    |        512817 | 4.91 GiB      | 10.04
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
| memory/innodb/hash0hash                         |             4 | 79.07 MiB     | 19.77
 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/memory                            |         17096 | 31.68 MiB     | 1.90
 KiB          |      22498 | 37.60 MiB  | 1.71 KiB       |
```

```
| memory/sql/String::value                        |        122277 | 27.94 MiB     |  239
 bytes        |     124699 | 29.47 MiB  |  247 bytes     |
| memory/sql/TABLE                                |          9927 | 24.67 MiB     | 2.55
 KiB          |       9929 | 24.68 MiB  | 2.55 KiB       |
| memory/innodb/lock0lock                         |          8888 | 19.71 MiB     | 2.27
 KiB          |       8888 | 19.71 MiB  | 2.27 KiB       |
| memory/sql/Prepared_statement::infrastructure |        257623 | 16.24 MiB     |   66
 bytes        |     257631 | 16.24 MiB  |   66 bytes     |
| memory/mysys/KEY_CACHE                          |             3 | 16.00 MiB     | 5.33
 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/innodb/sync0arr                          |             3 | 7.03 MiB      | 2.34
 MiB          |          3 | 7.03 MiB   | 2.34 MiB       |
| memory/sql/THD::main_mem_root                   |           815 | 6.56 MiB      | 8.24
 KiB          |        849 | 7.19 MiB   | 8.67 KiB       |
+-------------------------------------------------+---------------+---------------
+------------------+-----------+-----------+----------------+
10 rows in set (0.06 sec)
```

Por el nombre del evento, podemos decir que esta memoria se está utilizando para instrucciones preparadas. Si quiere ver qué conexiones utilizan esta memoria, puede consultar memory_by_thread_by_current_bytes.

En el siguiente ejemplo, cada conexión tiene aproximadamente 7 MiB asignados, con un nivel máximo de aproximadamente 6,29 MiB (`current_max_alloc`). Esto tiene sentido, ya que el ejemplo utiliza `sysbench` con 80 tablas y 800 conexiones con instrucciones preparadas. Si desea reducir el uso de memoria en este caso, puede optimizar el uso que hace su aplicación de las instrucciones preparadas para reducir el consumo de memoria.

```
mysql> SELECT * FROM sys.memory_by_thread_by_current_bytes;

+-----------+-------------------------------------------+-------------------
+------------------+------------------+------------------+-----------------+
| thread_id | user                                      | current_count_used |
 current_allocated | current_avg_alloc | current_max_alloc | total_allocated |
+-----------+-------------------------------------------+-------------------
+------------------+------------------+------------------+-----------------+
|        46 | rdsadmin@localhost                        |                405 | 8.47 MiB
          | 21.42 KiB         | 8.00 MiB          | 155.86 MiB      |
|        61 | reinvent@10.0.4.4                         |               1749 | 6.72 MiB
          | 3.93 KiB          | 6.29 MiB          | 14.24 MiB       |
|       101 | reinvent@10.0.4.4                         |               1845 | 6.71 MiB
          | 3.72 KiB          | 6.29 MiB          | 14.50 MiB       |
```

```
|        55 | reinvent@10.0.4.4                        |              1674 | 6.68 MiB
          | 4.09 KiB          | 6.29 MiB          | 14.13 MiB       |
|        57 | reinvent@10.0.4.4                        |              1416 | 6.66 MiB
          | 4.82 KiB          | 6.29 MiB          | 13.52 MiB       |
|       112 | reinvent@10.0.4.4                        |              1759 | 6.66 MiB
          | 3.88 KiB          | 6.29 MiB          | 14.17 MiB       |
|        66 | reinvent@10.0.4.4                        |              1428 | 6.64 MiB
          | 4.76 KiB          | 6.29 MiB          | 13.47 MiB       |
|        75 | reinvent@10.0.4.4                        |              1389 | 6.62 MiB
          | 4.88 KiB          | 6.29 MiB          | 13.40 MiB       |
|       116 | reinvent@10.0.4.4                        |              1333 | 6.61 MiB
          | 5.08 KiB          | 6.29 MiB          | 13.21 MiB       |
|        90 | reinvent@10.0.4.4                        |              1448 | 6.59 MiB
          | 4.66 KiB          | 6.29 MiB          | 13.58 MiB       |
|        98 | reinvent@10.0.4.4                        |              1440 | 6.57 MiB
          | 4.67 KiB          | 6.29 MiB          | 13.52 MiB       |
|        94 | reinvent@10.0.4.4                        |              1433 | 6.57 MiB
          | 4.69 KiB          | 6.29 MiB          | 13.49 MiB       |
|        62 | reinvent@10.0.4.4                        |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.48 MiB       |
|        87 | reinvent@10.0.4.4                        |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.25 MiB       |
|        99 | reinvent@10.0.4.4                        |              1346 | 6.54 MiB
          | 4.98 KiB          | 6.29 MiB          | 13.24 MiB       |
|       105 | reinvent@10.0.4.4                        |              1347 | 6.54 MiB
          | 4.97 KiB          | 6.29 MiB          | 13.34 MiB       |
|        73 | reinvent@10.0.4.4                        |              1335 | 6.54 MiB
          | 5.02 KiB          | 6.29 MiB          | 13.23 MiB       |
|        54 | reinvent@10.0.4.4                        |              1510 | 6.53 MiB
          | 4.43 KiB          | 6.29 MiB          | 13.49 MiB       |
.                                                                                .
.                                                                                .
.                                                                                .
|       812 | reinvent@10.0.4.4                        |              1259 | 6.38 MiB
          | 5.19 KiB          | 6.29 MiB          | 13.05 MiB       |
|       214 | reinvent@10.0.4.4                        |              1279 | 6.38 MiB
          | 5.10 KiB          | 6.29 MiB          | 12.90 MiB       |
|       325 | reinvent@10.0.4.4                        |              1254 | 6.38 MiB
          | 5.21 KiB          | 6.29 MiB          | 12.99 MiB       |
|       705 | reinvent@10.0.4.4                        |              1273 | 6.37 MiB
          | 5.13 KiB          | 6.29 MiB          | 13.03 MiB       |
```

```
|       530 | reinvent@10.0.4.4                          |               1268 | 6.37 MiB
          | 5.15 KiB          | 6.29 MiB          | 12.92 MiB       |
|       307 | reinvent@10.0.4.4                          |               1263 | 6.37 MiB
          | 5.17 KiB          | 6.29 MiB          | 12.87 MiB       |
|       738 | reinvent@10.0.4.4                          |               1260 | 6.37 MiB
          | 5.18 KiB          | 6.29 MiB          | 13.00 MiB       |
|       819 | reinvent@10.0.4.4                          |               1252 | 6.37 MiB
          | 5.21 KiB          | 6.29 MiB          | 13.01 MiB       |
|        31 | innodb/srv_purge_thread                    |              17810 | 3.14 MiB
          |  184 bytes        | 2.40 MiB          | 205.69 MiB      |
|        38 | rdsadmin@localhost                         |                599 | 1.76 MiB
          | 3.01 KiB          | 1.00 MiB          | 25.58 MiB       |
|         1 | sql/main                                   |               3756 | 1.32 MiB
          |  367 bytes        | 355.78 KiB        | 6.19 MiB        |
|       854 | rdsadmin@localhost                         |                 46 | 1.08 MiB
          | 23.98 KiB         | 1.00 MiB          | 5.10 MiB        |
|        30 | innodb/clone_gtid_thread                   |               1596 | 573.14
 KiB        |  367 bytes        | 254.91 KiB        | 970.69 KiB      |
|        40 | rdsadmin@localhost                         |                235 | 245.19
 KiB        | 1.04 KiB          | 128.88 KiB        | 808.64 KiB      |
|       853 | rdsadmin@localhost                         |                 96 | 94.63
 KiB         | 1009 bytes        | 29.73 KiB         | 422.45 KiB      |
|        36 | rdsadmin@localhost                         |                 33 | 36.29
 KiB         | 1.10 KiB          | 16.08 KiB         | 74.15 MiB       |
|        33 | sql/event_scheduler                        |                  3 | 16.27
 KiB         | 5.42 KiB          | 16.04 KiB         | 16.27 KiB       |
|        35 | sql/compress_gtid_table                    |                  8 | 14.20
 KiB         | 1.77 KiB          | 8.05 KiB          | 18.62 KiB       |
|        25 | innodb/fts_optimize_thread                 |                 12 | 1.86 KiB
          |  158 bytes        |  648 bytes        | 1.98 KiB        |
|        23 | innodb/srv_master_thread                   |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 24.40 KiB       |
|        24 | innodb/dict_stats_thread                   |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 1.35 KiB        |
|         5 | innodb/io_read_thread                      |                  1 |  144
 bytes         |  144 bytes        |  144 bytes        |  144 bytes      |
|         6 | innodb/io_read_thread                      |                  1 |  144
 bytes         |  144 bytes        |  144 bytes        |  144 bytes      |
|         2 | sql/aws_oscar_log_level_monitor            |                  0 |    0
 bytes         |    0 bytes        |    0 bytes        |    0 bytes      |
|         4 | innodb/io_ibuf_thread                      |                  0 |    0
 bytes         |    0 bytes        |    0 bytes        |    0 bytes      |
|         7 | innodb/io_write_thread                     |                  0 |    0
 bytes         |    0 bytes        |    0 bytes        |    0 bytes      |
```

```
|         8 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         9 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        10 | innodb/io_write_thread                     |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        11 | innodb/srv_lra_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        12 | innodb/srv_akp_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        18 | innodb/srv_lock_timeout_thread             |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |  248 bytes      |
|        19 | innodb/srv_error_monitor_thread            |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        20 | innodb/srv_monitor_thread                  |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        21 | innodb/buf_resize_thread                   |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        22 | innodb/btr_search_sys_toggle_thread        |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        32 | innodb/dict_persist_metadata_table_thread  |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        34 | sql/signal_handler                         |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
+-----------+--------------------------------------------+--------------------
+-------------------+------------------+-------------------+-----------------+
831 rows in set (2.48 sec)
```

Como se mencionó antes, el valor de ID de subproceso (`thd_id`) aquí puede hacer referencia a subprocesos en segundo plano del servidor o a conexiones de bases de datos. Si desea asignar valores de ID de subprocesos a ID de conexión a la base de datos, puede utilizar la tabla `performance_schema.threads` o la vista `sys.processlist`, donde `conn_id` es el ID de conexión.

```
mysql> SELECT thd_id,conn_id,user,db,command,state,time,last_wait FROM sys.processlist
 WHERE user='reinvent@10.0.4.4';

+--------+---------+------------------+----------+---------+----------------+------
+-------------------------------------------------+
| thd_id | conn_id | user             | db       | command | state          | time |
 last_wait                                        |
+--------+---------+------------------+----------+---------+----------------+------
+-------------------------------------------------+
```

```
|    590 |     562 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                          |
|    578 |     550 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                            |
|    579 |     551 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                          |
|    580 |     552 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                       |
|    581 |     553 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                       |
|    582 |     554 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                            |
|    583 |     555 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                            |
|    584 |     556 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                       |
|    585 |     557 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                          |
|    586 |     558 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                       |
|    587 |     559 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                          |
.
                                                 .
.
                                                 .
.
                                                 .
|    323 |     295 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                            |
|    324 |     296 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                       |
|    325 |     297 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                          |
|    326 |     298 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                       |
|    438 |     410 | reinvent@10.0.4.4 | sysbench | Execute | System lock    |    0 |
 wait/lock/table/sql/handler                     |
|    280 |     252 | reinvent@10.0.4.4 | sysbench | Sleep   | starting       |    0 |
 wait/io/socket/sql/client_connection            |
|     98 |      70 | reinvent@10.0.4.4 | sysbench | Query   | freeing items  |    0 |
 NULL                                            |
+--------+---------+-------------------+----------+---------+----------------+------
+-----------------------------------------------+
```

```
804 rows in set (5.51 sec)
```

Ahora detenemos la carga de trabajo de `sysbench`, lo que cierra las conexiones y libera memoria. Al volver a comprobar los eventos, podemos confirmar que la memoria se ha liberado, pero `high_alloc` nos sigue indicando cuál es el nivel máximo. La columna `high_alloc` puede resultar muy útil para identificar picos breves en el uso de memoria, cuando no se pueda identificar inmediatamente el uso de `current_alloc`, ya que muestra solo la memoria actualmente asignada.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Si desea restablecer `high_alloc`, puede truncar las tablas de resumen de la memoria de `performance_schema`, pero esto restablece toda la instrumentación de la memoria. Para obtener más información, consulte Performance Schema general table characteristics en la documentación de MySQL.

En el siguiente ejemplo, podemos observar que `high_alloc` se restablece tras el truncado.

```
mysql> TRUNCATE `performance_schema`.`memory_summary_global_by_event_name`;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
```

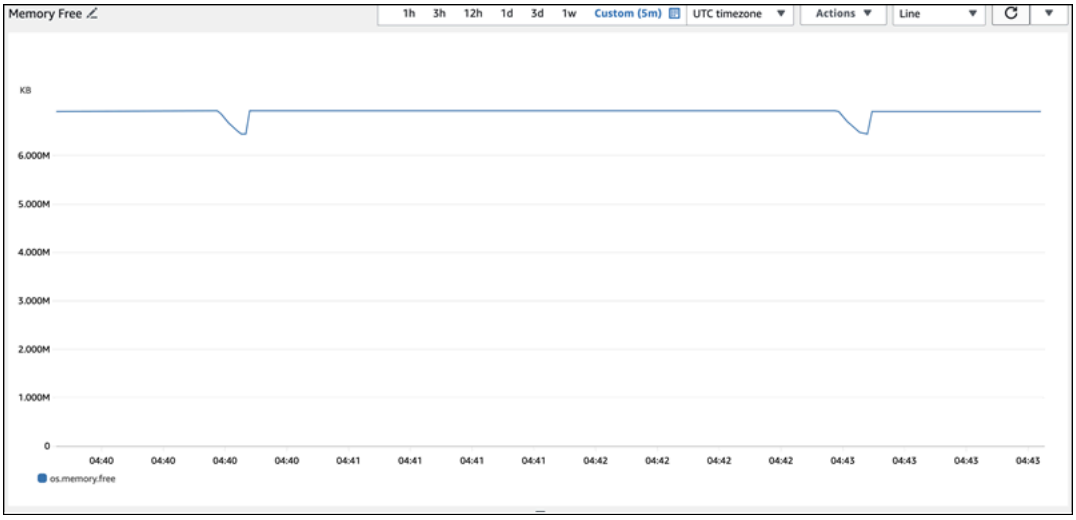
```
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |         17 | 253.80 KiB | 14.93 KiB      |
+----------------------------------------------+---------------+---------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Ejemplo 2: picos de memoria de transitorios

Los picos breves en el uso de memoria son algo habitual en un servidor de base de datos. Puede tratarse de descensos periódicos de la memoria que se puede liberar y que son difíciles solucionar mediante `current_alloc` en `sys.memory_global_by_current_bytes`, ya que la memoria ya se ha liberado.

> **Note**
>
> Si se han restablecido las estadísticas de Performance Schema o se ha reiniciado la instancia de la base de datos, esta información no estará disponible en `sys` o `performance_schema`. Para conservar esta información, recomendamos que configure la recopilación de métricas externas.

El siguiente gráfico de la métrica de `os.memory.free` en Supervisión mejorada muestra breves picos de 7 segundos en el uso de la memoria. Supervisión mejorada permite supervisar a intervalos de tan solo un segundo, lo que resulta perfecto para detectar picos transitorios como estos.



Para ayudar a diagnosticar la causa del uso de memoria en este caso, podemos utilizar una combinación de `high_alloc` en las vistas de resumen de memoria de `sys` y las tablas de resumen

de instrucciones de Performance Schema para tratar de identificar las sesiones y conexiones problemáticas.

Como era de esperar, dado que el uso de memoria no es elevado actualmente, no podemos detectar ningún infractor principal en la vista del esquema de `sys` bajo `current_alloc`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| memory/innodb/hash0hash                                                       |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/os0event                                                        |
 439372 | 60.34 MiB     |  144 bytes        |     439372 | 60.34 MiB  |  144 bytes
 |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/mysys/KEY_CACHE                                                        |
     3 | 16.00 MiB     | 5.33 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/performance_schema/events_statements_history_long                      |
     1 | 14.34 MiB     | 14.34 MiB         |          1 | 14.34 MiB  | 14.34 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error            |
   257 | 13.07 MiB     | 52.06 KiB         |        257 | 13.07 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     1 | 11.81 MiB     | 11.81 MiB         |          1 | 11.81 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.digest_text         |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.sql_text            |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
10 rows in set (0.01 sec)
```

Al ampliar la vista para ordenar por `high_alloc`, ahora podemos ver que el componente `memory/temptable/physical_ram` es muy buen candidato en este caso. En su nivel más alto, consumía 515,00 MiB.

Tal y como indica su nombre, `memory/temptable/physical_ram` instrumenta el uso de memoria del motor de almacenamiento TEMP en MySQL, que se introdujo en MySQL 8.0. Para obtener más información sobre cómo utiliza MySQL las tablas temporales, consulte Internal temporary table use in MySQL en la documentación de MySQL.

> **Note**
>
> En este ejemplo, estamos utilizando la vista `sys.x$memory_global_by_current_bytes`.

```
mysql> SELECT event_name, format_bytes(current_alloc) AS "currently allocated",
 sys.format_bytes(high_alloc) AS "high-water mark"
FROM sys.x$memory_global_by_current_bytes ORDER BY high_alloc DESC LIMIT 10;

+-------------------------------------------------------------------------------
+---------------------+-----------------+
| event_name                                                                    |
 currently allocated | high-water mark |
+-------------------------------------------------------------------------------
+---------------------+-----------------+
| memory/temptable/physical_ram                                                 | 4.00
 MiB             | 515.00 MiB      |
| memory/innodb/hash0hash                                                       | 79.07
 MiB            | 79.07 MiB       |
| memory/innodb/os0event                                                        | 63.95
 MiB            | 63.95 MiB       |
| memory/performance_schema/events_statements_summary_by_digest                | 40.28
 MiB            | 40.28 MiB       |
| memory/mysys/KEY_CACHE                                                        | 16.00
 MiB            | 16.00 MiB       |
| memory/performance_schema/events_statements_history_long                      | 14.34
 MiB            | 14.34 MiB       |
| memory/performance_schema/events_errors_summary_by_thread_by_error            | 13.07
 MiB            | 13.07 MiB       |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name | 11.81
 MiB            | 11.81 MiB       |
```

```
| memory/performance_schema/events_statements_summary_by_digest.digest_text   | 9.77
 MiB            | 9.77 MiB        |
| memory/performance_schema/events_statements_history_long.sql_text           | 9.77
 MiB            | 9.77 MiB        |
+------------------------------------------------------------------------------
+--------------------+----------------+
10 rows in set (0.00 sec)
```

En Ejemplo 1: uso elevado y continuo de memoria, comprobamos el uso de memoria actual de cada conexión para determinar qué conexión es responsable del uso de la memoria en cuestión. En este ejemplo, la memoria ya está liberada, por lo que comprobar el uso de memoria de las conexiones actuales no resulta útil.

Para profundizar y encontrar las instrucciones, los usuarios y los host infractores, utilizamos Performance Schema. Performance Schema contiene varias tablas de resumen de instrucciones divididas en diferentes dimensiones, como el nombre del evento, el resumen de instrucciones, el host, el subproceso y el usuario. Cada vista le permitirá profundizar en dónde se ejecutan determinadas instrucciones y qué es lo que hacen. Esta sección se centra en `MAX_TOTAL_MEMORY`, pero puede encontrar más información sobre todas las columnas disponibles en la documentación de las tablas de resumen de instrucciones de Performance Schema.

```
mysql> SHOW TABLES IN performance_schema LIKE 'events_statements_summary_%';

+-----------------------------------------------------------+
| Tables_in_performance_schema (events_statements_summary_%) |
+-----------------------------------------------------------+
| events_statements_summary_by_account_by_event_name         |
| events_statements_summary_by_digest                        |
| events_statements_summary_by_host_by_event_name            |
| events_statements_summary_by_program                       |
| events_statements_summary_by_thread_by_event_name          |
| events_statements_summary_by_user_by_event_name            |
| events_statements_summary_global_by_event_name             |
+-----------------------------------------------------------+
7 rows in set (0.00 sec)
```

En primer lugar, comprobamos `events_statements_summary_by_digest` para ver `MAX_TOTAL_MEMORY`.

A partir de esto, podemos ver lo siguiente:

- La consulta con el resumen `20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a` parece ser una buena candidata para este uso de memoria. El valor de `MAX_TOTAL_MEMORY` es 537450710, que coincide con el nivel máximo que vimos para el evento de `memory/temptable/physical_ram` en `sys.x$memory_global_by_current_bytes`.
- Se ha ejecutado cuatro veces (`COUNT_STAR`), la primera a las 04:08:34,943256 del 26 de marzo de 2024 y, la última, a las 04:43:06,998310 del 26 de marzo de 2024.

```
mysql> SELECT SCHEMA_NAME,DIGEST,COUNT_STAR,MAX_TOTAL_MEMORY,FIRST_SEEN,LAST_SEEN
FROM performance_schema.events_statements_summary_by_digest ORDER BY MAX_TOTAL_MEMORY
 DESC LIMIT 5;

+-------------+----------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| SCHEMA_NAME | DIGEST                                                           |
 COUNT_STAR | MAX_TOTAL_MEMORY | FIRST_SEEN                | LAST_SEEN
  |
+-------------+----------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| sysbench    | 20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a |
     4 |        537450710 | 2024-03-26 04:08:34.943256 | 2024-03-26 04:43:06.998310 |
| NULL        | f158282ea0313fefd0a4778f6e9b92fc7d1e839af59ebd8c5eea35e12732c45d |
     4 |          3636413 | 2024-03-26 04:29:32.712348 | 2024-03-26 04:36:26.269329 |
| NULL        | 0046bc5f642c586b8a9afd6ce1ab70612dc5b1fd2408fa8677f370c1b0ca3213 |
     2 |          3459965 | 2024-03-26 04:31:37.674008 | 2024-03-26 04:32:09.410718 |
| NULL        | 8924f01bba3c55324701716c7b50071a60b9ceaf17108c71fd064c20c4ab14db |
     1 |          3290981 | 2024-03-26 04:31:49.751506 | 2024-03-26 04:31:49.751506 |
| NULL        | 90142bbcb50a744fcec03a1aa336b2169761597ea06d85c7f6ab03b5a4e1d841 |
     1 |          3131729 | 2024-03-26 04:15:09.719557 | 2024-03-26 04:15:09.719557 |
+-------------+----------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
5 rows in set (0.00 sec)
```

Ahora que conocemos el resumen infractor, podemos obtener más detalles, como el texto de la consulta, el usuario que la ejecutó y dónde se ejecutó. Según el texto del resumen devuelto, podemos ver que se trata de una expresión común de tabla (CTE) que crea cuatro tablas temporales y realiza cuatro análisis de tablas, lo que resulta muy ineficiente.

```
mysql> SELECT
 SCHEMA_NAME,DIGEST_TEXT,QUERY_SAMPLE_TEXT,MAX_TOTAL_MEMORY,SUM_ROWS_SENT,SUM_ROWS_EXAMINED,SUM
FROM performance_schema.events_statements_summary_by_digest
WHERE DIGEST='20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a'\G;

*************************** 1. row ***************************
           SCHEMA_NAME: sysbench
           DIGEST_TEXT: WITH RECURSIVE `cte` ( `n` ) AS ( SELECT ? FROM `sbtest1` UNION
 ALL SELECT `id` + ? FROM `sbtest1` ) SELECT * FROM `cte`
     QUERY_SAMPLE_TEXT: WITH RECURSIVE cte (n) AS (   SELECT 1  from sbtest1 UNION ALL
  SELECT id + 1 FROM sbtest1) SELECT * FROM cte
      MAX_TOTAL_MEMORY: 537450710
         SUM_ROWS_SENT: 80000000
     SUM_ROWS_EXAMINED: 80000000
SUM_CREATED_TMP_TABLES: 4
     SUM_NO_INDEX_USED: 4
1 row in set (0.01 sec)
```

Para obtener más información sobre la tabla `events_statements_summary_by_digest` y otras tablas de resumen de instrucciones de Performance Schema, consulte Statement summary tables en la documentación de MySQL.

También puede ejecutar una instrucción EXPLAIN o EXPLAIN ANALYZE para obtener más detalles.

> **Note**
>
> `EXPLAIN ANALYZE` puede aportar más información que `EXPLAIN`, pero también ejecuta la consulta, así que debe tener cuidado.

```
-- EXPLAIN
mysql> EXPLAIN WITH RECURSIVE cte (n) AS (SELECT 1  FROM sbtest1 UNION ALL SELECT id +
 1 FROM sbtest1) SELECT * FROM cte;

+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
| id | select_type | table      | partitions | type  | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra       |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
```

```
|  1 | PRIMARY     | <derived2> | NULL       | ALL   | NULL          | NULL | NULL    |
 NULL | 19221520 |   100.00 | NULL        |
|  2 | DERIVED     | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
|  3 | UNION       | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

-- EXPLAIN format=tree
mysql> EXPLAIN format=tree WITH RECURSIVE cte (n) AS (SELECT 1 FROM sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
1 row in set (0.00 sec)

-- EXPLAIN ANALYZE
mysql> EXPLAIN ANALYZE WITH RECURSIVE cte (n) AS (SELECT 1 from sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6) (actual
 time=6666..9201 rows=20e+6 loops=1)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6) (actual
 time=6666..6666 rows=20e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0365..2006 rows=10e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0311..2494 rows=10e+6 loops=1)
1 row in set (10.53 sec)
```

Pero, ¿quién lo ha ejecutado? Podemos ver en Performance Schema que el usuario `destructive_operator` tenía un `MAX_TOTAL_MEMORY` de 537450710, lo que de nuevo coincide con los resultados anteriores.

> **Note**
>
> Performance Schema se almacena en la memoria, por lo que no se debe confiar en él como la única fuente de auditoría. Si necesita mantener un historial de las instrucciones ejecutadas y qué usuarios las ejecutan, le recomendamos que habilite Aurora Advanced Auditing. Si también necesita mantener información sobre el uso de la memoria, le recomendamos que configure la supervisión para exportar y almacenar estos valores.

```
mysql> SELECT USER,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_user_by_event_name
ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+----------------------+---------------------------+------------+------------------+
| USER                 | EVENT_NAME                | COUNT_STAR | MAX_TOTAL_MEMORY |
+----------------------+---------------------------+------------+------------------+
| destructive_operator | statement/sql/select      |          4 |        537450710 |
| rdsadmin             | statement/sql/select      |       4172 |          3290981 |
| rdsadmin             | statement/sql/show_tables |          2 |          3615821 |
| rdsadmin             | statement/sql/show_fields |          2 |          3459965 |
| rdsadmin             | statement/sql/show_status |         75 |          1914976 |
+----------------------+---------------------------+------------+------------------+
5 rows in set (0.00 sec)

mysql> SELECT HOST,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_host_by_event_name
WHERE HOST != 'localhost' AND COUNT_STAR>0 ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+------------+----------------------+------------+------------------+
| HOST       | EVENT_NAME           | COUNT_STAR | MAX_TOTAL_MEMORY |
+------------+----------------------+------------+------------------+
| 10.0.8.231 | statement/sql/select |          4 |        537450710 |
+------------+----------------------+------------+------------------+
1 row in set (0.00 sec)
```

Ejemplo 3: la memoria liberable cae continuamente y no se recupera

El motor de base de datos InnoDB emplea una variedad de eventos de seguimiento de memoria especializados para diferentes componentes. Estos eventos específicos permiten un seguimiento detallado del uso de la memoria en los principales subsistemas de InnoDB, por ejemplo:

- `memory/innodb/buf0buf`: dedicado a supervisar las asignaciones de memoria para el grupo de búferes de InnoDB.
- `memory/innodb/ibuf0ibuf`: realiza un seguimiento específico de los cambios de memoria relacionados con el búfer de cambios de InnoDB.

Para identificar los principales consumidores de memoria, podemos consultar `sys.memory_global_by_current_bytes`:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.28 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

Los resultados muestran que `memory/innodb/memory` es el principal consumidor, ya que utiliza 5,28 GiB de la memoria asignada actualmente. Este evento sirve como categoría para las asignaciones de memoria entre varios componentes de InnoDB no asociados a eventos de espera más específicos, como los `memory/innodb/buf0buf` mencionados anteriormente.

Una vez establecido que los componentes de InnoDB son los principales consumidores de memoria, podemos profundizar en los detalles con el siguiente comando de MySQL:

```
SHOW ENGINE INNODB STATUS \G;
```

El comando SHOW ENGINE INNODB STATUS proporciona un informe de estado completo del motor de almacenamiento de InnoDB, que incluye estadísticas detalladas de uso de memoria de los diferentes componentes de InnoDB. Puede ayudar a identificar qué estructuras u operaciones

específicas de InnoDB consumen más memoria. Para obtener información, consulte InnoDB in-memory structures en la documentación de MySQL.

Al analizar la sección `BUFFER POOL AND MEMORY` del informe de estado de InnoDB, vemos que se asignan 5 051 647 748 bytes (4,7 GiB) a la caché de objetos del diccionario, que representa el 89 % de la memoria rastreada por `memory/innodb/memory`.

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5051647748
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

La caché de objetos del diccionario es una caché global compartida que almacena en la memoria los objetos del diccionario de datos a los que se ha accedido anteriormente para permitir la reutilización de los objetos y mejorar el rendimiento. La elevada asignación de memoria a la caché de objetos del diccionario sugiere que hay una gran cantidad de objetos de base de datos en la caché del diccionario de datos.

Ahora que sabemos que la caché del diccionario de datos es un consumidor principal, procedemos a inspeccionar la caché del diccionario de datos para ver si hay tablas abiertas. Para encontrar el número de tablas en la caché de definiciones de tablas, consulte la variable de estado global open_table_definitions.

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 20000 |
+------------------------+-------+
1 row in set (0.00 sec)
```

Para obtener más información, consulte How MySQL Opens and Closes Tables en la documentación de MySQL.

Puede limitar el número de definiciones de tablas en la caché del diccionario de datos limitando el parámetro `table_definition_cache` en el clúster de base de datos o en el grupo de parámetros de la instancia de base de datos. Para Aurora MySQL, este valor sirve como límite flexible para el número de tablas en la caché de definición de tablas. El valor predeterminado depende de la clase de instancia y se establece de la siguiente manera:

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Cuando el número de tablas supera el límite de `table_definition_cache`, el mecanismo utilizado menos recientemente (LRU) expulsa y elimina las tablas de la memoria caché. Sin embargo, las tablas involucradas en relaciones de clave externa no se incluyen en la lista LRU, lo que impide su eliminación.

En nuestro escenario actual, ejecutamos FLUSH TABLES para borrar la caché de definiciones de tablas. Esta acción provoca una caída significativa de la variable de estado global Open_table_definitions, de 20 000 a 12, tal como se muestra a continuación:

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 12    |
+------------------------+-------+
1 row in set (0.00 sec)
```

A pesar de esta reducción, observamos que la asignación de memoria para `memory/innodb/memory` sigue siendo alta, de 5,18 GiB, y la memoria del diccionario asignada también permanece sin cambios. Esto queda patente en los siguientes resultados de la consulta:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.18 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
```

```
| memory/innodb/fil0fil                                              | 198.49 MiB    |
| memory/sql/binlog_io_cache                                         | 128.00 MiB    |
| memory/innodb/mem0mem                                              | 96.82 MiB     |
| memory/innodb/dict0dict                                            | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                         | 88.00 MiB     |
+--------------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5001599639
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

Este elevado uso de memoria de forma persistente se puede atribuir a las tablas implicadas en relaciones de claves externas. Estas tablas no se incluyen en la lista LRU para su eliminación, lo que explica por qué la asignación de memoria sigue siendo alta incluso después de vaciar la caché de definiciones de tablas.

Para solucionar este problema:

1. Revise y optimice el esquema de su base de datos, especialmente las relaciones de claves externas.
2. Considere la posibilidad de cambiar a una clase de instancia de base de datos más grande que tenga más memoria para acomodar los objetos de diccionario.

Si sigue estos pasos y comprende los patrones de asignación de memoria, podrá administrar mejor el uso de la memoria en su instancia de base de datos de Aurora MySQL y evitar posibles problemas de rendimiento debido a la presión sobre la memoria.

Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL

El parámetro de nivel de instancia `aurora_oom_response` de Aurora MySQL puede permitir que la instancia de base de datos monitoricela memoria del sistema y calcule la memoria consumida por

diferentes declaraciones y conexiones. Si el sistema funciona con poca memoria, puede realizar una lista de acciones para intentar liberar dicha memoria. Lo hace en un intento de evitar un reinicio de la base de datos debido a problemas de falta de memoria (OOM). El parámetro de nivel de instancia toma una cadena de acciones separadas por comas que una instancia de base de datos realiza cuando el nivel de memoria es bajo. El parámetro `aurora_oom_response` se admite en las versiones 2 y 3 de Aurora MySQL.

Se pueden usar los siguientes valores y combinaciones de ellos para el parámetro `aurora_oom_response`. La existencia de una cadena vacía significa que no se ha tomado ninguna acción y desactiva de forma efectiva la característica, lo que hace que la base de datos sea propensa a que se reinicie debido a OOM.

- `decline`: rechaza nuevas consultas una vez que la instancia de base de datos tiene poca memoria.
- `kill_connect`: cierra las conexiones de bases de datos que consumen una gran cantidad de memoria y finaliza las transacciones actuales y las instrucciones del lenguaje de definición de datos (DDL). Esta respuesta no se admite en la versión 2 de Aurora MySQL.

  Para obtener más información, consulte KILL statement en la documentación de MySQL.
- `kill_query`: finaliza las consultas en orden descendente de consumo de memoria hasta que la memoria de la instancia esté por encima del umbral bajo. Las instrucciones DDL no finalizan.

  Para obtener más información, consulte KILL statement en la documentación de MySQL.
- `print`: solo imprime las consultas que consumen una gran cantidad de memoria.
- `tune`: ajusta las cachés de tablas internas para liberar memoria en el sistema. Aurora MySQL reduce la memoria utilizada para cachés, por ejemplo, `table_open_cache` y `table_definition_cache`, en condiciones de poca memoria. Finalmente, Aurora MySQL restablece su uso de memoria a la normalidad cuando el sistema deja de tener poca memoria.

  Para obtener más información, consulte table_open_cache y table_definition_cache en la documentación de MySQL.
- `tune_buffer_pool`: reduce el tamaño del grupo de búferes para liberar parte de la memoria y ponerla a disposición del servidor de base de datos para procesar las conexiones. Este respuestas se admite para la versión 3.06 y versiones posteriores de Aurora MySQL.

Debe emparejar `tune_buffer_pool` con `kill_query` o `kill_connect` en el valor del parámetro `aurora_oom_response`. De lo contrario, no se redimensionará el grupo de búferes, ni siquiera si incluye `tune_buffer_pool` en el valor del parámetro.

En las versiones de Aurora MySQL anteriores a la 3.06, para las clases de instancias de base de datos con una memoria inferior o igual a 4 GiB, cuando a la instancia le falta memoria, las acciones predeterminadas incluyen `print`, `tune`, `decline` y `kill_query`. Para las clases de instancia de base de datos con memoria superior a 4 GiB, el valor del parámetro está vacío de manera predeterminada (deshabilitado).

En la versión 3.06 y posteriores de Aurora MySQL, para las clases de instancias de base de datos con una memoria inferior o igual a 4 GiB, Aurora MySQL también cierra las conexiones que consumen más memoria (`kill_connect`). Para las clases de instancia de base de datos con memoria superior a 4 GiB, el valor del parámetro es `print` de forma predeterminada.

En la versión 3.09 de Aurora MySQL y posteriores, para las clases de instancia de base de datos con memoria superior a 4 GiB, el valor del parámetro es `print,decline,kill_connect` de forma predeterminada.

Si se encuentra con frecuencia con problemas de falta de memoria, puede monitorear el uso de la memoria mediante tablas de resumen de memoria cuando `performance_schema` está habilitado.

Para ver las métricas de Amazon CloudWatch relacionadas con la OOM, consulte Métricas de nivel de instancia para Amazon Aurora. Para ver las variables de estado globales relacionadas con la OOM, consulte Variables de estado globales de Aurora MySQL.

Registro de bases de datos Aurora MySQL

Los registros de Aurora MySQL proporcionan información esencial sobre la actividad y los errores de la base de datos. Al habilitar estos registros, puede identificar y solucionar problemas, comprender el rendimiento de la base de datos y auditar la actividad de la base de datos. Le recomendamos que habilite estos registros para todas las instancias de base de datos de Aurora MySQL para garantizar un rendimiento y una disponibilidad óptimos de las bases de datos. Se pueden habilitar los siguientes tipos de registros. Cada registro contiene información específica que puede permitir descubrir el impacto en el procesamiento de las bases de datos.

- Error: Aurora MySQL solo escribe en el registro de errores durante el inicio, el cierre y cuando encuentra errores. Una instancia de base de datos puede pasar horas o días sin que se escriban nuevas entradas en el registro de errores. Si no hay entradas recientes, se debe a que el servidor no ha encontrado ningún error que genere una entrada en el registro. El registro de errores está habilitado de forma predeterminada. Para obtener más información, consulte Registros de errores de Aurora MySQL.
- General: el registro general proporciona información detallada sobre la actividad de la base de datos, incluidas todas las instrucciones SQL que ejecuta el motor de base de datos. Para obtener más información sobre cómo habilitar el registro general y configurar los parámetros de registro, consulte Registros generales y de consultas lentas de Aurora MySQL y The general query log en la documentación de MySQL.

  > **Note**
  >
  > Los registros generales pueden llegar a ser muy grandes y consumir espacio de almacenamiento. Para obtener más información, consulte Rotación y retención de registros en Aurora MySQL.

- Consulta lenta: el registro de consultas lentas consta de instrucciones SQL que tardan en ejecutarse un tiempo en segundos superior al que está configurado en long_query_time y que requieren al menos el número de filas que está configurado en min_examined_row_limit para examinarlas. Puede utilizar el registro de consultas lentas para buscar consultas que tarden mucho en ejecutarse y que, por lo tanto, sean candidatas para la optimización.

  El valor de predeterminado para `long_query_time` es de 10 segundos. Le recomendamos que comience con un valor alto para identificar las consultas más lentas y, a continuación, vaya bajando para ajustarlas con mayor precisión.

También puede utilizar parámetros relacionados, como `log_slow_admin_statements` y `log_queries_not_using_indexes`. Compare `rows_examined` con `rows_returned`. Si `rows_examined` es mucho mayor que `rows_returned`, esas consultas podrían estar creando un bloqueo.

En la versión 3 de Aurora MySQL, puede habilitar `log_slow_extra` para obtener más detalles. Para obtener más información, consulte Slow query log contents en la documentación de MySQL. También puede modificar `long_query_time` para la sesión para depurar la ejecución de consultas de forma interactiva, lo que resulta especialmente útil si `log_slow_extra` está habilitado de forma global.

Para obtener más información sobre cómo habilitar el registro de consultas lentas y configurar los parámetros de registro, consulte Registros generales y de consultas lentas de Aurora MySQL y The slow query log en la documentación de MySQL.

- Auditoría: el registro de auditoría monitorea y registra la actividad de la base de datos. El registro de auditoría para Aurora MySQL se llama auditoría avanzada. Para habilitar la auditoría avanzada, hay que establecer ciertos parámetros del clúster de base de datos. Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.
- Binario: el registro binario (binlog) contiene eventos que describen los cambios en la base de datos, como las operaciones de creación de tablas y los cambios en los datos de las tablas. También contiene eventos para instrucciones que podrían haber realizado cambios (por ejemplo, un DELETE que no se corresponde con ninguna fila), a menos que se utilice el registro basado en filas. El registro binario también contiene información sobre el tiempo que tardó cada instrucción en actualizar los datos.

  Al ejecutar un servidor con el registro binario habilitado, el rendimiento es un poco más lento. Sin embargo, las ventajas del registro binario, al permitirle configurar la replicación y las operaciones de restauración, por lo general, compensan esta pequeña disminución del rendimiento.

  > **Note**
  >
  > Aurora MySQL no requiere el registro binario para las operaciones de restauración.

  Para obtener más información sobre cómo habilitar el registro binario y configurar el formato binlog, consulte Configuración del registro binario de Aurora MySQL para bases de datos de Single-AZ y The binary log en la documentación de MySQL.

Puede publicar registros de errores, generales, de consultas lentas y de auditoría en Registros de Amazon CloudWatch. Para obtener más información, consulte Publicación de registros de base de datos en registros de Amazon Cloudwatch.

Otra herramienta útil para resumir archivos de registro lentos, generales y binarios es pt-query-digest.

Solución de problemas de conexión para bases de datos Aurora MySQL

Garantizar una conectividad fiable entre sus aplicaciones y su instancia de base de datos de RDS es fundamental para el buen funcionamiento de sus cargas de trabajo. Sin embargo, los problemas de conectividad pueden surgir como consecuencia de varios factores, como las configuraciones de la red, los problemas de autenticación o las limitaciones de recursos. El objetivo de esta guía es proporcionar un enfoque integral para solucionar problemas de conectividad con Aurora MySQL.

Contenido

- Identificación de problemas de conectividad de bases de datos para Aurora MySQL
- Recopilación de datos sobre problemas de conectividad para Aurora MySQL
- Supervisión de las conexiones de bases de datos para Aurora MySQL
  - Supervisión adicional para Aurora MySQL
- Códigos de error de conectividad para Aurora MySQL
- Recomendaciones de ajuste de parámetros para Aurora MySQL
- Ejemplos de solución de problemas de conexión de bases de datos para Aurora MySQL
  - Ejemplo 1: Solución de problemas de intentos de conexión fallidos
  - Ejemplo 2: Solución de problemas de desconexiones anómalas de un cliente
  - Ejemplo 3: Solución de problemas de intentos de conexión erróneos de IAM

Identificación de problemas de conectividad de bases de datos para Aurora MySQL

Identificar la categoría específica del problema de conectividad puede ayudar a reducir las posibles causas y guiar el proceso de solución de problemas. Cada categoría puede requerir enfoques y técnicas diferentes para el diagnóstico y la resolución. Los problemas de conectividad de las bases de datos se pueden clasificar, en términos generales, en las siguientes categorías.

Excepciones y errores de conexión

> Los errores de conexión y las excepciones se pueden producir por varios motivos, como cadenas de conexión incorrectas, errores de autenticación, interrupciones de la red o problemas con el servidor de la base de datos. Algunas de las causas son parámetros de conexión mal configurados, credenciales no válidas, interrupciones de la red o bloqueos o reinicios del servidor de base de datos. Los grupos de seguridad mal configurados, la configuración de la nube privada virtual (VPC), las listas de control de acceso (ACL) a la red y las tablas de enrutamiento asociadas a las subredes también pueden provocar problemas de conexión.

Límites de conexión alcanzado

Este problema se produce cuando el número de conexiones simultáneas al servidor de base de datos supera el límite máximo permitido. Los servidores de bases de datos suelen tener un límite de conexión máximo configurable definido por el parámetro max_connections en los clústeres y los grupos de parámetros de instancia. Al imponer un límite de conexión, el servidor de base de datos se asegura de que dispone de recursos suficientes (por ejemplo, memoria, CPU y controladores de archivos) para administrar las conexiones existentes de forma eficaz y ofrecer un rendimiento aceptable. Las causas pueden incluir pérdidas de conexión en la aplicación, una agrupación de conexiones ineficiente o un aumento inesperado de las solicitudes de conexión.

Tiempos de espera de conexión

Los tiempos de espera de conexión se producen cuando la aplicación cliente no puede establecer una conexión con el servidor de base de datos dentro de un periodo de tiempo de espera especificado. Algunas de las causas más comunes son problemas de red, sobrecarga del servidor, reglas de firewall y ajustes de conexión mal configurados.

Tiempos de espera de inactividad de conexión

El servidor de la base de datos puede cerrar automáticamente las conexiones inactivas que permanecen inactivas durante un periodo prolongado para conservar los recursos. Este tiempo de espera suele configurarse mediante la tecla `wait_timeout` y `interactive_timeout parameters`, y debe ajustarse en función de los patrones de uso de la conexión de la aplicación. Algunas causas son que la lógica de la aplicación deje las conexiones inactivas durante periodos prolongados o una administración inadecuada de las conexiones.

Desconexión intermitente de las conexiones existentes

Esta clase de errores se refiere a un escenario en el que las conexiones establecidas entre una aplicación cliente y la base de datos se interrumpen inesperadamente o se desconectan a intervalos irregulares, a pesar de estar activas y en uso. Estas desconexiones se producen de forma intermitente, lo que significa que se producen a intervalos irregulares y de forma no consistente. Estas pueden ser algunas de las causas:

- Problemas con el servidor de bases de datos, como reinicios o conmutaciones por error
- Administración inadecuada de la conexión de la aplicación
- Problemas de equilibrio de carga y proxy
- Inestabilidad de la red
- Problemas con componentes o middleware de terceros involucrados en la ruta de conexión
- Tiempos de espera de ejecución de la consulta

- Restricciones de recursos en el servidor o el cliente

Es fundamental identificar la causa raíz mediante una supervisión, un registro y un análisis exhaustivos, mientras que implementar mecanismos adecuados de administración de errores, agrupación de conexiones y reintentos puede ayudar a mitigar el impacto de estas desconexiones intermitentes en la funcionalidad de la aplicación y la experiencia del usuario.

Recopilación de datos sobre problemas de conectividad para Aurora MySQL

La recopilación de datos completos relacionados con los componentes de la aplicación, la base de datos, la red y la infraestructura es crucial para solucionar eficazmente los problemas de conectividad entre una aplicación y una base de datos de Aurora MySQL. Al recopilar los registros, las configuraciones y la información de diagnóstico pertinentes, obtendrá información valiosa que puede ayudar a identificar la causa raíz de los problemas de conectividad y guiarlo hacia una solución adecuada.

Los registros y las configuraciones de la red, como las reglas de los grupos de seguridad, la configuración de la VPC y las tablas de enrutamiento, son esenciales para identificar posibles cuellos de botella o errores de configuración relacionados con la red que podrían impedir que la aplicación establezca una conexión correcta con la base de datos. Al analizar estos componentes de la red, puede asegurarse de que los puertos necesarios estén abiertos, que se permitan las direcciones IP y que las configuraciones de enrutamiento estén configuradas correctamente.

Marcas temporales

Registre las marcas horarias exactas en las que se producen los problemas de conectividad. Esto puede ayudar a identificar patrones o correlacionar los problemas con otros eventos o actividades.

Registro del motor de base de datos

Además de los registros generales de la base de datos, revise los registros del motor de base de datos (por ejemplo, el registro de errores de MySQL y el registro de consultas lentas) para ver si hay información relevante o errores que puedan estar relacionados con los problemas de conectividad intermitente. Para obtener más información, consulte Registro de bases de datos Aurora MySQL.

Registros de aplicaciones cliente

Recopile registros detallados de las aplicaciones cliente que se conectan a la base de datos. Los registros de las aplicaciones permiten ver los intentos de conexión, los errores y cualquier

información pertinente desde la perspectiva de la aplicación, lo que puede revelar problemas relacionados con las cadenas de conexión, las credenciales de autenticación o la administración de las conexiones de la aplicación.

Los registros de las bases de datos, por otro lado, ofrecen información sobre los errores de la base de datos, las consultas lentas o los eventos que podrían estar contribuyendo a los problemas de conectividad. Para obtener más información, consulte Registro de bases de datos Aurora MySQL.

Variables de entorno del cliente

Compruebe si alguna variable de entorno o configuración del cliente podría estar afectando a la conexión de la base de datos, como la configuración del proxy, la configuración de SSL/TLS o cualquier otra variable pertinente.

Versiones de la biblioteca del cliente

Asegúrese de que el cliente utilice las versiones más recientes de los controladores, bibliotecas o marcos de bases de datos utilizados para la conectividad de las bases de datos. Las versiones desactualizadas pueden tener problemas conocidos o de compatibilidad.

Captura de la red de clientes

Realice una captura de red en el cliente mediante una herramienta como Wireshark o `tcpdump` en los momentos en que se produzcan problemas de conectividad. Esto puede ayudar a identificar cualquier problema o anomalía relacionados con la red del cliente.

Topología de la red del cliente

Comprenda la topología de la red del cliente, incluidos los firewalls, los equilibradores de carga u otros componentes, como el proxy RDS o el proxy SQL, que se conectan a la base de datos en lugar de que el cliente realice las conexiones directamente.

Configuración del sistema operativo del cliente

Compruebe la configuración del sistema operativo del cliente que pueda afectar a la conectividad de la red, como las reglas del firewall, la configuración del adaptador de red y cualquier otra configuración pertinente.

Configuración de grupo de conexiones

Si utiliza un mecanismo de agrupación de conexiones en su aplicación, revise los ajustes de configuración y supervise las métricas del grupo (por ejemplo, las conexiones activas, las conexiones inactivas y los tiempos de espera de conexión) para asegurarse de que el grupo

funcione correctamente. Revise también la configuración del grupo, como el tamaño máximo, el tamaño mínimo y la configuración de validación de la conexión, para asegurarse de que están configurados correctamente.

Cadena de conexión

La cadena de conexión suele incluir parámetros como el nombre de host o punto de conexión, el número de puerto, el nombre de la base de datos y las credenciales de autenticación. El análisis de la cadena de conexión puede ayudar a identificar posibles errores de configuración o ajustes incorrectos que puedan estar causando problemas de conectividad. Por ejemplo, un nombre de host o un número de puerto incorrectos pueden impedir que el cliente acceda a la instancia de la base de datos, mientras que las credenciales de autenticación no válidas pueden provocar errores de autenticación y el rechazo de la conexión. Además, la cadena de conexión puede revelar problemas relacionados con la agrupación de conexiones, los tiempos de espera u otros ajustes específicos de la conexión que podrían contribuir a los problemas de conectividad. Proporcionar la cadena de conexión completa utilizada por la aplicación cliente puede ayudar a identificar cualquier error de configuración en el cliente.

Métricas de la base de datos

Supervise las métricas de bases de datos, como el uso de la CPU, el uso de la memoria y de la E/S del disco, durante los momentos en que se producen problemas de conectividad. Estas pueden ayudar a identificar si la instancia de base de datos tiene problemas de rendimiento o de contención de recursos.

Versión del motor de base de datos

Tenga en cuenta la versión del motor de base de datos de Aurora MySQL. AWS publica periódicamente actualizaciones que abordan los problemas conocidos y las vulnerabilidades de seguridad e introducen mejoras de rendimiento. Por lo tanto, le recomendamos encarecidamente que actualice a las últimas versiones disponibles, ya que estas actualizaciones suelen incluir correcciones de errores y mejoras relacionadas específicamente con la conectividad, el rendimiento y la estabilidad. Proporcionar la información sobre la versión de la base de datos, junto con el resto de los detalles recopilados, puede ayudar a Soporte diagnosticar y resolver los problemas de conectividad de forma eficaz.

Métricas de red

Recopile métricas de la red, como la latencia, la pérdida de paquetes y el rendimiento, durante los momentos en que se producen problemas de conectividad. Herramientas como `ping`, `traceroute` y las herramientas de supervisión de red pueden ayudar a recopilar estos datos.

Datos de origen y cliente

Determine las direcciones IP de los servidores de aplicaciones, los equilibradores de carga o cualquier otro componente que inicie las conexiones de la base de datos. Puede ser una única dirección IP o un intervalo de direcciones IP (notación de CIDR). Si el origen es una instancia de Amazon EC2, también es útil revisar el tipo de instancia, la zona de disponibilidad, el ID de subred y los grupos de seguridad asociados a la instancia, así como los detalles de la interfaz de red, como la dirección IP privada y la dirección IP pública.

Al analizar minuciosamente los datos recopilados, puede identificar los errores de configuración, las limitaciones de recursos, las interrupciones de la red u otros problemas subyacentes que están causando los problemas de conectividad intermitentes o persistentes. Esta información le permite tomar medidas específicas, como ajustar las configuraciones, resolver problemas de red o abordar la administración de la conexión de aplicación.

Supervisión de las conexiones de bases de datos para Aurora MySQL

Para supervisar y solucionar problemas de conectividad, puede utilizar las siguientes métricas y características.

Métricas de CloudWatch

- `CPUUtilization`: un uso elevado de la CPU en la instancia de base de datos puede provocar una ejecución lenta de las consultas, lo que puede provocar tiempos de espera o rechazos de la conexión.
- `DatabaseConnections`: supervise el número de conexiones activas a la instancia de base de datos. Un número elevado de conexiones cercano al máximo configurado puede indicar posibles problemas de conectividad o el agotamiento del grupo de conexiones.
- `FreeableMemory`: tener poca memoria disponible puede provocar problemas de rendimiento y conectividad como consecuencia de la escasez de recursos.
- `NetworkReceiveThroughput` y `NetworkTransmitThroughput`: los picos o caídas inusuales en el rendimiento de la red pueden indicar problemas de conectividad o cuellos de botella en la red.

Métricas de Información de rendimiento

Para solucionar problemas de conectividad en Aurora MySQL mediante Información de rendimiento, analice las métricas de la base de datos como las siguientes:

- Aborted_clients
- Aborted_connects
- Connections
- max_connections
- Threads_connected
- Threads_created
- Threads_running

Estas métricas pueden ayudarlo a identificar los cuellos de botella en la conexión, detectar problemas de red o de autenticación, optimizar la agrupación de conexiones y garantizar una administración eficiente de los subprocesos. Para obtener más información, consulte Contadores de Información de rendimiento para Aurora MySQL.

Características de Información de rendimiento

- Carga de la base de datos: visualice la carga de la base de datos a lo largo del tiempo y correlaciónela con los problemas de conectividad o la degradación del rendimiento.
- Estadísticas de SQL: analice las estadísticas de SQL para identificar consultas u operaciones de base de datos ineficientes que puedan contribuir a los problemas de conectividad.
- Consultas principales: identifique y analice las consultas que consumen más recursos, lo que puede ayudar a identificar posibles cuellos de botella en el rendimiento o consultas de larga duración que pueden estar causando problemas de conectividad.

Al supervisar estas métricas y aprovechar Información de rendimiento, puede obtener visibilidad del rendimiento de la instancia de base de datos, el uso de recursos y los posibles cuellos de botella que podrían estar causando problemas de conectividad. Por ejemplo:

- Si `DatabaseConnections` se acerca mucho al límite máximo, puede indicar que se ha agotado el conjunto de conexiones o se ha administrado la conexión de forma inadecuada, lo que puede provocar problemas de conectividad.
- Un nivel alto de `CPUUtilization` o bajo de `FreeableMemory` puede indicar limitaciones de recursos, lo que puede provocar una ejecución lenta de las consultas y tiempos de espera o rechazos de la conexión.
- El análisis de las consultas principales y estadísticas de SQL puede ayudar a identificar las consultas ineficientes o que consumen muchos recursos y que pueden estar contribuyendo a los problemas de conectividad.

Además, la supervisión de registros de CloudWatch y la configuración de alarmas pueden ayudarlo a identificar y responder de forma proactiva a los problemas de conectividad antes de que se agraven.

Es importante tener en cuenta que, si bien estas métricas y herramientas pueden proporcionar información valiosa, deben utilizarse junto con otros pasos de solución de problemas. Al revisar también las configuraciones de red, las reglas de los grupos de seguridad y el manejo de la conexión de la aplicación, puede diagnosticar y resolver exhaustivamente los problemas de conectividad con las instancias de base de datos de Aurora MySQL.

Supervisión adicional para Aurora MySQL

Métricas de CloudWatch

- `AbortedClients`: realiza un seguimiento el número de conexiones de cliente que no se han cerrado correctamente.
- `AuroraSlowConnectionHandleCount`: realiza un seguimiento del número de operaciones de administración de conexiones lentas, lo que indica posibles problemas de conectividad o cuellos de botella en el rendimiento.
- `AuroraSlowHandshakeCount`: mide el número de operaciones de protocolos de enlace lentos, lo que también puede ser un indicador de problemas de conectividad.
- `ConnectionAttempts`: mide el número de intentos de conexión realizados a la instancia de base de datos de Aurora MySQL.

Variables de estado globales

`Aurora_external_connection_count`: muestra el número de conexiones de base de datos a la instancia de base de datos, excluidas las conexiones al servicio RDS utilizadas para las comprobaciones de estado de la base de datos.

Al supervisar estas métricas y variables de estado global, puede obtener visibilidad de los patrones de conexión, los errores y los posibles cuellos de botella que pueden estar causando problemas de conectividad con su instancia de Amazon Aurora MySQL.

Por ejemplo, un número elevado de problemas de `AbortedClients` o `AuroraSlowConnectionHandleCount` puede indicar problemas de conectividad.

Además, configurar las alarmas y notificaciones de CloudWatch puede ayudarlo a identificar y responder de forma proactiva a los problemas de conectividad antes de que se agraven y afecten al rendimiento de la aplicación.

Códigos de error de conectividad para Aurora MySQL

A continuación, se muestran algunos errores de conectividad comunes de las bases de datos Aurora MySQL, junto con sus códigos de error y sus explicaciones.

Código de error 1040: Too many connections

Este error se produce cuando el cliente intenta establecer más conexiones que el máximo permitido por el servidor de la base de datos. Entre las causas posibles, se incluyen:

- Configuración incorrecta de la agrupación de conexiones: si utiliza un mecanismo de agrupación de conexiones, asegúrese de que el tamaño máximo de la agrupación no sea demasiado alto y de que las conexiones se devuelvan correctamente a la agrupación.
- Configuración de la instancia de base de datos: compruebe la configuración máxima de conexiones permitidas para la instancia de base de datos y ajústela si es necesario configurando el parámetro `max_connections`.
- Alta concurrencia: si varios clientes o aplicaciones se conectan a la base de datos simultáneamente, es posible que se alcance el límite máximo de conexiones permitido.

Código de error 1045: Access denied for user '...'@'...' (using password: YES/NO)

Este error indica un error de autenticación al intentar conectarse a la base de datos. Entre las causas posibles, se incluyen:

- Compatibilidad con el complemento de autenticación: compruebe si el complemento de autenticación utilizado por el cliente es compatible con el mecanismo de autenticación del servidor de la base de datos.
- Nombre de usuario o contraseña incorrectos: compruebe que se estén utilizando el nombre de usuario y la contraseña correctos en la cadena de conexión o en el mecanismo de autenticación.
- Permisos de usuario: asegúrese de que el usuario tenga los permisos necesarios para conectarse a la instancia de base de datos desde el host o la red especificados.

Código de error 1049: Unknown database '...'

Este error indica que el cliente está intentando conectarse a una base de datos que no existe en el servidor. Entre las causas posibles, se incluyen:

- Base de datos no creada: asegúrese de que la base de datos especificada se haya creado en el servidor de bases de datos.

- Nombre de base de datos incorrecto: compruebe si el nombre de la base de datos utilizado en la cadena de conexión o consulta es correcto.
- Permisos de usuario: compruebe que el usuario tenga los permisos necesarios para acceder a la base de datos especificada.

Código de error 1153: Got a packet bigger than 'max_allowed_packet' bytes

Este error se produce cuando el cliente intenta enviar o recibir datos que superan el tamaño máximo de paquete permitido por el servidor de bases de datos. Entre las causas posibles, se incluyen:

- Consultas o conjuntos de resultados de gran tamaño: si se ejecutan consultas que implican grandes cantidades de datos, es posible que se exceda el límite de tamaño del paquete.
- Configuración de tamaño de paquete mal configurada: compruebe la configuración de `max_allowed_packet` en el servidor de la base de datos y ajústela si es necesario.
- Problemas de configuración de la red: asegúrese de que la configuración de la red (por ejemplo, el tamaño de la MTU) permita los tamaños de paquete requeridos.

Código de error 1226: User '...' has exceeded the 'max_user_connections' resource (current value: ...)

Este error indica que el usuario ha superado el número máximo de conexiones simultáneas permitido por el servidor de base de datos. Entre las causas posibles, se incluyen:

- Configuración incorrecta de la agrupación de conexiones: si utiliza un mecanismo de agrupación de conexiones, asegúrese de que el tamaño máximo de la agrupación no sea demasiado alto para el límite de conexiones del usuario.
- Configuración de la instancia de base de datos: compruebe la configuración de `max_user_connections` para la instancia de base de datos y ajústela si es necesario.
- Alta concurrencia: si varios clientes o aplicaciones se conectan a la base de datos simultáneamente con el mismo usuario, es posible que se alcance el límite máximo de conexiones específicas del usuario.

Código de error 2003: Can't connect to MySQL server on '...' (10061)

Este error suele producirse cuando el cliente no puede establecer una conexión TCP/IP con el servidor de la base de datos. Puede deberse a varios problemas, como los siguientes:

- Estado de la instancia de base de datos: asegúrese de que la instancia de base de datos esté en el estado `available` y no esté siendo sometida a ninguna operación de mantenimiento ni copia de seguridad.

- Reglas de firewall: compruebe si algún firewall (sistema operativo, red o grupo de seguridad) está bloqueando la conexión en el puerto especificado (normalmente el 3306 para MySQL).
- Nombre de host o punto de conexión incorrectos: asegúrese de que el nombre de host o punto de conexión utilizado en la cadena de conexión sea correcto y coincida con la instancia de base de datos.
- Problemas de conectividad de red: compruebe que el equipo cliente pueda acceder a la instancia de base de datos a través de la red. Compruebe si hay interrupciones en la red, problemas de enrutamiento o errores de configuración de la VPC o de la subred.

Código de error 2005: Unknown MySQL server host '...' (11001)

Este error se produce cuando el cliente no puede resolver el nombre de host o el punto de conexión del servidor de la base de datos en una dirección IP. Entre las causas posibles, se incluyen:

- Problemas de resolución de DNS: compruebe que el equipo cliente pueda resolver el nombre de host correctamente mediante DNS. Compruebe la configuración de DNS y la caché de DNS e intente utilizar la dirección IP en lugar del nombre de host.
- Nombre de host o punto de conexión incorrectos: compruebe si el nombre de host o el punto de conexión utilizados en la cadena de conexión son correctos.
- Problemas de configuración de red: asegúrese de que la configuración de red del cliente (por ejemplo, VPC, subred y tablas de enrutamiento) permita la resolución de DNS y la conectividad a la instancia de la base de datos.

Código de error 2026: SSL connection error

Este error se produce cuando hay un problema con la configuración de SSL/TLS o la validación del certificado durante el intento de conexión. Entre las causas posibles, se incluyen:

- Caducidad del certificado: compruebe si el certificado SSL/TLS utilizado por el servidor ha caducado y debe renovarse.
- Problemas con la validación del certificado: compruebe que el cliente puede validar correctamente el certificado SSL/TLS del servidor y que el certificado es de confianza.
- Problemas de configuración de red: asegúrese de que la configuración de red permita las conexiones SSL/TLS y no bloquee ni interfiera con el proceso de protocolo de enlace SSL/TLS.
- La configuración de SSL/TLS no coincide: asegúrese de que la configuración de SSL/TLS (por ejemplo, los conjuntos de cifrado y las versiones de los protocolos) del cliente y el servidor sea compatible.

Si comprende las explicaciones detalladas y las posibles causas de cada código de error, podrá solucionar mejor los problemas de conectividad al trabajar con bases de datos Aurora MySQL.

Recomendaciones de ajuste de parámetros para Aurora MySQL

Número máximo de conexiones

El ajuste de estos parámetros puede ayudar a evitar problemas de conexión causados por alcanzar el límite máximo de conexiones permitido. Asegúrese de que estos valores estén configurados adecuadamente en función de los requisitos de concurrencia y las limitaciones de recursos de la aplicación.

- `max_connections`: este parámetro especifica el número máximo de conexiones simultáneas que se permite a la instancia de base de datos.
- `max_user_connections`: este parámetro se puede especificar durante la creación y modificación del usuario, y establece el número máximo de conexiones simultáneas permitidas para una cuenta de usuario específica.

Tamaño del búfer de red

El aumento de estos valores puede mejorar el rendimiento de la red, especialmente en el caso de cargas de trabajo que implican grandes transferencias de datos o conjuntos de resultados. Sin embargo, tenga cuidado, ya que los tamaños de búfer más grandes pueden consumir más memoria.

- `net_buffer_length`: este parámetro establece el tamaño inicial de los búferes de conexión al cliente y de resultados, equilibrando el uso de memoria con el rendimiento de las consultas.
- `max_allowed_packet`: este parámetro especifica el tamaño máximo de un único paquete de red que la instancia de base de datos puede enviar o recibir.

Compresión de red (en el cliente)

Habilitar la compresión de la red puede reducir el uso del ancho de banda de la red, pero puede aumentar la sobrecarga de la CPU tanto en el cliente como en el servidor.

- `compress`: este parámetro habilita o deshabilita la compresión de red para la comunicación entre el cliente y el servidor.
- `compress_protocol`: este parámetro especifica el protocolo de compresión que se utilizará para la comunicación de red.

Ajuste del rendimiento de la red

Ajustar estos tiempos de espera puede ayudar a administrar las conexiones inactivas y evitar el agotamiento de los recursos, pero tenga cuidado, ya que los valores bajos pueden provocar la finalización prematura de las conexiones.

- `interactive_timeout`: este parámetro especifica el número de segundos que el servidor espera a que se produzca actividad en una conexión interactiva antes de cerrarla.
- `wait_timeout`: este parámetro determina el número de segundos que el servidor espera a que se produzca actividad en una conexión no interactiva antes de cerrarla.

Configuración de tiempo de espera de la red

Ajustar estos tiempos de espera puede ayudar a solucionar los problemas relacionados con las conexiones lentas o que no responden. Sin embargo, tenga cuidado de no usar valores demasiado bajos, ya que puede provocar fallos de conexión prematuros.

- `net_read_timeout`: este parámetro especifica el número de segundos que se deben esperar para recibir más datos de una conexión antes de finalizar la operación de lectura.
- `net_write_timeout`: este parámetro determina el número de segundos que se debe esperar a que se escriba un bloque en una conexión antes de finalizar la operación de escritura.

Ejemplos de solución de problemas de conexión de bases de datos para Aurora MySQL

Los siguientes ejemplos muestran cómo identificar y solucionar los problemas de conexión a bases de datos de Aurora MySQL.

Ejemplo 1: Solución de problemas de intentos de conexión fallidos

Los intentos de conexión pueden fallar por varios motivos, incluidos los errores de autenticación, los errores del protocolo de enlace SSL/TLS, el límite `max_connections` alcanzado y las restricciones de recursos de la instancia de base de datos.

Puede realizar un seguimiento del número de conexiones fallidas desde Información de rendimiento, o mediante el siguiente comando.

```
mysql> show global status like 'aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
```

```
| Aborted_connects | 7     |
+------------------+-------+
1 row in set (0.00 sec)
```

Si el número de `Aborted_connects` aumenta con el tiempo, es posible que la aplicación tenga problemas de conectividad intermitentes.

Puede usar Aurora Advanced Auditing para registrar las conexiones y desconexiones de las conexiones del cliente. Para ello, defina los siguientes parámetros en el grupo de parámetros de clúster de base de datos:

- `server_audit_logging` = `1`
- `server_audit_events` = `CONNECT`

El siguiente es un extracto de los registros de auditoría de un inicio de sesión fallido.

```
1728498527380921,auora-mysql-node1,user_1,172.31.49.222,147189,0,FAILED_CONNECT,,,1045
1728498527380940,auora-mysql-node1,user_1,172.31.49.222,147189,0,DISCONNECT,,,0
```

Donde:

- `1728498527380921`: la marca temporal de la época en la que se produjo el error de inicio de sesión
- `aurora-mysql-node1`: el identificador de instancia del nodo del clúster de Aurora MySQL en el que se produjo un error de conexión
- `user_1`: el nombre del usuario de base de datos para el que no se pudo iniciar sesión
- `172.31.49.222`: la dirección IP privada del cliente desde el que se estableció la conexión
- `147189`: el ID de conexión del inicio de sesión fallido
- `FAILED_CONNECT`: indica que la conexión ha fallado
- `1045`: el código de devolución Un valor distinto de cero indica un error. En este caso, `1045` corresponde al acceso denegado.

Para obtener más información, consulte Server error codes y Client error codes en la documentación de MySQL.

También puede examinar los registros de errores de Aurora MySQL para ver si hay algún mensaje de error relacionado, por ejemplo:

```
2024-10-09T19:26:59.310443Z 220 [Note] [MY-010926] [Server] Access denied for user
 'user_1'@'172.31.49.222' (using password: YES) (sql_authentication.cc:1502)
```

Ejemplo 2: Solución de problemas de desconexiones anómalas de un cliente

Puede realizar un seguimiento del número de desconexiones anómalas de un cliente desde Información de rendimiento o mediante el siguiente comando.

```
mysql> show global status like 'aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 9     |
+-----------------+-------+
1 row in set (0.01 sec)
```

Si el número de `Aborted_clients` aumenta con el tiempo, significa que la aplicación no cierra correctamente las conexiones a la base de datos. Si las conexiones no se cierran correctamente, se pueden producir pérdidas de recursos y posibles problemas de rendimiento. Dejar las conexiones abiertas innecesariamente puede consumir recursos del sistema, como la memoria y los descriptores de archivos, lo que puede provocar que la aplicación o el servidor dejen de responder o se reinicien.

Puede utilizar la siguiente consulta para identificar las cuentas que no cierran conexiones correctamente. Recupera el nombre de la cuenta de usuario, el host desde el que se conecta el usuario, el número de conexiones no cerradas y el porcentaje de conexiones no cerradas.

```
SELECT
    ess.user,
    ess.host,
    (a.total_connections - a.current_connections) - ess.count_star AS not_closed,
    (((a.total_connections - a.current_connections) - ess.count_star) * 100) /
 (a.total_connections - a.current_connections) AS pct_not_closed
FROM
    performance_schema.events_statements_summary_by_account_by_event_name AS ess
    JOIN performance_schema.accounts AS a ON (ess.user = a.user AND ess.host = a.host)
WHERE
    ess.event_name = 'statement/com/quit'
    AND (a.total_connections - a.current_connections) > ess.count_star;

+----------+--------------+------------+----------------+
```

```
| user     | host          | not_closed | pct_not_closed |
+----------+---------------+------------+----------------+
| user1    | 172.31.49.222 |          1 |        33.3333 |
| user1    | 172.31.93.250 |       1024 |        12.1021 |
| user2    | 172.31.93.250 |         10 |        12.8551 |
+----------+---------------+------------+----------------+
3 rows in set (0.00 sec)
```

Tras identificar las cuentas de usuario y los hosts desde los que no se cierran las conexiones, puede comprobar el código que no cierra correctamente las conexiones.

Por ejemplo, con el conector MySQL en Python, utilice el método `close()` del objeto de conexión para cerrar las conexiones. A continuación, se muestra un ejemplo de función que establece una conexión a una base de datos, realiza una consulta y cierra la conexión:

```
import mysql.connector

def execute_query(query):
    # Establish a connection to the database
    connection = mysql.connector.connect(
        host="your_host",
        user="your_username",
        password="your_password",
        database="your_database"
    )

    try:
        # Create a cursor object
        cursor = connection.cursor()

        # Execute the query
        cursor.execute(query)

        # Fetch and process the results
        results = cursor.fetchall()
        for row in results:
            print(row)

    finally:
        # Close the cursor and connection
        cursor.close()
        connection.close()
```

En este ejemplo, se llama al método `connection.close()` en el bloque `finally` para asegurarse de que la conexión está cerrada, se produzca o no una excepción.

Ejemplo 3: Solución de problemas de intentos de conexión erróneos de IAM

La conectividad con usuarios de AWS Identity and Access Management (IAM) puede errores por varios motivos, como:

- Configuración de política de IAM incorrecta
- Credenciales de seguridad caducadas
- Problemas de conectividad de red
- No coinciden los permisos de la base de datos

Para solucionar estos errores de autenticación, active la característica de exportación de registros de `iam-db-auth-error` en el servicio de la base de datos relacional (RDS) de Amazon o base de datos de Aurora. Esto le permitirá ver los mensajes de error de autenticación detallados en el grupo de registros de CloudWatch para el clúster de Amazon RDS o Amazon Aurora.

Una vez habilitados, puede revisar estos registros para identificar y resolver la causa específica de los errores de autenticación de IAM.

Por ejemplo:

```
2025-09-22T12:02:30,806 [ERROR] Failed to authorize the connection request for
 user 'user_1' due to an internal IAM DB Auth error. (Status Code: 500, Error Code:
 InternalError)
```

y

```
2025-09-22T12:02:51,954 [ERROR] Failed to authenticate the connection request for user
 'user_2' because the provided token is malformed or otherwise invalid. (Status Code:
 400, Error Code: InvalidToken)
```

Para obtener orientación sobre la solución de problemas, consulte la guía de solución de problemas de Aurora para la autenticación de bases de datos de IAM.

Solución de problemas de rendimiento de consultas de las bases de datos Aurora MySQL

MySQL proporciona control del optimizador de consultas a través de variables del sistema que afectan a la forma en que se evalúan los planes de consultas, las optimizaciones intercambiables, las sugerencias de optimizadores e índices y el modelo de costos del optimizador. Estos puntos de datos pueden ser útiles no solo al comparar diferentes entornos MySQL, sino también para comparar los planes de ejecución de consultas anteriores con los planes de ejecución actuales y para comprender la ejecución general de una consulta MySQL en cualquier momento.

El rendimiento de las consultas depende de muchos factores, como el plan de ejecución, el esquema y tamaño de las tablas, las estadísticas, los recursos, los índices y la configuración de los parámetros. El ajuste de las consultas requiere identificar cuellos de botella y optimizar la ruta de ejecución.

- Busque el plan de ejecución de la consulta y compruebe si la consulta utiliza los índices adecuados. Para optimizar la consulta, puede utilizar `EXPLAIN` y revisar los detalles de cada plan.
- Aurora MySQL versión 3 (compatible con MySQL 8.0 Community Edition) utiliza una instrucción `EXPLAIN ANALYZE`. La instrucción `EXPLAIN ANALYZE` es una herramienta de elaboración de perfiles que indica en dónde MySQL dedica tiempo a su consulta y por qué. Con `EXPLAIN ANALYZE`, Aurora MySQL planifica, prepara y ejecuta la consulta mientras cuenta las filas y mide el tiempo empleado en varios puntos del plan de ejecución. Cuando se completa la consulta, `EXPLAIN ANALYZE` imprime el plan y sus medidas en lugar del resultado de la consulta.
- Mantenga las estadísticas del esquema actualizadas mediante la instrucción `ANALYZE`. A veces, el optimizador de consultas puede elegir planes de ejecución inadecuados debido a que las estadísticas están desactualizadas. Esto puede provocar un rendimiento deficiente de una consulta debido a que las estimaciones de cardinalidad de las tablas y los índices no son exactas. La columna `last_update` de la tabla innodb_table_stats muestra la última vez que se actualizaron las estadísticas del esquema, lo que es un buen indicador de que están “estancadas”.
- Pueden producirse otros problemas, como un sesgo en la distribución de los datos, que no se tienen en cuenta para determinar la cardinalidad de la tabla. Para obtener más información, consulte Estimating ANALYZE TABLE complexity for InnoDB tables y Histogram statistics in MySQL en la documentación de MySQL.

Descripción del tiempo que emplean las consultas

Estas son formas de determinar el tiempo que emplean las consultas:

- Elaboración de perfiles
- Performance Schema
- Optimizador de consultas

Elaboración de perfiles

La elaboración de perfiles está deshabilitada de forma predeterminada. Habilite la elaboración de perfiles y, a continuación, ejecute la consulta lenta y revise su perfil.

```
SET profiling = 1;
Run your query.
SHOW PROFILE;
```

1. Identifique la etapa en la que pasa más tiempo. De acuerdo con General thread states, en la documentación de MySQL, leer y procesar filas de una instrucción `SELECT` determinada suele ser el estado que tarda más tiempo en ejecutarse a lo largo de la vida útil de una consulta. Puede usar la instrucción `EXPLAIN` para saber cómo MySQL ejecuta esta consulta.
2. Revise el registro de consultas lentas para evaluar `rows_examined` y `rows_sent` para asegurarse de que la carga de trabajo es similar en cada entorno. Para obtener más información, consulte Registro de bases de datos Aurora MySQL.
3. Ejecute el siguiente comando para las tablas que forman parte de la consulta identificada:

   ```
   SHOW TABLE STATUS\G;
   ```

4. Obtenga la siguiente salida antes y después de ejecutar la consulta en cada entorno:

   ```
   SHOW GLOBAL STATUS;
   ```

5. Ejecute los siguientes comandos en cada entorno para comprobar si hay alguna otra consulta o sesión que influya en el rendimiento de esta consulta de ejemplo.

   ```
   SHOW FULL PROCESSLIST;

   SHOW ENGINE INNODB STATUS\G;
   ```

A veces, cuando los recursos del servidor están ocupados, esto afecta a todas las demás operaciones del servidor, incluidas las consultas. También puede obtener información periódicamente cuando se ejecutan consultas o configurar un trabajo de `cron` para obtener información a intervalos útiles.

Performance Schema

Performance Schema proporciona información útil sobre el rendimiento del tiempo de ejecución del servidor y, al mismo tiempo, tiene un impacto mínimo en ese rendimiento. Es diferente de `information_schema`, que proporciona información de esquema sobre la instancia de base de datos. Para obtener más información, consulte Descripción general de Performance Schema para Información de rendimiento en Aurora MySQL.

Seguimiento del optimizador de consultas

Para entender por qué se eligió un plan de consultas en particular para su ejecución, puede configurar `optimizer_trace` para acceder al optimizador de consultas de MySQL.

Ejecute un seguimiento del optimizador para ver información exhaustiva sobre todas las rutas disponibles para el optimizador y su elección.

```
SET SESSION OPTIMIZER_TRACE="enabled=on";
SET optimizer_trace_offset=-5, optimizer_trace_limit=5;

-- Run your query.
SELECT * FROM table WHERE x = 1 AND y = 'A';

-- After the query completes:
SELECT * FROM information_schema.OPTIMIZER_TRACE;
SET SESSION OPTIMIZER_TRACE="enabled=off";
```

Revisión de la configuración del optimizador de consultas

Aurora MySQL versión 3 (compatible con MySQL 8.0 Community Edition) tiene muchos cambios relacionados con el optimizador en comparación con Aurora MySQL versión 2 (compatible con MySQL 5.7 Community Edition). Si tiene algunos valores personalizados para `optimizer_switch`, le recomendamos que revise las diferencias entre los valores predeterminados y establezca los valores de `optimizer_switch` que mejor se adapten a su carga de trabajo. También le recomendamos que pruebe las opciones disponibles para la versión 3 de Aurora MySQL para examinar el rendimiento de las consultas.

> **Note**
>
> La versión 3 de Aurora MySQL utiliza el valor predeterminado de 20 de la comunidad para el parámetro innodb_stats_persistent_sample_pages.

Puede utilizar el siguiente comando para mostrar los valores de `optimizer_switch`:

```
SELECT @@optimizer_switch\G;
```

En la tabla siguiente, se muestran los valores predeterminados de `optimizer_switch` en Aurora MySQL versiones 2 y 3.

Opción	Aurora MySQL versión 2	Aurora MySQL versión 3
batched_key_access	off	off
block_nested_loop	on	on
condition_fanout_filter	on	on
derived_condition_pushdown	–	on
derived_merge	on	on
duplicateweedout	on	on
engine_condition_pushdown	on	on
firstmatch	on	on
hash_join	off	on
hash_join_cost_based	on	–
hypergraph_optimizer	–	off
index_condition_pushdown	on	on
index_merge	on	on

Opción	Aurora MySQL versión 2	Aurora MySQL versión 3
index_merge_intersection	on	on
index_merge_sort_union	on	on
index_merge_union	on	on
loosescan	on	on
materialization	on	on
mrr	on	on
mrr_cost_based	on	on
prefer_ordering_index	on	on
semijoin	on	on
skip_scan	–	on
subquery_materialization_cost_based	on	on
subquery_to_derived	–	off
use_index_extensions	on	on
use_invisible_indexes	–	off

Para obtener más información, consulte Switchable optimizations (MySQL 5.7) y Switchable optimizations (MySQL 8.0) en la documentación de MySQL.

Referencia de Amazon Aurora MySQL

Esta referencia incluye información sobre parámetros de Aurora MySQL, variables de estado y extensiones SQL generales o diferencias con el motor de base de datos MySQL de la comunidad.

Temas

- Parámetros de configuración de Aurora MySQL
- Variables de estado globales de Aurora MySQL
- Eventos de espera de Aurora MySQL
- Estados del subproceso Aurora MySQL
- Niveles de aislamiento de Aurora MySQL
- Sugerencias de Aurora MySQL
- Referencia de procedimientos almacenados en Aurora MySQL
- Aurora MySQL: tablas de information_schema específica

Parámetros de configuración de Aurora MySQL

El clúster de bases de datos Amazon Aurora MySQL se administra de la misma forma que otras instancias de base de datos de Amazon RDS: con los parámetros de un grupo de parámetros de base de datos. Amazon Aurora difiere de otros motores de base de datos en los que hay un clúster de bases de datos que contiene varias instancias de base de datos. Como resultado, algunos de los parámetros que utiliza para administrar su clúster de bases de datos de Aurora MySQL se aplican a todo el clúster. Los demás parámetros se aplican solo a una instancia de base de datos determinada en el clúster de bases de datos.

Utilice grupos de parámetros de clúster de bases de datos para administrar los parámetros de nivel de clúster. Utilice grupos de parámetros de base de datos para administrar los parámetros de nivel de instancia. Todas las instancias de base de datos en un clúster de bases de datos de Aurora MySQL son compatibles con el motor de base de datos de MySQL. Sin embargo, aplique algunos de los parámetros del motor de base de datos MySQL en el nivel de clúster y administre estos parámetros mediante los grupos de parámetros de clúster de bases de datos. No puede detectar los parámetros de nivel del clúster en el grupo de parámetros de base de datos para una instancia en un clúster de bases de datos de Aurora. Después aparecerá una lista de parámetros de nivel de clúster en este tema.

Puede administrar tanto los parámetros de nivel de clúster como los de nivel de instancia con la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS. Utilice comandos independientes para administrar los parámetros de nivel de clúster y los parámetros de nivel de instancia. Por ejemplo, puede utilizar el comando de la CLI modify-db-clúster-parameter-group para administrar parámetros de nivel de clúster en un grupo de parámetros del clúster de bases de datos. Puede utilizar el comando de la CLI modify-db-parameter-group para administrar parámetros de nivel de instancia en un grupo de parámetros de base de datos para una instancia de base de datos en un clúster de bases de datos.

Puede ver tanto los parámetros de nivel de clúster como los de nivel de instancia en la consola o usando la CLI o la API de RDS. Por ejemplo, puede utilizar el comando de la AWS CLI describe-db-clúster-parameters para ver parámetros de nivel de clúster en un grupo de parámetros del clúster de bases de datos. Puede utilizar el comando de la CLI describe-db-parameters para ver parámetros de nivel de instancia en un grupo de parámetros de base de datos para una instancia de base de datos en un clúster de bases de datos.

> **Note**
>
> Cada grupo de parámetros predeterminados contiene los valores predeterminados para todos los parámetros del grupo de parámetros. Si el parámetro tiene “engine default” para este valor, consulte la documentación de MySQL o PostgreSQL específica de la versión para obtener el valor predeterminado real.
> A menos que se indique lo contrario, los parámetros que figuran en las siguientes tablas son válidos para las versiones 2 y 3 de Aurora MySQL.

Para obtener más información acerca de los grupos de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora. Para conocer las reglas y restricciones para clústeres Aurora Serverless v1, consulte Grupos de parámetros de Aurora Serverless v1.

Temas

- Parámetros de nivel de clúster
- Parámetros de nivel de instancia
- Parámetros de MySQL que no se aplican a Aurora MySQL

Parámetros de nivel de clúster

En la siguiente tabla se muestran todos los parámetros que afectan a todo el clúster de bases de datos Aurora MySQL.

Nombre del parámetro	Modificable	Notas
`aurora_binlog_read_buffer_size`	Sí	Solo afecta a los clústeres que utilizan replicación de registro binario (binlog). Para obtener información acerca de la replicación de binlog, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario). Eliminado de Aurora MySQL versión 3.
`aurora_binlog_replication_max_yield_seconds`	Sí	Solo afecta a los clústeres que utilizan replicación de registro binario (binlog). Para obtener información acerca de la replicación de binlog, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).
`aurora_binlog_replication_sec_index_parallel_workers`	Sí	Establece el número total de subprocesos paralelos disponibles para aplicar cambios de índice secundarios al replicar transacciones para tablas grandes con más de un índice secundario. El parámetro está establecido en `0` (deshabilitado) de forma predeterminada. Este parámetro está disponible en la versión 3.06 y versiones posteriores de Aurora MySQL. Para obtener más información, consulte Optimización de

Nombre del parámetro	Modificable	Notas
		la replicación de registros binarios para Aurora MySQL.
aurora_binlog_use_large_read_buffer	Sí	Solo afecta a los clústeres que utilizan replicación de registro binario (binlog). Para obtener información acerca de la replicación de binlog, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario). Eliminado de Aurora MySQL versión 3.
aurora_disable_hash_join	Sí	Establezca este valor en ON para desactivar la optimización de las combinaciones hash en Aurora MySQL versión 2.09 o posteriores. No se admite en la versión 3. Para obtener más información, consulte Consulta paralela para Amazon Aurora MySQL.
aurora_enable_replica_log_compression	Sí	Para obtener más información, consulte Consideraciones sobre el rendimiento de la replicación de Amazon Aurora MySQL. No aplica cambios a clústeres que formen parte de una base de datos global de Aurora. Eliminado de Aurora MySQL versión 3.
aurora_enable_repl_bin_log_filtering	Sí	Para obtener más información, consulte Consideraciones sobre el rendimiento de la replicación de Amazon Aurora MySQL. No aplica cambios a clústeres que formen parte de una base de datos global de Aurora. Eliminado de Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`aurora_enable_staggered_replica_restart`	Sí	Esta configuración está disponible en la versión 3 de Aurora MySQL, pero no se usa.
`aurora_enable_zdr`	Sí	Esta configuración se activa de forma predeterminada en Aurora MySQL 2.10 y posteriores. Para obtener más información, consulte Reinicio sin tiempo de inactividad (ZDR) para Amazon Aurora MySQL.
`aurora_enhanced_binlog`	Sí	Establezca el valor de este parámetro en 1 para activar el binlog mejorado en Aurora MySQL versión 3.03.1 y posteriores. Para obtener más información, consulte Configuración del binlog mejorado para Aurora MySQL.
`aurora_jemalloc_background_thread`	Sí	Utilice este parámetro para permitir que un subproceso en segundo plano realice operaciones de mantenimiento de memoria. Los valores permitidos son `0` (deshabilitado) y 1 (habilitado). El valor predeterminado es `0`. Este parámetro se aplica a la versión 3.04 y versiones posteriores de Aurora MySQL.

Nombre del parámetro	Modificable	Notas
`aurora_jemalloc_dirty_decay_ms`	Sí	Utilice este parámetro para retener la memoria liberada durante un tiempo determinado (en milisegundos). La retención de la memoria permite una reutilización más rápida. Los valores permitidos son: El valor predeterminado (`0`) devuelve toda la memoria al sistema operativo como memoria liberable. Este parámetro se aplica a la versión 3.04 y versiones posteriores de Aurora MySQL.
`aurora_jemalloc_tcache_enabled`	Sí	Utilice este parámetro para atender solicitudes de memoria pequeñas (hasta 32 KiB) en una caché local de subprocesos, evitando los ámbitos de memoria. Los valores permitidos son `0` (deshabilitado) y 1 (habilitado). El valor predeterminado es 1. Este parámetro se aplica a la versión 3.04 y versiones posteriores de Aurora MySQL.
`aurora_load_from_s3_role`	Sí	Para obtener más información, consulte Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3. Actualmente no está disponible en Aurora MySQL versión 3. Utilice `aws_default_s3_role` .

Nombre del parámetro	Modificable	Notas
aurora_mask_password_hashes_type	Sí	Esta configuración se activa de forma predeterminada en Aurora MySQL 2.11 y posteriores. Utilice esta configuración para ocultar los hash de contraseñas de Aurora MySQL en las consultas lentas y los registros de de auditoría. Los valores permitidos son 0 y 1 (predeterminado). Cuando se establece en 1, las contraseñas se registran como <secret>. Cuando se establece en 0, las contraseñas se registran como valores hash (#).
aurora_select_into_s3_role	Sí	Para obtener más información, consulte Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3. Actualmente no está disponible en Aurora MySQL versión 3. Utilice aws_default_s3_role .

Nombre del parámetro	Modificable	Notas
`authentication_kerberos_caseins_cmp`	Sí	Controla la comparación de nombres de usuario del complemento de `authentication_kerberos` que no distingue mayúsculas de minúsculas. Configúrelo en `true` para que la comparación no distinga entre mayúsculas y minúsculas. De forma predeterminada, se utiliza la comparación entre mayúsculas y minúsculas (`false`). Para obtener más información, consulte Uso de la autenticación Kerberos para Aurora MySQL. Este parámetro está disponible en Aurora MySQL versión 3.03 y posterior.
`auto_increment_increment`	Sí	
`auto_increment_offset`	Sí	
`aws_default_lambda_role`	Sí	Para obtener más información, consulte Invocación de una función de Lambda desde un clúster de base de datos de Amazon Aurora MySQL.

Nombre del parámetro	Modificable	Notas
aws_default_s3_role	Sí	Se utiliza al invocar la instrucción LOAD DATA FROM S3, LOAD XML FROM S3 o SELECT INTO OUTFILE S3 desde el clúster de base de datos. En la versión 2 de Aurora MySQL, el rol de IAM especificado en este parámetro se usa cuando no se especifica un rol de IAM para aurora_load_from_s3_role o aurora_select_into_s3_role en la instrucción correspondiente. En la versión 3 de Aurora MySQL, siempre se utiliza el rol de IAM especificado para este parámetro. Para obtener más información, consulte Asociación de un rol de IAM con un clúster de base de datos Amazon Aurora MySQL.
binlog_backup	Sí	Establezca el valor de este parámetro en 0 para activar el binlog mejorado en Aurora MySQL versión 3.03.1 y posteriores. Puede desactivar este parámetro solo si usa el binlog mejorado. Para obtener más información, consulte Configuración del binlog mejorado para Aurora MySQL.

Nombre del parámetro	Modificable	Notas
binlog_checksum	Sí	La API AWS CLI y RDS informa un valor de None si este parámetro no está establecido. En ese caso, Aurora MySQL utiliza el valor predeterminado del motor, que es CRC32. Esto es diferente de la configuración explícita de NONE, que desactiva la suma de comprobación.
binlog-do-db	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_format	Sí	Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).
binlog_group_commit_sync_delay	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_group_commit_sync_no_delay_count	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog-ignore-db	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_replication_globaldb	Sí	Establezca el valor de este parámetro en 0 para activar el binlog mejorado en Aurora MySQL versión 3.03.1 y posteriores. Puede desactivar este parámetro solo si usa el binlog mejorado. Para obtener más información, consulte Configuración del binlog mejorado para Aurora MySQL.

Nombre del parámetro	Modificable	Notas
binlog_row_image	No	
binlog_row_metadata	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_row_value_options	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_rows_query_log_events	Sí	
binlog_transaction_compression	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_transaction_compression_level_zstd	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_transaction_dependency_history_size	Sí	Este parámetro establece un límite superior en el número de hashes de fila que se guardan en la memoria y se utiliza para buscar la transacción que modificó por última vez una fila determinada. Cuando se alcanza este número de hashes, se purga el historial. Este parámetro se aplica a la versión 2.12 y versiones posteriores y a la versión 3 de Aurora MySQL.
binlog_transaction_dependency_tracking	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
character-set-client-handshake	Sí	
character_set_client	Sí	

Nombre del parámetro	Modificable	Notas
character_set_connection	Sí	
character_set_database	Sí	
character_set_filesystem	Sí	
character_set_results	Sí	
character_set_server	Sí	
collation_connection	Sí	
collation_server	Sí	
completion_type	Sí	
default_storage_engine	No	Los clústeres de Aurora MySQL usan el motor de almacenamiento de InnoDB para todos sus datos.
enforce_gtid_consistency	A veces	Se puede modificar en la versión 2 de Aurora MySQL y versiones posteriores.
event_scheduler	Sí	Indica el estado del programador de eventos. Solo se puede modificar en el nivel del clúster en Aurora MySQL versión 3.
gtid-mode	A veces	Se puede modificar en la versión 2 de Aurora MySQL y versiones posteriores.

Nombre del parámetro	Modificable	Notas
`information_schema_stats_expiry`	Sí	El número de segundos después de los cuales el servidor de bases de datos MySQL recupera los datos del motor de almacenamiento y los reemplaza en la memoria caché. Los valores permitidos son: Este parámetro se aplica a Aurora MySQL versión 3.
`init_connect`	Sí	El comando que ejecutará el servidor para cada cliente que se conecte. Utilice comillas dobles (") en la configuración para evitar errores de conexión, por ejemplo: `SET optimizer_switch="hash_join=off"` En la versión 3 de Aurora MySQL, este parámetro no se aplica a los usuarios que tienen el privilegio `CONNECTION_ADMIN` . Esto incluye al usuario maestro de Aurora. Para obtener más información, consulte Modelo de privilegios basado en roles.
`innodb_adaptive_hash_index`	Sí	Puede modificar este parámetro en el nivel del clúster de base de datos en las versiones 2 y 3 de Aurora MySQL. El índice hash adaptativo no se admite en instancias de base de datos de lector.

Nombre del parámetro	Modificable	Notas
innodb_aurora_instant_alter_column_allowed	Sí	Controla si el INSTANT algoritmo se puede utilizar para ALTER COLUMN operaciones a nivel global. Los valores permitidos son los siguientes: • 0: el algoritmo INSTANT no está permitido para las operaciones ALTER COLUMN (OFF). Se revierte a otros algoritmos. • 1: el algoritmo INSTANT no está permitido para las operaciones ALTER COLUMN (ON). Este es el valor predeterminado. Para obtener más información, consulte Optimizing Locking Operations en la documentación de MySQL. Este parámetro se aplica a la versión 3.04 y versiones posteriores de Aurora MySQL.
innodb_autoinc_lock_mode	Sí	
innodb_checksums	No	Eliminado de Aurora MySQL versión 3.
innodb_cmp_per_index_enabled	Sí	
innodb_commit_concurrency	Sí	
innodb_data_home_dir	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.

Nombre del parámetro	Modificable	Notas
`innodb_deadlock_detect`	Sí	Esta opción se utiliza para deshabilitar la detección de bloqueos en la versión 2.11 y versiones posteriores y la versión 3 de Aurora MySQL. En los sistemas de alta simultane idad, la detección de bloqueos puede provocar una ralentización cuando numerosos hilos esperan el mismo bloqueo. Consulte la documentación de MySQL para obtener más información sobre este parámetro.
`innodb_default_row_format`	Sí	Este parámetro define el formato de fila predeterminado para las tablas de InnoDB (incluidas las tablas temporale s de InnoDB creadas por el usuario). Se aplica a las versiones 2 y 3 de Aurora MySQL. Su valor puede ser `DYNAMIC`, `COMPACT` o `REDUNDANT`. .
`innodb_file_per_table`	Sí	El parámetro afecta a cómo se organiza el almacenamiento de tablas. Para obtener más información, consulte Escalado del almacenamiento.

Nombre del parámetro	Modificable	Notas
innodb_flush_log_at_trx_commit	Sí	Le recomendamos encarecidamente que utilice el valor predeterminado de 1. En la versión 3 de Aurora MySQL, antes de poder configurar este parámetro con un valor distinto de 1, primero debe cambiar el valor de innodb_trx_commit_allow_data_loss a 1. Para obtener más información, consulte Configuración de la frecuencia de vaciado del búfer de registro.
innodb_ft_max_token_size	Sí	
innodb_ft_min_token_size	Sí	
innodb_ft_num_word_optimize	Sí	
innodb_ft_sort_pll_degree	Sí	
innodb_online_alter_log_max_size	Sí	
innodb_optimize_fulltext_only	Sí	
innodb_page_size	No	

Nombre del parámetro	Modificable	Notas
innodb_print_all_deadlocks	Sí	Cuando está activado, registra información sobre todos los interbloqueos de InnoDB en el registro de errores de Aurora MySQL. Para obtener más información, consulte Minimización y solución de problemas de los interbloqueos de Aurora MySQL.
innodb_purge_batch_size	Sí	
innodb_purge_threads	Sí	
innodb_rollback_on_timeout	Sí	
innodb_rollback_segments	Sí	
innodb_spin_wait_delay	Sí	
innodb_strict_mode	Sí	
innodb_support_xa	Sí	Eliminado de Aurora MySQL versión 3.
innodb_sync_array_size	Sí	
innodb_sync_spin_loops	Sí	
innodb_stats_include_delete_marked	Sí	Cuando este parámetro está habilitado, InnoDB incluye los registros marcados como borrados al calcular las estadísticas persistentes del optimizador. Este parámetro se aplica a la versión 2.12 y versiones posteriores y a la versión 3 de Aurora MySQL.
innodb_table_locks	Sí	

Nombre del parámetro	Modificable	Notas
innodb_trx_commit_allow_data_loss	Sí	En Aurora MySQL versión 3, establezca el valor de este parámetro en 1 de modo que pueda cambiar el valor de innodb_flush_log_at_trx_commit . El valor predeterminado de innodb_trx_commit_allow_data_loss es 0. Para obtener más información, consulte Configuración de la frecuencia de vaciado del búfer de registro.
innodb_undo_directory	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
internal_tmp_disk_storage_engine	Sí	Controla qué motor de almacenamiento en memoria se utiliza para las tablas temporales internas. Los valores permitidos son INNODB y MYISAM. Este parámetro se aplica a Aurora MySQL versión 2.
internal_tmp_mem_storage_engine	Sí	Controla qué motor de almacenamiento en memoria se utiliza para las tablas temporales internas. Los valores permitidos son MEMORY y TempTable . Este parámetro se aplica a Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`key_buffer_size`	Sí	Memoria caché de claves para tablas MyISAM. Para obtener más informaci ón, consulte keycache->cache_lock mutex.
`lc_time_names`	Sí	
`log_error_suppression_list`	Sí	Especifica una lista de códigos de error que no se registran en el registro de errores de MySQL. Esto le permite ignorar ciertas condiciones de error no críticas para ayudar a mantener limpios los registros de errores. Para obtener más información, consulte log_error _suppression_list en la documentación de MySQL. Este parámetro se aplica a la versión 3.03 y versiones posteriores de Aurora MySQL.
`low_priority_updates`	Sí	Las operaciones `INSERT`, `UPDATE`, `DELETE` y `LOCK TABLE WRITE` esperan hasta que no haya ninguna operación `SELECT` pendiente. Este parámetro solo afecta a los motores de almacenamiento que utilizan únicament e el bloqueo en el nivel de tabla (MyISAM, MEMORY, MERGE). Este parámetro se aplica a Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`lower_case_table_names`	Sí (para la versión 2 de Aurora MySQL) Solo en el momento de la creación del clúster (versión 3 de Aurora MySQL)	En las versiones 2.10 de Aurora MySQL y posteriores 2.x, asegúrese de reiniciar todas las instancias del lector después de cambiar esta configuración y reiniciar la instancia del escritor. Para obtener más información, consulte Reinicio de un clúster de Aurora con disponibilidad de lectura. En Aurora MySQL versión 3, el valor de este parámetro se establece de forma permanente en el momento de crear el clúster. Si utiliza un valor no predeterminado para esta opción, configure el grupo de parámetros personalizados Aurora MySQL versión 3 antes de actualizar y especifique el grupo de parámetros durante la operación de restauración de instantáneas que crea el clúster de la versión 3. Con una base de datos global de Aurora basada en Aurora MySQL, no se puede realizar una actualización local desde la versión 2 a la versión 3 de Aurora MySQL si el parámetro `lower_case_table_names` está activado. Para obtener más información sobre los métodos que puede utilizar, consulte Actualizaciones de la versión principal.
`master-info-repository`	Sí	Eliminado de Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`master_verify_checksum`	Sí	Aurora MySQL versión 2. Usar `source_verify_checksum` en Aurora MySQL versión 3.
`max_delayed_threads`	Sí	Establece el número máximo de subprocesos para gestionar las instrucciones `INSERT DELAYED`. Este parámetro se aplica a Aurora MySQL versión 3.
`max_error_count`	Sí	Número máximo de mensajes de error, advertencia y nota que se almacenará para su visualización. Este parámetro se aplica a Aurora MySQL versión 3.
`max_execution_time`	Sí	El tiempo de espera para ejecutar instrucciones `SELECT`, en milisegundos. Los valores pueden ser de `0` a `18446744073709551615`. Si se establece en `0`, no hay tiempo de espera. Para obtener más información, consulte max_execution_time en la documentación de MySQL.
`min_examined_row_limit`	Sí	Utilice este parámetro para evitar que se registren las consultas que examinan un número de filas inferior al especificado. Este parámetro se aplica a Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`partial_revokes`	No	Este parámetro se aplica a Aurora MySQL versión 3.
`preload_buffer_size`	Sí	Tamaño del búfer que se asigna al precargar los índices. Este parámetro se aplica a Aurora MySQL versión 3.
`query_cache_type`	Sí	Eliminado de Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`read_only`	Sí	Cuando este parámetro está activado, el servidor no permite actualizaciones, excepto las que realizan los subprocesos de réplica. Los valores válidos para Aurora MySQL versión 2 son los siguientes: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` - `ON` para réplicas de lectura. Este es el valor predeterminado. • `{TrueIfClusterReplica}` - `ON` para clústeres de réplicas, como réplicas de lectura entre regiones, clústeres secundarios en una base de datos global de Aurora e implementaciones azul/verde. Los valores válidos para Aurora MySQL versión 3 son los siguientes: • `0` – `OFF`. Este es el valor predeterminado. • `1` – `ON` • `{TrueIfClusterReplica}` - `ON` para clústeres de réplicas, como réplicas de lectura entre regiones, clústeres secundarios en una base de datos global de Aurora e implementaciones azul/verde.

Nombre del parámetro	Modificable	Notas
		En la versión 3 de Aurora MySQL, este parámetro no se aplica a los usuarios que tienen el privilegio `CONNECTION_ADMIN` . Esto incluye al usuario maestro de Aurora. Para obtener más información, consulte Modelo de privilegios basado en roles.
`relay-log-space-limit`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replica_parallel_type`	Sí	Este parámetro permite la ejecución en paralelo en la réplica de todos los subprocesos no confirmados que ya se encuentran en la fase de preparación, sin infringir la coherencia. Se aplica a la versión 3 de Aurora MySQL. En la versión 3.03.* y versiones anteriores de Aurora MySQL, el valor predeterminado es DATABASE. En la versión 3.04 y versiones anteriores de Aurora MySQL, el valor predeterminado es LOGICAL_CLOCK.
`replica_preserve_commit_order`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replica_transaction_retries`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
`replica_type_conversions`	Sí	Este parámetro determina las conversiones de tipo utilizadas en las réplicas. Los valores permitidos son `ALL_LOSSY` `ALL_NON_LOSSY` , `ALL_SIGNED` y `ALL_UNSIGNED` . Para obtener más información, consulte el tema de Replicación con definiciones de tablas diferentes sobre el origen y la réplica en la documentación de MySQL. Este parámetro se aplica a Aurora MySQL versión 3.
`replicate-do-db`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replicate-do-table`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replicate-ignore-db`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replicate-ignore-table`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replicate-wild-do-table`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`replicate-wild-ignore-table`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`require_secure_transport`	Sí	Este parámetro se aplica a Aurora MySQL versión 2 y 3. Para obtener más información, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

Nombre del parámetro	Modificable	Notas
`rpl_read_size`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`server_audit_cw_upload`	No	Este parámetro ha quedado obsoleto en Aurora MySQL. Utilice `server_audit_logs_upload` . Para obtener más información, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.
`server_audit_events`	Sí	Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.
`server_audit_excl_users`	Sí	Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.
`server_audit_incl_users`	Sí	Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.
`server_audit_logging`	Sí	Para conocer las instrucciones sobre la carga de los registros en registros de Amazon Cloudwatch, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.

Nombre del parámetro	Modificable	Notas
server_audit_logs_upload	Sí	Puede publicar registros de auditoría en CloudWatch Logs activando la auditoría avanzada y configurando este parámetro en 1. El valor predeterminado para el parámetro server_audit_logs_upload es 0. Para obtener más información, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.
server_id	No	
skip-character-set-client-handshake	Sí	
skip_name_resolve	No	
slave-skip-errors	Sí	Solo se aplica a clústeres de la versión 2 de Aurora MySQL, con compatibilidad MySQL 5.7.
source_verify_checksum	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
sync_frm	Sí	Eliminado de Aurora MySQL versión 3.
thread_cache_size	Sí	La cantidad de subprocesos que se van a almacenar en caché. Este parámetro se aplica a Aurora MySQL versiones 2 y 3.

Nombre del parámetro	Modificable	Notas
`time_zone`	Sí	De manera predeterminada, la zona horaria de un clúster de base de datos de Aurora es el horario universal coordinado (UTC). En su lugar, puede definir la zona horaria de las instancias del clúster de base de datos en la zona horaria local de su aplicación. Para obtener más información, consulte Zona horaria local para los clústeres de base de datos de Amazon Aurora.
`tls_version`	Sí	Para obtener más información, consulte Versiones de TLS para Aurora MySQL.

Parámetros de nivel de instancia

En la siguiente tabla se muestran todos los parámetros que afectan a una instancia de base de datos concreta de un clúster de bases de datos Aurora MySQL.

Nombre del parámetro	Modificable	Notas
`activate_all_roles_on_login`	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
`allow-suspicious-udfs`	No	
`aurora_disable_hash_join`	Sí	Establezca este valor en `ON` para desactivar la optimización de las combinaciones hash en Aurora MySQL versión 2.09 o posteriores. No se admite en la versión 3. Para obtener más información, consulte Consulta paralela para Amazon Aurora MySQL.

Nombre del parámetro	Modificable	Notas
`aurora_lab_mode`	Sí	Para obtener más información, consulte Modo lab de Amazon Aurora MySQL. Eliminado de Aurora MySQL versión 3.
`aurora_oom_response`	Sí	Este parámetro se admite en las versiones 2 y 3 de Aurora MySQL. Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.
`aurora_parallel_query`	Sí	Establezca este valor en `ON` para activar la consulta paralela en las versiones de Aurora MySQL 2.09 o posteriores. El parámetro anterior de `aurora_pq` no se utiliza en estas versiones. Para obtener más información, consulte Consulta paralela para Amazon Aurora MySQL.
`aurora_pq`	Sí	Establezca el valor en `OFF` para desactivar la consulta paralela para instancias de base de datos específicas en versiones de Aurora MySQL anteriores a 2.09. En la versión 2.09 o posteriores, active y desactive la consulta paralela con `aurora_parallel_query` en su lugar. Para obtener más información, consulte Consulta paralela para Amazon Aurora MySQL.

Nombre del parámetro	Modificable	Notas
`aurora_read_replica_read_committed`	Sí	Habilita el nivel de aislamiento `READ COMMITTED` para las réplicas de Aurora y cambia el comportamiento del aislamiento para reducir el lag de purgado durante las consultas de ejecución prolongada. Habilite esta configuración solo si comprende los cambios de comportamiento y cómo afectan a los resultados de su consulta. Por ejemplo, esta configuración utiliza un aislamiento menos estricto que el MySQL predeterminado. Cuando se habilita, las consultas de ejecución prolongada pueden ver más de una copia de la misma fila ya que Aurora reorganiza los datos de la tabla mientras se ejecuta la consulta. Para obtener más información, consulte Niveles de aislamiento de Aurora MySQL.
`aurora_tmptable_enable_per_table_limit`	Sí	Determina si el parámetro `tmp_table_size` controla el tamaño máximo de las tablas temporales en memoria creadas por el motor de almacenamiento `TempTable` en la versión 3.04 y versiones posteriores de Aurora MySQL. Para obtener más información, consulte Limitación del tamaño de las tablas temporales internas en memoria.

Nombre del parámetro	Modificable	Notas
aurora_use_vector_instructions	Sí	Cuando este parámetro está habilitado, Aurora MySQL utiliza las instrucciones de procesamiento vectorial optimizadas que proporcionan las CPU modernas para mejorar el rendimiento en las cargas de trabajo con un uso intensivo de E/S. Esta configuración se activa de forma predeterminada en Aurora MySQL 2.11 y posteriores.
autocommit	Sí	
automatic_sp_privileges	Sí	
back_log	Sí	
basedir	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
binlog_cache_size	Sí	
binlog_max_flush_queue_time	Sí	
binlog_order_commits	Sí	
binlog_stmt_cache_size	Sí	
binlog_transaction_compression	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
binlog_transaction_compression_level_zstd	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
bulk_insert_buffer_size	Sí	

Nombre del parámetro	Modificable	Notas
concurrent_insert	Sí	
connect_timeout	Sí	
core-file	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
datadir	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
default_authentication_plugin	No	Este parámetro se aplica a Aurora MySQL versión 3.
default_time_zone	No	
default_tmp_storage_engine	Sí	Motor de almacenamiento predeterminado para tablas temporales.
default_week_format	Sí	
delay_key_write	Sí	
delayed_insert_limit	Sí	
delayed_insert_timeout	Sí	
delayed_queue_size	Sí	
div_precision_increment	Sí	
end_markers_in_json	Sí	
eq_range_index_dive_limit	Sí	

Nombre del parámetro	Modificable	Notas
event_scheduler	A veces	Indica el estado del programador de eventos. Solo se puede modificar a nivel de clúster en Aurora MySQL versión 3.
explicit_defaults_for_timestamp	Sí	
flush	No	
flush_time	Sí	
ft_boolean_syntax	No	
ft_max_word_len	Sí	
ft_min_word_len	Sí	
ft_query_expansion_limit	Sí	
ft_stopword_file	Sí	
general_log	Sí	Para conocer las instrucciones sobre la carga de los registros en CloudWatch Logs, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.
general_log_file	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
group_concat_max_len	Sí	
host_cache_size	Sí	

Nombre del parámetro	Modificable	Notas
`init_connect`	Sí	El comando que ejecutará el servidor para cada cliente que se conecte. Utilice comillas dobles (") en la configuración para evitar errores de conexión, por ejemplo: `SET optimizer_switch="hash_join =off"` En la versión 3 de Aurora MySQL, este parámetro no se aplica a los usuarios que tienen el privilegio `CONNECTIO N_ADMIN` , incluido el usuario maestro de Aurora. Para obtener más informaci ón, consulte Modelo de privilegios basado en roles.
`innodb_adaptive_hash_index`	Sí	Puede modificar este parámetro en el nivel de la instancia de base de datos en la versión 2 de Aurora MySQL. Solo se puede modificar en el nivel del clúster de base de datos en Aurora MySQL versión 3. El índice hash adaptativo no se admite en instancias de base de datos de lector.
`innodb_adaptive_max_sleep_d elay`	Sí	La modificación de este parámetro no tiene ningún efecto, porque `innodb_th read_concurrency` es siempre 0 para Aurora.

Nombre del parámetro	Modificable	Notas
innodb_aurora_max_partitions_for_range	Sí	En algunos casos en los que las estadísticas persistentes no estén disponibles, puede utilizar este parámetro para mejorar el rendimiento de las estimaciones del recuento de filas en las tablas divididas. Puede configurarlo en un valor comprendido entre 0 y 8192, donde el valor determina el número de particiones que se van a comprobar durante la estimación del recuento de filas. El valor predeterminado es 0, que se estima utilizando todas las particiones, de acuerdo con el comportamiento predeterminado de MySQL. Este parámetro está disponible para la versión 3.03.1 y posteriores de Aurora MySQL.
innodb_autoextend_increment	Sí	
innodb_buffer_pool_dump_at_shutdown	No	
innodb_buffer_pool_dump_now	No	
innodb_buffer_pool_filename	No	
innodb_buffer_pool_load_abort	No	

Nombre del parámetro	Modificable	Notas
innodb_buffer_pool_load_at_startup	No	
innodb_buffer_pool_load_now	No	
innodb_buffer_pool_size	Sí	El valor predeterminado se represent a con una fórmula. Para obtener más información sobre cómo se calcula el valor de DBInstanceClassMemory de la fórmula, consulte Variables de las fórmulas de parámetros de base de datos.
innodb_change_buffer_max_size	No	Aurora MySQL no utiliza el búfer de cambio de InnoDB en absoluto.
innodb_compression_failure_threshold_pct	Sí	
innodb_compression_level	Sí	
innodb_compression_pad_pct_max	Sí	
innodb_concurrency_tickets	Sí	La modificación de este parámetro no tiene ningún efecto, porque innodb_thread_concurrency es siempre 0 para Aurora.

Nombre del parámetro	Modificable	Notas
innodb_deadlock_detect	Sí	Esta opción se utiliza para deshabilitar la detección de bloqueos en la versión 2.11 y versiones posteriores y la versión 3 de Aurora MySQL. En los sistemas de alta simultane idad, la detección de bloqueos puede provocar una ralentización cuando numerosos hilos esperan el mismo bloqueo. Consulte la documentación de MySQL para obtener más información sobre este parámetro.
innodb_file_format	Sí	Eliminado de Aurora MySQL versión 3.
innodb_flushing_avg_loops	No	
innodb_force_load_corrupted	No	
innodb_ft_aux_table	Sí	
innodb_ft_cache_size	Sí	
innodb_ft_enable_stopword	Sí	
innodb_ft_server_stopword_table	Sí	
innodb_ft_user_stopword_table	Sí	
innodb_large_prefix	Sí	Eliminado de Aurora MySQL versión 3.
innodb_lock_wait_timeout	Sí	
innodb_log_compressed_pages	No	

Nombre del parámetro	Modificable	Notas
innodb_lru_scan_depth	Sí	
innodb_max_purge_lag	Sí	
innodb_max_purge_lag_delay	Sí	
innodb_monitor_disable	Sí	
innodb_monitor_enable	Sí	
innodb_monitor_reset	Sí	
innodb_monitor_reset_all	Sí	
innodb_old_blocks_pct	Sí	
innodb_old_blocks_time	Sí	
innodb_open_files	Sí	
innodb_print_all_deadlocks	Sí	Cuando está activado, registra información sobre todos los interbloqueos de InnoDB en el registro de errores de Aurora MySQL. Para obtener más información, consulte Minimización y solución de problemas de los interbloqueos de Aurora MySQL.
innodb_random_read_ahead	Sí	
innodb_read_ahead_threshold	Sí	
innodb_read_io_threads	No	

Nombre del parámetro	Modificable	Notas
innodb_read_only	No	Aurora MySQL administra el estado de solo lectura y lectura/escritura de las instancias de base de datos según el tipo de clúster. Por ejemplo, un clúster aprovisionado dispone de una instancia de base de datos de lectura/escritura (la instancia principal) y mis otras instancia s en el clúster son de solo lectura (las réplicas de Aurora).
innodb_replication_delay	Sí	
innodb_sort_buffer_size	Sí	
innodb_stats_auto_recalc	Sí	
innodb_stats_method	Sí	
innodb_stats_on_metadata	Sí	
innodb_stats_persistent	Sí	
innodb_stats_persistent_sam ple_pages	Sí	
innodb_stats_transient_samp le_pages	Sí	
innodb_thread_concurrency	No	
innodb_thread_sleep_delay	Sí	La modificación de este parámetro no tiene ningún efecto, porque innodb_th read_concurrency es siempre 0 para Aurora.

Nombre del parámetro	Modificable	Notas
interactive_timeout	Sí	Aurora evalúa el valor mínimo de interactive_timeout y wait_timeout . Utiliza ese mínimo como tiempo de espera para finalizar todas las sesiones inactivas, tanto interactivas como no interactivas.
internal_tmp_disk_storage_engine	Sí	Controla qué motor de almacenamiento en memoria se utiliza para las tablas temporales internas. Los valores permitidos son INNODB y MYISAM. Este parámetro se aplica a Aurora MySQL versión 2.
internal_tmp_mem_storage_engine	Sí	Controla qué motor de almacenamiento en memoria se utiliza para las tablas temporales internas. Los valores permitidos son MEMORY y TempTable . Este parámetro se aplica a Aurora MySQL versión 3.
join_buffer_size	Sí	
keep_files_on_create	Sí	
key_buffer_size	Sí	Memoria caché de claves para tablas MyISAM. Para obtener más información, consulte keycache->cache_lock mutex.
key_cache_age_threshold	Sí	
key_cache_block_size	Sí	
key_cache_division_limit	Sí	

Nombre del parámetro	Modificable	Notas
`local_infile`	Sí	
`lock_wait_timeout`	Sí	
`log-bin`	No	Si `binlog_format` se establece en `STATEMENT`, `MIXED`, o `ROW`, `log-bin` se establecerá automáticamente en `ON`. Si se establece `binlog_format` en `OFF`, `log-bin` se establecerá automáticamente en `OFF`. Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).
`log_bin_trust_function_creators`	Sí	
`log_bin_use_v1_row_events`	Sí	Eliminado de Aurora MySQL versión 3.
`log_error`	No	
`log_error_suppression_list`	Sí	Especifica una lista de códigos de error que no se registran en el registro de errores de MySQL. Esto le permite ignorar ciertas condiciones de error no críticas para ayudar a mantener limpios los registros de errores. Para obtener más información, consulte log_error_suppression_list en la documentación de MySQL. Este parámetro se aplica a la versión 3.03 y versiones posteriores de Aurora MySQL.

Nombre del parámetro	Modificable	Notas
log_output	Sí	
log_queries_not_using_indexes	Sí	
log_slave_updates	No	Aurora MySQL versión 2. Usar log_replica_updates en Aurora MySQL versión 3.
log_replica_updates	No	Aurora MySQL versión 3
log_throttle_queries_not_using_indexes	Sí	
log_warnings	Sí	Eliminado de Aurora MySQL versión 3.
long_query_time	Sí	
low_priority_updates	Sí	Las operaciones INSERT, UPDATE, DELETE y LOCK TABLE WRITE esperan hasta que no haya ninguna operación SELECT pendiente. Este parámetro solo afecta a los motores de almacenamiento que utilizan únicamente el bloqueo en el nivel de tabla (MyISAM, MEMORY, MERGE). Este parámetro se aplica a Aurora MySQL versión 3.
max_allowed_packet	Sí	
max_binlog_cache_size	Sí	
max_binlog_size	No	
max_binlog_stmt_cache_size	Sí	

Nombre del parámetro	Modificable	Notas
max_connect_errors	Sí	
max_connections	Sí	El valor predeterminado se represent a con una fórmula. Para obtener más información sobre cómo se calcula el valor de DBInstanceClassMem ory de la fórmula, consulte Variables de las fórmulas de parámetros de base de datos. Para ver los valores predeterminados en función de la clase de instancia, consulte Número máximo de conexiones a una instancia de base de datos Aurora MySQL.
max_delayed_threads	Sí	Establece el número máximo de subprocesos para gestionar las instrucci ones INSERT DELAYED. Este parámetro se aplica a Aurora MySQL versión 3.
max_error_count	Sí	Número máximo de mensajes de error, advertencia y nota que se almacenará para su visualización. Este parámetro se aplica a Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
max_execution_time	Sí	El tiempo de espera para ejecutar instrucciones SELECT, en milisegundos. Los valores pueden ser de 0 a 18446744073709551615 . Si se establece en 0, no hay tiempo de espera. Para obtener más información, consulte max_execution_time en la documentación de MySQL.
max_heap_table_size	Sí	
max_insert_delayed_threads	Sí	
max_join_size	Sí	
max_length_for_sort_data	Sí	Eliminado de Aurora MySQL versión 3.
max_prepared_stmt_count	Sí	
max_seeks_for_key	Sí	
max_sort_length	Sí	
max_sp_recursion_depth	Sí	
max_tmp_tables	Sí	Eliminado de Aurora MySQL versión 3.
max_user_connections	Sí	
max_write_lock_count	Sí	
metadata_locks_cache_size	Sí	Eliminado de Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
min_examined_row_limit	Sí	Utilice este parámetro para evitar que se registren las consultas que examinan un número de filas inferior al especific ado. Este parámetro se aplica a Aurora MySQL versión 3.
myisam_data_pointer_size	Sí	
myisam_max_sort_file_size	Sí	
myisam_mmap_size	Sí	
myisam_sort_buffer_size	Sí	
myisam_stats_method	Sí	
myisam_use_mmap	Sí	
net_buffer_length	Sí	
net_read_timeout	Sí	
net_retry_count	Sí	
net_write_timeout	Sí	
old-style-user-limits	Sí	
old_passwords	Sí	Eliminado de Aurora MySQL versión 3.
optimizer_prune_level	Sí	
optimizer_search_depth	Sí	

Nombre del parámetro	Modificable	Notas
optimizer_switch	Sí	Para obtener información acerca de las características de Aurora MySQL que utilizan este modificador, consulte Prácticas recomendadas con Amazon Aurora MySQL.
optimizer_trace	Sí	
optimizer_trace_features	Sí	
optimizer_trace_limit	Sí	
optimizer_trace_max_mem_size	Sí	
optimizer_trace_offset	Sí	
performance-schema-consumer-events-waits-current	Sí	
performance-schema-instrument	Sí	
performance_schema	Sí	
performance_schema_accounts_size	Sí	
performance_schema_consumer_global_instrumentation	Sí	
performance_schema_consumer_thread_instrumentation	Sí	
performance_schema_consumer_events_stages_current	Sí	

Nombre del parámetro	Modificable	Notas
performance_schema_consumer_events_stages_history	Sí	
performance_schema_consumer_events_stages_history_long	Sí	
performance_schema_consumer_events_statements_current	Sí	
performance_schema_consumer_events_statements_history	Sí	
performance_schema_consumer_events_statements_history_long	Sí	
performance_schema_consumer_events_waits_history	Sí	
performance_schema_consumer_events_waits_history_long	Sí	
performance_schema_consumer_statements_digest	Sí	
performance_schema_digests_size	Sí	
performance_schema_events_stages_history_long_size	Sí	
performance_schema_events_stages_history_size	Sí	

Nombre del parámetro	Modificable	Notas
performance_schema_events_statements_history_long_size	Sí	
performance_schema_events_statements_history_size	Sí	
performance_schema_events_transactions_history_long_size	Sí	
performance_schema_events_transactions_history_size	Sí	
performance_schema_events_waits_history_long_size	Sí	
performance_schema_events_waits_history_size	Sí	
performance_schema_hosts_size	Sí	
performance_schema_max_cond_classes	Sí	
performance_schema_max_cond_instances	Sí	
performance_schema_max_digest_length	Sí	
performance_schema_max_file_classes	Sí	
performance_schema_max_file_handles	Sí	

Nombre del parámetro	Modificable	Notas
performance_schema_max_file_instances	Sí	
performance_schema_max_index_stat	Sí	
performance_schema_max_memory_classes	Sí	
performance_schema_max_metadata_locks	Sí	
performance_schema_max_mutex_classes	Sí	
performance_schema_max_mutex_instances	Sí	
performance_schema_max_prepared_statements_instances	Sí	
performance_schema_max_program_instances	Sí	
performance_schema_max_rwlock_classes	Sí	
performance_schema_max_rwlock_instances	Sí	
performance_schema_max_socket_classes	Sí	
performance_schema_max_socket_instances	Sí	
performance_schema_max_sql_text_length	Sí	

Nombre del parámetro	Modificable	Notas
performance_schema_max_stage_classes	Sí	
performance_schema_max_statement_classes	Sí	
performance_schema_max_statement_stack	Sí	
performance_schema_max_table_handles	Sí	
performance_schema_max_table_instances	Sí	
performance_schema_max_table_lock_stat	Sí	
performance_schema_max_thread_classes	Sí	
performance_schema_max_thread_instances	Sí	
performance_schema_session_connect_attrs_size	Sí	
performance_schema_setup_actors_size	Sí	
performance_schema_setup_objects_size	Sí	

Nombre del parámetro	Modificable	Notas
performance_schema_show_processlist	Sí	Este parámetro determina qué implementación SHOW PROCESSLIST utilizar: • La implementación predeterminada se repite en los subprocesos activos desde el administrador de subprocesos mientras mantiene un mutex global. Esto puede provocar un rendimiento lento, especialmente en sistemas ocupados. • La implementación SHOW PROCESSLIST alternativa se basa en la tabla processlist del esquema de rendimiento. Esta implementación consulta los datos del subproceso activo del esquema de rendimiento en lugar del administrador de subprocesos y no requiere un mutex. Este parámetro se aplica a la versión 2.12 y versiones posteriores y a la versión 3 de Aurora MySQL.
performance_schema_users_size	Sí	
pid_file	No	
plugin_dir	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.

Nombre del parámetro	Modificable	Notas
port	No	Aurora MySQL administra las propiedades de conexión e implementa una configuración coherente para todas las instancias de base de datos en un clúster.
preload_buffer_size	Sí	Tamaño del búfer que se asigna al precargar los índices. Este parámetro se aplica a Aurora MySQL versión 3.
profiling_history_size	Sí	
query_alloc_block_size	Sí	
query_cache_limit	Sí	Eliminado de Aurora MySQL versión 3.
query_cache_min_res_unit	Sí	Eliminado de Aurora MySQL versión 3.
query_cache_size	Sí	El valor predeterminado se representa con una fórmula. Para obtener más información sobre cómo se calcula el valor de DBInstanceClassMemory de la fórmula, consulte Variables de las fórmulas de parámetros de base de datos. Eliminado de Aurora MySQL versión 3.
query_cache_type	Sí	Eliminado de Aurora MySQL versión 3.
query_cache_wlock_invalidate	Sí	Eliminado de Aurora MySQL versión 3.
query_prealloc_size	Sí	
range_alloc_block_size	Sí	

Nombre del parámetro	Modificable	Notas
`read_buffer_size`	Sí	

Nombre del parámetro	Modificable	Notas
`read_only`	Sí	Cuando este parámetro está activado, el servidor no permite actualizaciones, excepto las que realizan los subprocesos de réplica. Los valores válidos para Aurora MySQL versión 2 son los siguientes: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` - `ON` para réplicas de lectura. Este es el valor predeterminado. • `{TrueIfClusterReplica}` - `ON` para instancias en clústeres de réplicas, como réplicas de lectura entre regiones, clústeres secundarios en una base de datos global de Aurora e implementaciones azul/verde. Le recomendamos que utilice el grupo de parámetros del clúster de base de datos de la versión 2 de Aurora MySQL para asegurarse de que el parámetro `read_only` se aplica a las nuevas instancias de escritor en caso de conmutación por error. **Note** Las instancias de lector siempre son de solo lectura, porque Aurora MySQL establece

Nombre del parámetro	Modificable	Notas
		`innodb_read_only` en 1 en todos los lectores. Por lo tanto, `read_only` es redundante en las instancias del lector. Eliminado en el nivel de la instancia en Aurora MySQL versión 3.
`read_rnd_buffer_size`	Sí	
`relay-log`	No	
`relay_log_info_repository`	Sí	Eliminado de Aurora MySQL versión 3.
`relay_log_recovery`	No	
`replica_checkpoint_group`	Sí	Aurora MySQL versión 3
`replica_checkpoint_period`	Sí	Aurora MySQL versión 3
`replica_parallel_workers`	Sí	Aurora MySQL versión 3
`replica_pending_jobs_size_max`	Sí	Aurora MySQL versión 3
`replica_skip_errors`	Sí	Aurora MySQL versión 3
`replica_sql_verify_checksum`	Sí	Aurora MySQL versión 3
`safe-user-create`	Sí	
`secure_auth`	Sí	Este parámetro está siempre activado en Aurora MySQL versión 2. Al intentar desactivarlo, se produce un error. Eliminado de Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
secure_file_priv	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
show_create_table_verbosity	Sí	La habilitación de esta variable provoca que SHOW_CREATE_TABLE muestre ROW_FORMAT con independencia de si es el formato predeterminado. Este parámetro se aplica a la versión 2.12 y versiones posteriores y a la versión 3 de Aurora MySQL.
skip-slave-start	No	
skip_external_locking	No	
skip_show_database	Sí	
slave_checkpoint_group	Sí	Aurora MySQL versión 2. Usar replica_checkpoint_group en Aurora MySQL versión 3.
slave_checkpoint_period	Sí	Aurora MySQL versión 2. Usar replica_checkpoint_period en Aurora MySQL versión 3.
slave_parallel_workers	Sí	Aurora MySQL versión 2. Usar replica_parallel_workers en Aurora MySQL versión 3.
slave_pending_jobs_size_max	Sí	Aurora MySQL versión 2. Usar replica_pending_jobs_size_max en Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
slave_sql_verify_checksum	Sí	Aurora MySQL versión 2. Usar replica_sql_verify_checksum en Aurora MySQL versión 3.
slow_launch_time	Sí	
slow_query_log	Sí	Para conocer las instrucciones sobre la carga de los registros en CloudWatch Logs, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.
slow_query_log_file	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
socket	No	
sort_buffer_size	Sí	
sql_mode	Sí	
sql_select_limit	Sí	
stored_program_cache	Sí	
sync_binlog	No	
sync_master_info	Sí	
sync_source_info	Sí	Este parámetro se aplica a Aurora MySQL versión 3.
sync_relay_log	Sí	Eliminado de Aurora MySQL versión 3.
sync_relay_log_info	Sí	
sysdate-is-now	Sí	

Nombre del parámetro	Modificable	Notas
table_cache_element_entry_ttl	No	
table_definition_cache	Sí	El valor predeterminado se represent a con una fórmula. Para obtener más información sobre cómo se calcula el valor de DBInstanceClassMemory de la fórmula, consulte Variables de las fórmulas de parámetros de base de datos.
table_open_cache	Sí	El valor predeterminado se represent a con una fórmula. Para obtener más información sobre cómo se calcula el valor de DBInstanceClassMemory de la fórmula, consulte Variables de las fórmulas de parámetros de base de datos.
table_open_cache_instances	Sí	
temp-pool	Sí	Eliminado de Aurora MySQL versión 3.
temptable_max_mmap	Sí	Este parámetro se aplica a Aurora MySQL versión 3. Para obtener más información, consulte Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3.
temptable_max_ram	Sí	Este parámetro se aplica a Aurora MySQL versión 3. Para obtener más información, consulte Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3.

Nombre del parámetro	Modificable	Notas
temptable_use_mmap	Sí	Este parámetro se aplica a Aurora MySQL versión 3. Para obtener más información, consulte Nuevo comportamiento de tabla temporal en Aurora MySQL versión 3.
thread_cache_size	Sí	La cantidad de subprocesos que se van a almacenar en caché. Este parámetro se aplica a Aurora MySQL versiones 2 y 3.
thread_handling	No	
thread_stack	Sí	
timed_mutexes	Sí	
tmp_table_size	Sí	Define el tamaño máximo de las tablas temporales internas en memoria creadas por el motor de almacenamiento MEMORY en la versión 3 de Aurora MySQL. En la versión 3.04 y versiones posteriores de Aurora MySQL, define el tamaño máximo de las tablas temporales internas en memoria creadas por el motor de almacenamiento TempTable cuando aurora_tmptable_enable_per_table_limit está configurado en ON. Para obtener más información, consulte Limitación del tamaño de las tablas temporales internas en memoria.

Nombre del parámetro	Modificable	Notas
tmpdir	No	Aurora MySQL utiliza instancias administradas en las que no accede al sistema de archivos directamente.
transaction_alloc_block_size	Sí	
transaction_isolation	Sí	Este parámetro se aplica a Aurora MySQL versión 3. Sustituye a tx_isolation .
transaction_prealloc_size	Sí	
tx_isolation	Sí	Eliminado de Aurora MySQL versión 3. Se sustituye por transaction_isolation .
updatable_views_with_limit	Sí	
validate-password	No	
validate_password_dictionary_file	No	
validate_password_length	No	
validate_password_mixed_case_count	No	
validate_password_number_count	No	
validate_password_policy	No	
validate_password_special_char_count	No	

Nombre del parámetro	Modificable	Notas
`wait_timeout`	Sí	Aurora evalúa el valor mínimo de `interactive_timeout` y `wait_timeout`. Utiliza ese mínimo como tiempo de espera para finalizar todas las sesiones inactivas, tanto interactivas como no interactivas.

Parámetros de MySQL que no se aplican a Aurora MySQL

Debido a las diferencias de arquitectura entre Aurora MySQL y MySQL, algunos parámetros de MySQL no se aplican a Aurora MySQL.

Los siguientes parámetros de MySQL no se aplican a Aurora MySQL. Esta lista no es exhaustiva.

- `activate_all_roles_on_login`: este parámetro no se aplica a Aurora MySQL versión 2. Está disponible en Aurora MySQL versión 3.
- `big_tables`
- `bind_address`
- `character_sets_dir`
- `innodb_adaptive_flushing`
- `innodb_adaptive_flushing_lwm`
- `innodb_buffer_pool_chunk_size`
- `innodb_buffer_pool_instances`
- `innodb_change_buffering`
- `innodb_checksum_algorithm`
- `innodb_data_file_path`
- `innodb_dedicated_server`
- `innodb_doublewrite`
- `innodb_flush_log_at_timeout`: este parámetro no se aplica a Aurora MySQL. Para obtener más información, consulte Configuración de la frecuencia de vaciado del búfer de registro.
- `innodb_flush_method`
- `innodb_flush_neighbors`

- innodb_io_capacity
- innodb_io_capacity_max
- innodb_log_buffer_size
- innodb_log_file_size
- innodb_log_files_in_group
- innodb_log_spin_cpu_abs_lwm
- innodb_log_spin_cpu_pct_hwm
- innodb_log_writer_threads
- innodb_max_dirty_pages_pct
- innodb_numa_interleave
- innodb_page_size
- innodb_redo_log_capacity
- innodb_redo_log_encrypt
- innodb_undo_log_encrypt
- innodb_undo_log_truncate
- innodb_undo_logs
- innodb_undo_tablespaces
- innodb_use_native_aio
- innodb_write_io_threads

Variables de estado globales de Aurora MySQL

Aurora MySQL incluye variables de estado de la comunidad MySQL y variables que son exclusivas de Aurora. Puede examinar estas variables para obtener información sobre lo que ocurre dentro del motor de base de datos. Para obtener más información sobre las variables de estado en la comunidad MySQL, consulte Server Status Variables en la documentación de MySQL 8.0 de la comunidad.

Puede encontrar los valores actuales de las variables de estado globales de Aurora MySQL mediante una instrucción como la siguiente:

```
show global status like '%aurora%';
```

La siguiente tabla describe las variables de estado globales que utiliza Aurora MySQL.

Nombre	Descripción
`AuroraDb_commits`	El número total de confirmaciones desde el último reinicio.
`AuroraDb_commit_latency`	La latencia de confirmación agregada desde el último reinicio.
`AuroraDb_ddl_stmt_duration`	La latencia de DDL agregada desde el último reinicio.
`AuroraDb_select_stmt_duration`	La latencia de la instrucción `SELECT` agregada desde el último reinicio.
`AuroraDb_insert_stmt_duration`	La latencia de la instrucción `INSERT` agregada desde el último reinicio.
`AuroraDb_update_stmt_duration`	La latencia de la instrucción `UPDATE` agregada desde el último reinicio.
`AuroraDb_delete_stmt_duration`	La latencia de la instrucción `DELETE` agregada desde el último reinicio.
`Aurora_binlog_io_cache_allocated`	El número de bytes asignados a la memoria caché de E/S de binlog.
`Aurora_binlog_io_cache_read_requests`	El número de solicitudes de lectura realizadas a la memoria caché de E/S.
`Aurora_binlog_io_cache_reads`	El número de solicitudes de lectura que se atendieron desde la memoria caché de E/S.
`Aurora_enhanced_binlog`	Indica si el binlog mejorado está habilitado o deshabilitado para esta instancia de base de datos. Para obtener más información, consulte Configuración del binlog mejorado para Aurora MySQL.

Nombre	Descripción
`Aurora_external_connection_count`	El número de conexiones de base de datos a la instancia de base de datos, excluidas las conexiones al servicio RDS utilizadas para las comprobaciones de estado de la base de datos.
`Aurora_fast_insert_cache_hits`	Un contador que se incrementa cuando el cursor en caché se recupera y se verifica correctamente. Para obtener más informaci ón sobre la inserción rápida en caché, consulte Mejoras del rendimiento de Amazon Aurora MySQL.
`Aurora_fast_insert_cache_misses`	Un contador que se incrementa cuando el cursor en caché deja de ser válido y Aurora realiza un recorrido normal del índice. Para obtener más información sobre la inserción rápida en caché, consulte Mejoras del rendimiento de Amazon Aurora MySQL.
`Aurora_fts_cache_memory_used`	La cantidad de memoria en bytes que utiliza el sistema de búsqueda de texto completo de InnoDB. Esta variable se aplica a Aurora MySQL versión 3.07 y posteriores.
`Aurora_fwd_master_dml_stmt_count`	El número total de instrucciones DML reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 2 de Aurora MySQL.
`Aurora_fwd_master_dml_stmt_ duration`	La duración total de las instrucciones DML reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 2 de Aurora MySQL.

Nombre	Descripción
`Aurora_fwd_master_errors_rpc_timeout`	El número de veces que no se pudo establecer una conexión reenviada en el escritor.
`Aurora_fwd_master_errors_session_limit`	El número de consultas reenviadas que se rechazan debido a `session full` en el escritor.
`Aurora_fwd_master_errors_session_timeout`	El número de veces que finaliza una sesión de reenvío debido a un tiempo de espera del escritor.
`Aurora_fwd_master_open_sessions`	El número de sesiones reenviadas en la instancia de base de datos del escritor. Esta variable se aplica a la versión 2 de Aurora MySQL.
`Aurora_fwd_master_select_stmt_count`	El número total de instrucciones `SELECT` reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 2 de Aurora MySQL.
`Aurora_fwd_master_select_stmt_duration`	La duración total de las instrucciones `SELECT` reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 2 de Aurora MySQL.
`Aurora_fwd_writer_dml_stmt_count`	El número total de instrucciones DML reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 3 de Aurora MySQL.
`Aurora_fwd_writer_dml_stmt_duration`	La duración total de las instrucciones DML reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 3 de Aurora MySQL.

Nombre	Descripción
`Aurora_fwd_writer_errors_rpc_timeout`	El número de veces que no se pudo establecer una conexión reenviada en el escritor.
`Aurora_fwd_writer_errors_session_limit`	El número de consultas reenviadas que se rechazan debido `asession full` en el escritor.
`Aurora_fwd_writer_errors_session_timeout`	El número de veces que finaliza una sesión de reenvío debido a un tiempo de espera del escritor.
`Aurora_fwd_writer_open_sessions`	El número de sesiones reenviadas en la instancia de base de datos del escritor. Esta variable se aplica a la versión 3 de Aurora MySQL.
`Aurora_fwd_writer_select_stmt_count`	El número total de instrucciones `SELECT` reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 3 de Aurora MySQL.
`Aurora_fwd_writer_select_stmt_duration`	La duración total de las instrucciones `SELECT` reenviadas a esta instancia de base de datos del escritor. Esta variable se aplica a la versión 3 de Aurora MySQL.
`Aurora_lockmgr_buffer_pool_memory_used`	Cantidad de memoria del grupo de búfer en bytes que utiliza el administrador de bloqueo de Aurora MySQL.
`Aurora_lockmgr_memory_used`	La cantidad de memoria en bytes que utiliza el administrador de bloqueo de Aurora MySQL.

Nombre	Descripción
`Aurora_ml_actual_request_cnt`	El recuento acumulado de solicitudes que hace Aurora MySQL a los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.
`Aurora_ml_actual_response_cnt`	El recuento acumulado de respuestas que recibe Aurora MySQL de los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.
`Aurora_ml_cache_hit_cnt`	El número acumulado de aciertos de la caché interna que Aurora MySQL recibe de los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.
`Aurora_ml_logical_request_cnt`	La cantidad de solicitudes lógicas que la instancia de base de datos ha evaluado para enviarse a los servicios de machine learning de Aurora desde el último restablecimiento de estado. Dependiendo de si se ha utilizado el procesamiento por lotes, este valor puede ser superior a `Aurora_ml_actual_request_cnt`. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.

Nombre	Descripción
`Aurora_ml_logical_response_cnt`	El recuento acumulado de respuestas que recibe Aurora MySQL de los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.
`Aurora_ml_retry_request_cnt`	La cantidad de solicitudes reintentadas que la instancia de base de datos ha enviado a los servicios de machine learning de Aurora desde el último restablecimiento de estado. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.
`Aurora_ml_single_request_cnt`	El número acumulado de funciones de machine learning de Aurora evaluadas por un modo distinto al modo de lote en todas las consultas ejecutadas por usuarios de la instancia de base de datos. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.
`aurora_oom_avoidance_recovery_state`	Indica si la recuperación de prevención de falta de memoria (OOM) de Aurora está en estado `ACTIVE` o `INACTIVE` para esta instancia de base de datos.

Nombre	Descripción
`aurora_oom_reserved_mem_enter_kb`	Representa el umbral para introducir el estado `RESERVED` en el mecanismo de gestión de OOM de Aurora. Cuando la memoria disponible en el servidor se encuentra por debajo de este umbral, `aurora_oom_status` cambia a `RESERVED`, lo que indica que el servidor se acerca a un nivel crítico de uso de memoria.
`aurora_oom_reserved_mem_exit_kb`	Representa el umbral para salir del estado `RESERVED` en el mecanismo de gestión de OOM de Aurora. Cuando la memoria disponible en el servidor supera este umbral, `aurora_oom_status` vuelve a ser `NORMAL`, lo que indica que el servidor ha vuelto a un estado más estable con recursos de memoria suficientes.
`aurora_oom_status`	Representa el estado de OOM actual de esta instancia de base de datos. Cuando el valor es `NORMAL`, indica que hay suficientes recursos de memoria. Si el valor cambia a `RESERVED`, indica que el servidor tiene poca memoria disponible. En función de la configuración del parámetro `aurora_oom_response`, se adopta una acción u otra. Para obtener más información, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.

Nombre	Descripción
`Aurora_pq_bytes_returned`	El número de bytes de estructuras de datos de tuplas transmitido al nodo director durante las consultas en paralelo. Debe dividirse entre 16 384 para compararse con `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	El número máximo de sesiones de consultas en paralelo que se pueden ejecutar simultáneamente en esta instancia de base de datos Aurora. Este es un número fijo que depende de la clase de instancia de base de datos de AWS.
`Aurora_pq_pages_pushed_down`	El número de páginas de datos (cada una con un tamaño fijo de 16 KiB) en las que las consultas en paralelo evitaron una transmisión de red al nodo director.
`Aurora_pq_request_attempted`	El número de sesiones de consultas en paralelo solicitadas. Este valor podría representar más de una sesión por consulta, dependiendo de los constructos de SQL como subconsultas y uniones.
`Aurora_pq_request_executed`	El número de sesiones de consultas en paralelo ejecutadas correctamente.

Nombre	Descripción
`Aurora_pq_request_failed`	El número de sesiones de consultas en paralelo que devolvieron un error al cliente. En algunos casos, una solicitud de una consulta en paralelo podría producir un error, por ejemplo, debido a un problema en la capa de almacenamiento. En tales casos, la parte de la consulta que haya producido un error vuelve a intentarse usando el mecanismo de consultas no paralelas. Si la consulta reintenta da también produce un error, se devolverá un error al cliente y este contador se incrementará.
`Aurora_pq_request_in_progress`	El número de sesiones de consultas en paralelo en curso actualmente. Este número se aplica a la instancia de base de datos de Aurora concreta a la que se conecta, no a todo el clúster de base de datos de Aurora. Para ver si una instancia de base de datos está cerca de su límite de simultaneidad, compare este valor con el de `Aurora_pq_max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	El número de veces que las consultas en paralelo no se han elegido para satisfacer una consulta. Este valor es la suma de varios otros contadores más detallados. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_below_min_rows`	El número de veces que las consultas en paralelo no se han elegido debido al número de filas de la tabla. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.

Nombre	Descripción
`Aurora_pq_request_not_chosen_column_bit`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas debido a un tipo de datos no admitido en la lista de columnas proyectadas.
`Aurora_pq_request_not_chosen_column_geometry`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene columnas con el tipo de datos `GEOMETRY`. Para obtener información acerca de las versiones de Aurora MySQL que eliminan esta limitación, consulte Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL.
`Aurora_pq_request_not_chosen_column_lob`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene columnas con un tipo de datos `LOB` o columnas `VARCHAR` que se almacenan externamente debido a la longitud declarada. Para obtener información acerca de las versiones de Aurora MySQL que eliminan esta limitación, consulte Actualización de clústeres de consultas en paralelo para la versión 3 de Aurora MySQL.
`Aurora_pq_request_not_chosen_column_virtual`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla contiene una columna virtual.

Nombre	Descripción
`Aurora_pq_request_not_chosen_custom_charset`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene columnas con un conjunto de caracteres personalizado.
`Aurora_pq_request_not_chosen_fast_ddl`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque actualmente se altera la tabla por una instrucción `ALTER` DDL rápida.
`Aurora_pq_request_not_chosen_few_pages_outside_buffer_pool`	El número de veces que las consultas en paralelo no se han elegido, incluso aunque menos del 95 % de los datos de tabla ya estuviera en el grupo de búfer, porque no había suficientes datos de tabla fuera de búfer para que valiera la pena realizar una consulta en paralelo.
`Aurora_pq_request_not_chosen_full_text_index`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla tiene índices de texto completo.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	El número de veces que las consultas en paralelo no se han elegido debido a que un porcentaje elevado de datos de tabla (actualmente, superior al 95 %) ya estaba en el grupo de búfer. En estos casos, el optimizador determina que leer los datos del grupo de búfer es más eficiente. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.

Nombre	Descripción
`Aurora_pq_request_not_chosen_index_hint`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta incluye una sugerencia de índice.
`Aurora_pq_request_not_chosen_innodb_table_format`	El número de solicitudes de consulta en paralelo que utilizan la ruta de procesamiento de consultas no paralelas porque la tabla utiliza un formato de fila de InnoDB no admitido. La consulta en paralelo de Aurora solo se aplica a los formatos de fila `COMPACT`, `REDUNDANT` y `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	El número de solicitudes de consultas en paralelo que usaron la ruta de procesamiento de consultas no en paralelo, debido a que la consulta se estaba iniciando en una transacción de ejecución prolongada. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_no_where_clause`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta no incluye ninguna cláusula `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta utiliza un análisis de intervalo en un índice.
`Aurora_pq_request_not_chosen_row_length_too_long`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la longitud total combinada de todas las columnas es demasiado larga.

Nombre	Descripción
`Aurora_pq_request_not_chosen_small_table`	El número de veces que las consultas en paralelo no se han elegido debido al tamaño general de la tabla, según lo determinado por el número de filas y la longitud promedio de las filas. Una instrucción `EXPLAIN` puede incrementar este contador aunque la consulta no se realiza en realidad.
`Aurora_pq_request_not_chosen_temporary_table`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta hace referencia a tablas temporales que utilizan los tipos de tabla `MyISAM` o `memory` no admitidos.
`Aurora_pq_request_not_chosen_tx_isolation`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta utiliza un nivel de aislamiento de transacciones no admitido. En las instancias de base de datos del lector, la consulta paralela solo se aplica a los niveles de aislamiento `REPEATABLE READ` y `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	El número de solicitudes de consulta paralela que utilizan la ruta de procesamiento de consultas no paralelas porque la consulta forma parte de una instrucción `UPDATE` o `DELETE`.

Nombre	Descripción
`Aurora_pq_request_not_chosen_unsupported_access`	El número de solicitudes de consultas en paralelo que usan la ruta de procesamiento de consultas no en paralelo porque la cláusula `WHERE` no cumple los criterios de consultas en paralelo. Este resultado puede producirse si la consulta no requiere un análisis de uso intensivo de datos o si la consulta es una instrucción `DELETE` o `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	El número de solicitudes de consultas en paralelo que usan la ruta de procesamiento de consultas no en paralelo porque el clúster de base de datos de Aurora MySQL no utiliza una configuración de almacenamiento de clúster de Aurora compatible. Para obtener más información, consulte Limitaciones. Este parámetro se aplica a la versión 3.04 y versiones posteriores de Aurora MySQL.
`Aurora_pq_request_throttled`	El número de veces que las consultas en paralelo no se han elegido debido a que el número máximo de consultas en paralelo simultáneas que ya se están ejecutando en una instancia de base de datos Aurora concreta.
`Aurora_repl_bytes_received`	Número de bytes replicados en una instancia de base de datos del lector de Aurora MySQL desde el último reinicio. Para obtener más información, consulte Replicación con Amazon Aurora MySQL.

Nombre	Descripción
`Aurora_reserved_mem_exceeded_incidents`	El número de veces que el motor ha superado los límites de memoria reservada desde el último reinicio. Si `aurora_oom_response` está configurado, este umbral define cuándo se activan las actividades de evitación de memoria insuficiente (OOM). Para obtener más información sobre la respuesta de OOM de Aurora MySQL, consulte Solución de problemas de memoria insuficiente de bases de datos Aurora MySQL.
`Aurora_thread_pool_thread_count`	El número actual de subprocesos del grupo de subprocesos de Aurora. Para obtener más información sobre el grupo de subprocesos de Aurora MySQL, consulte Grupo de subprocesos.
`Aurora_tmz_version`	Indica la versión actual de la información de zona horaria utilizada por el clúster de base de datos. Los valores siguen el formato IANA (Internet Assigned Numbers Authority): `YYYYsuffix` , por ejemplo, `2022a` y `2023c`. Este parámetro se aplica a la versión 2.12 y versiones posteriores y a la versión 3.04 y versiones posteriores de Aurora MySQL.
`Aurora_zdr_oom_threshold`	Representa el umbral de memoria, en kilobytes (KB), para que una instancia de base de datos Aurora inicie un reinicio sin tiempo de inactividad (ZDR) para recuperarse de posibles problemas relacionados con la memoria.

Nombre	Descripción
server_aurora_das_running	Indica si los flujos de actividad de la base de datos (DAS) están habilitados o deshabilitados en esta instancia de base de datos. Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.

Variables de estado de MySQL que no se aplican a Aurora MySQL

Debido a las diferencias de arquitectura entre Aurora MySQL y MySQL, algunas variables de estado de MySQL no se aplican a Aurora MySQL.

Las siguientes variables de estado de MySQL no se aplican a Aurora MySQL. Esta lista no es exhaustiva.

- innodb_buffer_pool_bytes_dirty
- innodb_buffer_pool_pages_dirty
- innodb_buffer_pool_pages_flushed

Aurora MySQL versión 3 elimina las siguientes variables de estado que estaban en Aurora MySQL versión 2:

- AuroraDb_lockmgr_bitmaps0_in_use
- AuroraDb_lockmgr_bitmaps1_in_use
- AuroraDb_lockmgr_bitmaps_mem_used
- AuroraDb_thread_deadlocks
- available_alter_table_log_entries
- Aurora_lockmgr_memory_used
- Aurora_missing_history_on_replica_incidents
- Aurora_new_lock_manager_lock_release_cnt
- Aurora_new_lock_manager_lock_release_total_duration_micro
- Aurora_new_lock_manager_lock_timeout_cnt

- Aurora_total_op_memory
- Aurora_total_op_temp_space
- Aurora_used_alter_table_log_entries
- Aurora_using_new_lock_manager
- Aurora_volume_bytes_allocated
- Aurora_volume_bytes_left_extent
- Aurora_volume_bytes_left_total
- Com_alter_db_upgrade
- Compression
- External_threads_connected
- Innodb_available_undo_logs
- Last_query_cost
- Last_query_partial_plans
- Slave_heartbeat_period
- Slave_last_heartbeat
- Slave_received_heartbeats
- Slave_retried_transactions
- Slave_running
- Time_since_zero_connections

Estas variables de estado de MySQL están disponibles en la versión 2 de Aurora MySQL, pero no están disponibles en la versión 3 de Aurora MySQL:

- Innodb_redo_log_enabled
- Innodb_undo_tablespaces_total
- Innodb_undo_tablespaces_implicit
- Innodb_undo_tablespaces_explicit
- Innodb_undo_tablespaces_active

Eventos de espera de Aurora MySQL

He aquí algunos eventos de espera frecuentes de Aurora MySQL.

> **Note**
>
> Para obtener información sobre cómo ajustar el rendimiento de Aurora MySQL mediante eventos de espera, consulte Ajuste de Aurora MySQL con eventos de espera.
> Para obtener información sobre las convenciones de nomenclatura de los eventos de espera de MySQL, consulte Performance Schema Instrument Naming Conventions en la documentación de MySQL.

cpu

El número de conexiones activas que están listas para ejecutarse es siempre mayor que el número de vCPU. Para obtener más información, consulte cpu.

io/aurora_redo_log_flush

Una sesión está conservando datos en el almacenamiento de Aurora. Este evento de espera suele ser para una operación de E/S de escritura en Aurora MySQL. Para obtener más información, consulte io/aurora_redo_log_flush.

io/aurora_respond_to_client

El procesamiento de consultas se ha completado y se devuelven los resultados al cliente de la aplicación para las siguientes versiones de Aurora MySQL: 2.10.2 y versiones posteriores a 2.10, 2.09.3 y versiones posteriores a 2.09 y 2.07.7 y versiones posteriores a 2.07. Compare el ancho de banda de red de la clase de instancia de base de datos con el tamaño del conjunto de resultados que se devuelve. Además, verifique los tiempos de respuesta del lado del cliente. Si el cliente no responde y no puede procesar los paquetes TCP, pueden producirse caídas de paquetes y retransmisiones TCP. Esta situación afecta negativamente al ancho de banda de la red. En las versiones anteriores a 2.10.2, 2.09.3 y 2.07.7, el evento de espera incluye erróneamente el tiempo de inactividad. Para obtener información sobre cómo ajustar la base de datos cuando esta espera es destacada, consulte io/aurora_respond_to_client.

io/file/csv/data

Los subprocesos escriben a las tablas con un formato de valores separados por comas (CSV). Compruebe el uso de tablas CSV. Una causa habitual de este evento es que se haya establecido `log_output` en una tabla.

io/file/sql/binlog

Un subproceso está esperando por un archivo de registro binario (binlog) que se está escribiendo en disco.

io/redo_log_flush

Una sesión está conservando datos en el almacenamiento de Aurora. Este evento de espera suele ser para una operación de E/S de escritura en Aurora MySQL. Para obtener más información, consulte io/redo_log_flush.

io/socket/sql/client_connection

El programa `mysqld` está ocupado creando subprocesos para gestionar las nuevas conexiones de clientes entrantes. Para obtener más información, consulte io/socket/sql/client_connection.

io/table/sql/handler

El motor está esperando el acceso a una tabla. Este evento se produce independientemente de si los datos se almacenan en caché en el grupo de búferes o si se accede en el disco. Para obtener más información, consulte io/table/sql/handler.

lock/table/sql/handler

Este evento de espera es un controlador de evento de espera de bloqueo de tabla. Para obtener más información sobre los eventos de átomo y molécula del esquema de rendimiento, consulte Performance Schema Atom and Molecule Events en la documentación de MySQL.

synch/cond/innodb/row_lock_wait

Varias instrucciones de lenguaje de manipulación de datos (DML) acceden a las mismas filas de base de datos al mismo tiempo. Para obtener más información, consulte synch/cond/innodb/row_lock_wait.

synch/cond/innodb/row_lock_wait_cond

Varias instrucciones DML acceden a las mismas filas de base de datos al mismo tiempo. Para obtener más información, consulte synch/cond/innodb/row_lock_wait_cond.

synch/cond/sql/MDL_context::COND_wait_status

Los subprocesos están esperando por un bloqueo de metadatos de tabla. El motor utiliza este tipo de bloqueo para administrar el acceso simultáneo a un esquema de base de datos y garantizar la coherencia de los datos. Para obtener más información, consulte Optimizing Locking Operations en la documentación de MySQL. Para obtener información sobre

cómo ajustar la base de datos cuando este evento es destacado, consulte synch/cond/sql/MDL_context::COND_wait_status.

synch/cond/sql/MYSQL_BIN_LOG::COND_done

Ha activado el registro binario. Puede haber un alto rendimiento de confirmaciones, un gran número de transacciones que se comprometen o réplicas que leen binlogs. Considere utilizar instrucciones de varias filas o agrupar estados de cuenta en una transacción. En Aurora, utilice bases de datos globales en lugar de replicación de registros binarios o utilice los parámetros `aurora_binlog_*`.

synch/mutex/innodb/aurora_lock_thread_slot_futex

Varias instrucciones DML acceden a las mismas filas de base de datos al mismo tiempo. Para obtener más información, consulte synch/mutex/innodb/aurora_lock_thread_slot_futex.

synch/mutex/innodb/buf_pool_mutex

El grupo de búferes no es lo suficientemente grande como para almacenar el conjunto de datos de trabajo. O bien, la carga de trabajo accede a páginas de una tabla específica, lo que genera contención en el grupo de búferes. Para obtener más información, consulte synch/mutex/innodb/buf_pool_mutex.

synch/mutex/innodb/fil_system_mutex

El proceso está esperando el acceso a la memoria caché de memoria del espacio de tabla. Para obtener más información, consulte synch/mutex/innodb/fil_system_mutex.

synch/mutex/innodb/trx_sys_mutex

Las operaciones comprueban, actualizan, eliminan o agregan ID de transacción en InnoDB de forma coherente o controlada. Estas operaciones requieren un llamada mutex de `trx_sys`, a la que se le realiza un seguimiento mediante la instrumentación Performance Schema. Las operaciones incluyen la administración del sistema de transacciones cuando se inicia o cierra la base de datos, reversiones, limpiezas de deshacer, acceso de lectura de filas y cargas de grupos de búfer. La carga elevada de la base de datos con un gran número de transacciones da como resultado la aparición frecuente de este evento de espera. Para obtener más información, consulte synch/mutex/innodb/trx_sys_mutex.

synch/mutex/mysys/KEY_CACHE::cache_lock

El mutex de `keycache->cache_lock` controla el acceso a la caché de claves de las tablas MyISAM. Si bien Aurora MySQL no permite el uso de tablas MyISAM para almacenar datos persistentes, se utilizan para almacenar tablas temporales internas. Considere

la posibilidad de comprobar los contadores con el estado `created_tmp_tables` o `created_tmp_disk_tables`, ya que, en determinadas situaciones, las tablas temporales se escriben en el disco cuando ya no caben en la memoria.

synch/mutex/sql/FILE_AS_TABLE::LOCK_offsets

El motor adquiere este mutex al abrir o crear un archivo de metadatos de tabla. Cuando este evento de espera se produce con una frecuencia excesiva, el número de tablas que se crean o abren ha aumentado.

synch/mutex/sql/FILE_AS_TABLE::LOCK_shim_lists

El motor adquiere este mutex mientras realiza operaciones tales como `reset_size`, `detach_contents`, o bien `add_contents` en la estructura interna que hace un seguimiento de las tablas abiertas. El mutex sincroniza el acceso al contenido de la lista. Cuando este evento de espera se produce con alta frecuencia, indica un cambio repentino en el conjunto de tablas a las que se había accedido anteriormente. El motor necesita acceder a tablas nuevas o dejar de lado el contexto relacionado con las tablas a las que se ha accedido anteriormente.

synch/mutex/sql/LOCK_open

El número de tablas que están abriendo las sesiones supera el tamaño de la caché de definiciones de tablas o de la caché abierta de tablas. Aumente el tamaño de estas cachés. Para obtener más información, consulte How MySQL Opens and Closes Tables.

synch/mutex/sql/LOCK_table_cache

El número de tablas que están abriendo las sesiones supera el tamaño de la caché de definiciones de tablas o de la caché abierta de tablas. Aumente el tamaño de estas cachés. Para obtener más información, consulte How MySQL Opens and Closes Tables.

synch/mutex/sql/LOG

En este evento de espera, hay subprocesos esperando por un bloqueo de log. Por ejemplo, un subproceso podría esperar a un bloqueo para escribir en el archivo log de consultas lentas.

synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit

En este evento de espera, hay un subproceso esperando a adquirir un bloqueo con la intención de efectuar una confirmación en el log binario. La contención de logs binarios se puede producir en las bases de datos cuya velocidad de cambio es muy alta. Según la versión de MySQL, hay algunos bloqueos que se utilizan para proteger la coherencia y durabilidad del log binario. En RDS for MySQL, los registros binarios se utilizan para la replicación y para el proceso de backup automatizado. En Aurora MySQL, los logs binarios no se necesitan para la replicación o las

copias de seguridad nativas. Están deshabilitados de forma predeterminada, pero se pueden habilitar y utilizar para la replicación externa o la captura de datos de cambios. Para obtener más información, consulte The Binary Log en la documentación de MySQL.

sync/mutex/sql/MYSQL_BIN_LOG::LOCK_dump_thread_metrics_collection

Si el registro binario está activado, el motor adquiere este mutex cuando imprime métricas de subprocesos de volcado activos en el registro de errores del motor y en el mapa de operaciones interno.

sync/mutex/sql/MYSQL_BIN_LOG::LOCK_inactive_binlogs_map

Si el registro binario está activado, el motor adquiere este mutex cuando agrega, elimina o busca en la lista de archivos binlog detrás del último.

sync/mutex/sql/MYSQL_BIN_LOG::LOCK_io_cache

Si el registro binario está activado, el motor adquiere este mutex durante las operaciones de caché de E/S binlog de Aurora: asignar, cambiar el tamaño, liberar, escribir, leer, purgar y acceder a la información de la caché. Si este evento se produce con frecuencia, el motor accede a la caché donde se almacenan los eventos binlog. Para reducir los tiempos de espera, reduzca las confirmaciones. Intente agrupar varios estados de cuenta en una sola transacción.

synch/mutex/sql/MYSQL_BIN_LOG::LOCK_log

Ha activado el registro binario. Puede haber un alto rendimiento de confirmaciones, muchas transacciones que se comprometen o réplicas que leen binlogs. Considere utilizar instrucciones de varias filas o agrupar estados de cuenta en una transacción. En Aurora, utilice bases de datos globales en lugar de replicación de registros binarios o utilice los parámetros `aurora_binlog_*`.

synch/mutex/sql/SERVER_THREAD::LOCK_sync

El mutex `SERVER_THREAD::LOCK_sync` se adquiere durante la programación, el procesamiento o el lanzamiento de subprocesos para la escritura de archivos. La ocurrencia excesiva de este evento de espera indica una mayor actividad de escritura en la base de datos.

synch/mutex/sql/TABLESPACES:lock

El motor adquiere el mutex de `TABLESPACES:lock` durante las siguientes operaciones de espacio de tabla: crear, eliminar, truncar y extender. La ocurrencia excesiva de este evento de espera indica una alta frecuencia de operaciones de espacio de tabla. Un ejemplo es cargar una gran cantidad de datos en la base de datos.

synch/rwlock/innodb/dict

En este evento de espera, hay subprocesos esperando por un rwlock aplicado al diccionario de datos de InnoDB.

synch/rwlock/innodb/dict_operation_lock

En este evento de espera, hay subprocesos manteniendo bloqueos en operaciones del diccionario de datos de InnoDB.

synch/rwlock/innodb/dict sys RW lock

Al mismo tiempo se activan un gran número de declaraciones de lenguaje de control de datos (DCL) simultáneas en el código de lenguaje de definición de datos (DDL). Reduzca la dependencia de la aplicación de los DDL durante la actividad regular de la aplicación.

synch/rwlock/innodb/index_tree_rw_lock

Un gran número de instrucciones de lenguaje de manipulación de datos (DML) acceden al mismo objeto de base de datos al mismo tiempo. Intente usar instrucciones de varias filas. Además, distribuya la carga de trabajo sobre distintos objetos de base de datos. Por ejemplo, implementar particiones.

synch/sxlock/innodb/dict_operation_lock

Al mismo tiempo se activan un gran número de declaraciones de lenguaje de control de datos (DCL) simultáneas en el código de lenguaje de definición de datos (DDL). Reduzca la dependencia de la aplicación de los DDL durante la actividad regular de la aplicación.

synch/sxlock/innodb/dict_sys_lock

Al mismo tiempo se activan un gran número de declaraciones de lenguaje de control de datos (DCL) simultáneas en el código de lenguaje de definición de datos (DDL). Reduzca la dependencia de la aplicación de los DDL durante la actividad regular de la aplicación.

synch/sxlock/innodb/hash_table_locks

La sesión no ha encontrado páginas del grupo de búferes. El motor necesita leer un archivo o modificar la lista menos usada recientemente (LRU) para el grupo de búferes. Considere aumentar el tamaño de la caché del búfer y mejorar las rutas de acceso para las consultas pertinentes.

synch/sxlock/innodb/index_tree_rw_lock

Muchas instrucciones de lenguaje de manipulación de datos (DML) acceden al mismo objeto de base de datos al mismo tiempo. Intente usar instrucciones de varias filas. Además, distribuya la carga de trabajo sobre distintos objetos de base de datos. Por ejemplo, implementar particiones.

synch/mutex/innodb/temp_pool_manager_mutex

Este evento de espera se produce cuando una sesión está esperando para adquirir un mutex para administrar el grupo de espacios de tablas temporales de sesión.

Para obtener más información la solución de problemas de los eventos de espera de sincronización, consulte ¿Por qué mi instancia de base de datos MySQL muestra un gran número de sesiones activas esperando en eventos de espera SYNCH en Performance Insights?.

Estados del subproceso Aurora MySQL

Los siguientes son algunos estados de subproceso frecuentes de Aurora MySQL.

checking permissions

El subproceso comprueba si el servidor tiene los privilegios necesarios para ejecutar la instrucción.

checking query cache for query

El servidor comprueba si la consulta actual está presente en la caché de consultas.

cleaned up

Este es el estado final de una conexión cuyo trabajo está completo pero que el cliente no ha cerrado. La mejor solución es cerrar explícitamente la conexión en código. O bien, puede establecer un valor inferior para `wait_timeout` en el grupo de parámetros.

closing tables

El subproceso está vaciando los datos de la tabla modificados en disco y cerrando las tablas usadas. Si no se trata de una operación rápida, verifique las métricas de consumo de ancho de banda de red con respecto al ancho de banda de red de clase de instancia. Además, verifique que los valores de los parámetros de `table_open_cache` y `table_definition_cache` permiten abrir simultáneamente suficientes tablas para que el motor no tenga que abrir y cerrar tablas con frecuencia. Estos parámetros influyen en el consumo de memoria de la instancia.

converting HEAP to MyISAM

La consulta está convirtiendo una tabla temporal de en memoria a en disco. Esta conversión es necesaria porque las tablas temporales creadas por MySQL en los pasos intermedios del procesamiento de consultas crecieron demasiado grandes para la memoria. Verifique los valores de `tmp_table_size` y `max_heap_table_size`. En versiones posteriores, el nombre del estado de este subproceso es `converting HEAP to ondisk`.

converting HEAP to ondisk

El subproceso está convirtiendo una tabla temporal interna de una tabla en memoria a una tabla en disco.

copy to tmp table

El subproceso está procesando una instrucción `ALTER TABLE`. Este estado se produce después de crear la tabla con la nueva estructura, pero antes de copiar las filas en ella. Para un subproceso en este estado, puede utilizar el esquema de rendimiento para obtener información sobre el progreso de la operación de copia.

crear índice de ordenación

Aurora MySQL está realizando una ordenación porque no puede utilizar un índice existente para satisfacer la cláusula `ORDER BY` o `GROUP BY` de una consulta. Para obtener más información, consulte crear índice de ordenación.

creación de tablas

El subproceso está creando una tabla permanente o temporal.

delayed commit ok done

Una confirmación asíncrona en Aurora MySQL ha recibido un acuse de recibo y se ha completado.

delayed commit ok initiated

El subproceso de Aurora MySQL ha iniciado el proceso de confirmación asíncrona, pero está a la espera de confirmación. Por lo general, es el tiempo de confirmación genuino de una transacción.

delayed send ok done

Un subproceso de trabajo de Aurora MySQL vinculado a una conexión se puede liberar mientras se envía una respuesta al cliente. El subproceso puede comenzar otro trabajo. El estado `delayed send ok` significa que se ha completado el acuse de recibo asíncrono al cliente.

delayed send ok initiated

Un subproceso de trabajo de Aurora MySQL ha enviado una respuesta asíncrona a un cliente y ahora es libre de trabajar para otras conexiones. La transacción ha iniciado un proceso de confirmación asíncrono que aún no se ha confirmado.

executing

El subproceso ha comenzado a ejecutar una instrucción.

freeing items

El subproceso ha ejecutado un comando. Parte de la liberación de elementos realizada durante este estado implica la caché de consultas. Este estado suele ir seguido de una limpieza.

init

Este estado se produce antes de la inicialización de las instrucciones `ALTER TABLE`, `DELETE`, `INSERT`, `SELECT`, o bien `UPDATE`. Las acciones en este estado incluyen vaciar el registro binario o el registro InnoDB y una limpieza de la caché de consultas.

El origen ha enviado todo el binlog a la réplica; esperando más actualizaciones

El nodo principal ha finalizado su parte de la replicación. El subproceso está esperando que se ejecuten más consultas para poder escribir en el registro binario (binlog).

opening tables

El subproceso intenta abrir una tabla. Esta operación es rápida a menos que un `ALTER TABLE` o una instrucción `LOCK TABLE` tenga que finalizar o supera el valor de `table_open_cache`.

optimizing

El servidor está realizando optimizaciones iniciales para una consulta.

preparing

Este estado se produce durante la optimización de consultas.

query end

Este estado se produce después de procesar una consulta, pero antes del estado de liberación de elementos.

removing duplicates

Aurora MySQL no pudo optimizar una operación `DISTINCT` en la fase inicial de una consulta. Aurora MySQL debe eliminar todas las filas duplicadas antes de enviar el resultado al cliente.

searching rows for update

El subproceso encuentra todas las filas coincidentes antes de actualizarlas. Esta etapa es necesaria si el `UPDATE` está cambiando el índice que utiliza el motor para buscar las filas.

sending binlog event to slave

El subproceso lee un evento del registro binario y lo envía a la réplica.

sending cached result to client

El servidor está tomando el resultado de una consulta de la caché de consultas y lo envía al cliente.

envío de datos

El subproceso está leyendo y procesando filas para una instrucción `SELECT`, pero aún no ha comenzado a enviar datos al cliente. El proceso identifica qué páginas contienen los resultados necesarios para satisfacer la consulta. Para obtener más información, consulte envío de datos.

sending to client

El servidor está escribiendo un paquete para el cliente. En versiones anteriores de MySQL, este evento de espera estaba etiquetado `writing to net`.

starting

Esta es la primera etapa al comienzo de la ejecución de la declaración.

statistics

El servidor está calculando estadísticas para desarrollar un plan de ejecución de consultas. Si un subproceso está en este estado durante mucho tiempo, es probable que el servidor esté vinculado al disco mientras realiza otro trabajo.

storing result in query cache

El servidor almacena el resultado de una consulta en la caché de consultas.

system lock

El subproceso ha llamado a `mysql_lock_tables`, pero el estado del subproceso no se ha actualizado desde la llamada. Este estado general se produce por muchas razones.

actualización

El subproceso se está preparando para comenzar a actualizar la tabla.

updating

El subproceso busca filas y las está actualizando.

user lock

El subproceso emitió una llamada a `GET_LOCK`. El subproceso solicitó un bloqueo de aviso y lo está esperando, o planea solicitarlo.

waiting for more updates

El nodo principal ha finalizado su parte de la replicación. El subproceso está esperando que se ejecuten más consultas para poder escribir en el registro binario (binlog).

waiting for schema metadata lock

Es una espera para bloquear metadatos.

waiting for stored function metadata lock

Es una espera para bloquear metadatos.

waiting for stored procedure metadata lock

Es una espera para bloquear metadatos.

waiting for table flush

El subproceso está ejecutando `FLUSH TABLES` y está esperando a que todos los subprocesos cierren sus tablas. O el subproceso recibió una notificación de que la estructura subyacente de una tabla ha cambiado, por lo que debe volver a abrir la tabla para obtener la nueva estructura. Para volver a abrir la tabla, el subproceso debe esperar hasta que todos los demás subprocesos hayan cerrado la tabla. Esta notificación se lleva a cabo si otro subproceso ha utilizado una de las siguientes instrucciones de la tabla: `FLUSH TABLES`, `ALTER TABLE`, `RENAME TABLE`, `REPAIR TABLE`, `ANALYZE TABLE`, o bien `OPTIMIZE TABLE`.

waiting for table level lock

Una sesión mantiene un bloqueo en una tabla mientras otra intenta adquirir el mismo bloqueo en la misma tabla.

waiting for table metadata lock

Aurora MySQL utiliza el bloqueo de metadatos para administrar el acceso simultáneo a los objetos de base de datos y garantizar la coherencia de los datos. En este evento de espera, una sesión mantiene un bloqueo de metadatos en una tabla mientras otra sesión intenta adquirir el mismo bloqueo en la misma tabla. Cuando Performance Schema está

habilitado, este estado de subproceso se informa como evento de espera `synch/cond/sql/MDL_context::COND_wait_status`.

writing to net

El servidor está escribiendo un paquete para la red. En versiones posteriores de MySQL, este evento de espera está etiquetado `Sending to client`.

Niveles de aislamiento de Aurora MySQL

Aprenda cómo las instancias de base de datos en un clúster de Aurora MySQL implementan la propiedad de aislamiento de la base de datos. Este tema le ayuda a comprender cómo el comportamiento predeterminado de Aurora MySQL encuentra un equilibrio entre la consistencia estricta y el alto rendimiento. Puede utilizar esta información para decidir cuándo cambiar la configuración predeterminada en función de las características de su carga de trabajo.

Niveles de aislamiento disponibles para las instancias de escrituras

Puede utilizar los niveles de aislamiento `REPEATABLE READ`, `READ COMMITTED`, `READ UNCOMMITTED` y `SERIALIZABLE` en la instancia principal de un clúster de base de datos de Aurora MySQL. Estos niveles de aislamiento funcionan de la misma manera en Aurora MySQL que en RDS for MySQL.

Nivel de aislamiento REPEATABLE READ para las instancias del lector

De manera predeterminada, las instancias de base de datos de Aurora MySQL que están configuradas como réplicas de Aurora de solo lectura siempre utilizan el nivel de aislamiento `REPEATABLE READ`. Estas instancias de base de datos ignoran cualquier instrucción `SET TRANSACTION ISOLATION LEVEL` y continúan utilizando el nivel de aislamiento `REPEATABLE READ`.

Nivel de aislamiento READ COMMITTED para las instancias del lector

Si su aplicación incluye una carga de trabajo de escritura intensiva en la instancia principal y solicitudes de larga ejecución en las réplicas de Aurora, puede experimentar lag de purgado considerable. El lag de purgado se produce cuando la recogida de elementos no utilizados interna se bloquea debido a consultas de ejecución prolongada. El síntoma que ve es un valor alto para `history list length` en el resultado del comando `SHOW ENGINE INNODB STATUS`. Puede supervisar este valor utilizando la métrica `RollbackSegmentHistoryListLength` en

CloudWatch. Un lag de purgado considerable puede reducir la eficacia de los índices secundarios, disminuir el rendimiento general de las consultas y provocar un desperdicio de espacio de almacenamiento.

Si experimenta esos problemas, puede utilizar una opción de configuración de nivel de sesión de Aurora MySQL, `aurora_read_replica_read_committed`, para utilizar el nivel de aislamiento de `READ COMMITTED` en réplicas de Aurora. Cuando utilice esta configuración, puede ayudar a reducir las ralentizaciones y el espacio desaprovechado que pueda derivarse de la realización de consultas de ejecución prolongada al mismo tiempo que las transacciones que modifican sus tablas.

Recomendamos que se asegure de que comprende el comportamiento específico de Aurora MySQL del aislamiento `READ COMMITTED` antes de utilizar esta configuración. La réplica de Aurora del comportamiento de `READ COMMITTED` cumple con el estándar ANSI SQL. Sin embargo, el aislamiento es menos estricto que el comportamiento `READ COMMITTED` de MySQL típico con el que puede que esté más familiarizado. Por lo tanto, puede ser que vea resultados de consultas diferentes en `READ COMMITTED` en una réplica de lectura de Aurora MySQL que para la misma consulta en `READ COMMITTED` en la instancia principal de Aurora MySQL o en RDS para MySQL. Puede considerar utilizar la configuración de `aurora_read_replica_read_committed` para esos casos de uso como un informe completo que analiza una base de datos muy grande. Por el contrario, puede evitarla para consultas breves con conjuntos de resultados pequeños, donde la precisión y la repetibilidad son importantes.

El nivel de aislamiento `READ COMMITTED` no está disponible para las sesiones de un clúster secundario de una base de datos global de Aurora que utilicen la función de reenvío de escritura. Para obtener información sobre el reenvío de escritura, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

Uso de READ COMMITTED para lectores

Para utilizar el nivel de aislamiento `READ COMMITTED` para las réplicas de Aurora, establezca la opción de configuración `aurora_read_replica_read_committed` en `ON`. Utilice esta configuración en el nivel de sesión mientras esté conectado a una réplica de Aurora específica. Para ello, ejecute los comandos de SQL siguientes.

```
set session aurora_read_replica_read_committed = ON;
set session transaction isolation level read committed;
```

Puede utilizar esta opción de configuración de manera temporal para realizar consultas ad hoc únicas e interactivas. Puede que también desee ejecutar una aplicación de informes o de análisis de

datos que se beneficie del nivel de aislamiento `READ COMMITTED`, mientras deja el predeterminado sin cambiar para otras aplicaciones.

Cuando la configuración `aurora_read_replica_read_committed` está activada, utilice el comando `SET TRANSACTION ISOLATION LEVEL` para especificar el nivel de aislamiento para las transacciones adecuadas.

```
set transaction isolation level read committed;
```

Diferencias en el comportamiento READ COMMITTED en las réplicas de Aurora

La configuración `aurora_read_replica_read_committed` hace que el nivel de aislamiento `READ COMMITTED` esté disponible para una réplica de Aurora, con un comportamiento de consistencia que está optimizado para las transacciones de ejecución prolongada. El nivel de aislamiento `READ COMMITTED` en las réplicas de Aurora tiene un aislamiento menos estricto que en las instancias principales de Aurora. Por ese motivo, habilite esta configuración solo en las réplicas de Aurora en las que sabe que sus consultas pueden aceptar la posibilidad de ciertos tipos de resultados inconsistentes.

Sus consultas pueden experimentar ciertos tipos de anomalías de lectura cuando se enciende la configuración `aurora_read_replica_read_committed`. Dos tipos de anomalías son especialmente importantes para entender y manejar su código de aplicación. Una lectura no repetible se produce cuando otra transacción se confirma mientras su consulta está en ejecución. Una consulta de ejecución prolongada puede ver diferentes datos al principio de la consulta que los que ve al final. Una lectura fantasma se produce cuando otras transacciones hacen que las filas existentes se reorganicen mientras su consulta se ejecuta y su consulta lee dos veces una o más filas.

Sus consultas pueden experimentar recuentos de filas inconsistentes como resultado de las lecturas fantasmas. Puede que sus consultas también devuelvan resultados incompletos o inconsistentes debido a las lecturas no repetibles. Por ejemplo, suponga que una operación conjunta hace referencia a tablas que las instrucciones SQL modifican de forma simultánea como `INSERT` o `DELETE`. En este caso, la consulta conjunta puede leer una fila de una tabla pero no la fila correspondiente de otra tabla.

El estándar ANSI SQL permite a estos comportamientos el nivel de aislamiento `READ COMMITTED`. Sin embargo, estos comportamientos son diferentes de la implementación típica de MySQL de `READ COMMITTED`. Por lo tanto, antes de habilitar la configuración

`aurora_read_replica_read_committed`, compruebe cualquier código existente de SQL para verificar si opera según lo esperado en el modelo de consistencia secundario.

Puede que los recuentos de filas y otros resultados no sean altamente consistentes en el nivel de aislamiento `READ COMMITTED` mientras se habilita este nivel de aislamiento. Por lo tanto, típicamente habilita la configuración solo mientras ejecuta consultas analíticas que agregan grandes cantidades de datos y no precisan una precisión absoluta. SI no tiene estos tipos de consultas de ejecución prolongada junto a una carga de trabajo de escritura intensiva, probablemente no necesite la configuración `aurora_read_replica_read_committed`. Sin la combinación de solicitudes de ejecución prolongada y una carga de trabajo de escritura intensiva, no es probable que tenga problemas con extensión de la lista del historial.

Example Consultas que muestran un comportamiento de aislamiento para READ COMMITTED en las réplicas de Aurora

El siguiente ejemplo le muestra cómo las consultas `READ COMMITTED` en una réplica de Aurora puede devolver resultados no repetibles si las transacciones modifican las tablas asociadas al mismo tiempo. La tabla `BIG_TABLE` contiene 1 millón de filas antes de que comience cualquier consulta. Otras instrucciones del lenguaje de manipulación de datos (DML) añaden, eliminan o cambian filas mientras se ejecutan.

Las consultas en la instancia principal de Aurora en el nivel de aislamiento `READ COMMITTED` producen resultados predecibles. Sin embargo, la sobrecarga de mantener la vista de lectura consistente durante la vida útil de cada consulta de ejecución prolongada puede llevar a una recopilación de elementos no utilizados cara más adelante.

Las consultas en la réplica de Aurora en el nivel de aislamiento `READ COMMITTED` se optimizan para minimizar esta sobrecarga de recopilación de elementos no utilizados. La desventaja es que los resultados pueden variar dependiendo de si las consultas recuperan filas que se han añadido, eliminado o reorganizado por transacciones que se confirman cuando la consulta se ejecuta. Las consultas pueden tener en cuenta estas filas pero no están obligadas a ello. Para fines demostrativos, las consultas comprueban solo el número de filas en la tabla utilizando la función `COUNT(*)`.

Time	Instrucción DML en la instancia principal de Aurora	Consulta en la instancia principal de Aurora con READ COMMITTED	Consulta en la réplica de Aurora con READ COMMITTED
T1	`INSERT INTO big_table SELECT * FROM other_table LIMIT 1000000; COMMIT;`		
T2		C1: `SELECT COUNT(*) FROM big_table;`	C2: `SELECT COUNT(*) FROM big_table;`
T3	`INSERT INTO big_table (c1, c2) VALUES (1, 'one more row'); COMMIT;`		
T4		Si C1 termina ahora, el resultado es 1 000 000.	Si C2 termina ahora, el resultado es 1 000 000 o 1 000 001.
T5	`DELETE FROM big_table LIMIT 2; COMMIT;`		
T6		Si C1 termina ahora, el resultado es 1 000 000.	Si C2 termina ahora, el resultado es 1 000 000 o 1 000 001, o 999 999 o 999 998.
T7	`UPDATE big_table SET c2 =`		

Time	Instrucción DML en la instancia principal de Aurora	Consulta en la instancia principal de Aurora con READ COMMITTED	Consulta en la réplica de Aurora con READ COMMITTED
	`CONCAT(c2,c2,c2); COMMIT;`		
T8		Si C1 termina ahora, el resultado es 1 000 000.	Si C2 termina ahora, el resultado es 1 000 000 o 1 000 001, o 999 999 o posiblemente algún número mayor.
T9		C3: `SELECT COUNT(*) FROM big_table;`	C4: `SELECT COUNT(*) FROM big_table;`
T10		Si C3 termina ahora, el resultado es 999 999.	Si C4 termina ahora, el resultado es 999 999.
T11		C5: `SELECT COUNT(*) FROM parent_table p JOIN child_table c ON (p.id = c.id) WHERE p.id = 1000;`	C6: `SELECT COUNT(*) FROM parent_table p JOIN child_table c ON (p.id = c.id) WHERE p.id = 1000;`

Time	Instrucción DML en la instancia principal de Aurora	Consulta en la instancia principal de Aurora con READ COMMITTED	Consulta en la réplica de Aurora con READ COMMITTED
T12	`INSERT INTO parent_table (id, s) VALUES (1000, 'hello'); INSERT INTO child_table (id, s) VALUES (1000, 'world'); COMMIT;`		
T13		Si C5 termina ahora, el resultado es 0.	Si C6 termina ahora, el resultado es 0 o 1.

Si las consultas terminan rápidamente, antes de que cualquier otra transacción realice instrucciones DML y los envíe, los resultados son predecibles y lo mismo ocurre entre la instancia primaria y la réplica de Aurora. Examinemos las diferencias de comportamiento en detalle, empezando por la primera consulta.

Los resultados para C1 son muy predecibles, ya que `READ COMMITTED` en la instancia principal utiliza un modelo de coherencia alta similar al del nivel de aislamiento `REPEATABLE READ`.

Los resultados para C2 puede variar dependiendo de las transacciones que se confirman mientras esa consulta se ejecuta. Por ejemplo, suponga que otras transacciones realizan instrucciones DML y las confirman mientras las consultas se ejecutan. En este caso, la consulta en la réplica de Aurora con el nivel de aislamiento `READ COMMITTED` puede o no tener en cuenta los cambios, Los recuentos de filas no se pueden predecir de la misma manera que en el nivel de aislamiento `REPEATABLE READ`. Tampoco se pueden predecir como consultas que se ejecutan en el nivel de aislamiento `READ COMMITTED` en la instancia principal o en una instancia de RDS for MySQL.

El estándar `UPDATE` en T7 no cambia en realidad el número de filas en la tabla. Sin embargo, al cambiar la extensión de una columna de extensión variable, esta instrucción puede hacer que las filas se reorganicen de manera interna. Una transacción `READ COMMITTED` de ejecución prolongada

puede observar la versión antigua de una fila y más adelante, en la misma consulta, observar la nueva versión de la misma fila. La consulta también puede omitir las versiones nueva y antigua de la fila, por lo que el recuento de filas puede ser diferente de lo esperado.

Los resultados de C5 y C6 pueden ser idénticos o ligeramente diferentes. La consulta C6 en la réplica de Aurora en `READ COMMITTED` puede observar, pero no está obligada a ello, las filas nuevas se confirman mientras que la consulta se ejecuta. También podría ver la fila de una tabla, pero no la de la otra. Si la consulta conjunta no encuentra una fila que coincida en ambas tablas, devuelve un recuento de cero. Si la consulta encuentra las dos filas en `PARENT_TABLE` y `CHILD_TABLE`, la consulta devuelve un recuento de uno. En una consulta de ejecución prolongada, las búsquedas de las tablas combinadas puede producirse en momentos muy separados.

> **Note**
>
> Estas diferencias de comportamiento depende del momento en el que se confirmen las transacciones y el momento en el que las consultas procesan las filas subyacentes de la tabla. Por lo tanto, es muy probable que vea esas diferencias en consultas de informes que tardan minutos u horas y que se ejecutan en clústeres de Aurora que procesan transacciones de OLTP al mismo tiempo. Estos son los tipos de cargas de trabajo mixtas que más se benefician del nivel de aislamiento `READ COMMITTED` en las réplicas de Aurora.

Sugerencias de Aurora MySQL

Puede utilizar consejos de SQL con consultas de Aurora MySQL para ajustar el rendimiento. También puede utilizar sugerencias para evitar que los planes de ejecución de consultas importantes cambien en función de condiciones impredecibles.

> **Tip**
>
> Para comprobar el efecto que tiene una sugerencia en una consulta, examine el plan de consulta producido por la instrucción `EXPLAIN`. Compare los planes de consulta con y sin la sugerencia.

En Aurora MySQL versión 3, puede utilizar todas las sugerencias disponibles en MySQL Community Edition 8.0. Para obtener detalles sobre estas sugerencias, consulte Optimizer Hints (Sugerencias del optimizador) en el Manual de referencia de MySQL.

Las siguientes sugerencias están disponibles en Aurora MySQL versión 2. Estas sugerencias se aplican a consultas que utilizan la característica de combinación hash de Aurora MySQL versión 2, especialmente a consultas que utilizan la optimización de consultas paralelas.

PQ, NO_PQ

Especifica si se debe obligar al optimizador a utilizar consultas paralelas por tabla o por consulta.

PQ obliga al optimizador a utilizar una consulta paralela para las tablas especificadas o para toda la consulta (bloque). NO_PQ impide que el optimizador utilice consultas paralelas para tablas especificadas o para toda la consulta (bloque).

Esta sugerencia está disponible en Aurora MySQL 2.11 y versiones posteriores. En los siguientes ejemplos se muestra cómo usar esta sugerencia.

> **Note**
>
> Al especificar un nombre de tabla, el optimizador se ve obligado a aplicar la sugerencia PQ/NO_PQ solo en las tablas seleccionadas. Si no se especifica un nombre de tabla, se aplicará la sugerencia PQ/NO_PQ a todas las tablas afectadas por el bloque de consulta.

```
EXPLAIN SELECT /*+ PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;

EXPLAIN SELECT /*+ NO_PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;
```

HASH_JOIN, NO_HASH_JOIN

Activa o desactiva la capacidad del optimizador de consultas paralelas de elegir si desea usar el método de optimización de combinación hash para una consulta. `HASH_JOIN` permite al optimizador usar la combinación hash si ese mecanismo es más eficiente. `NO_HASH_JOIN` impide que el optimizador utilice la combinación hash para la consulta. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores. No tiene efecto en Aurora MySQL versión 3.

En los siguientes ejemplos se muestra cómo usar esta sugerencia.

```
EXPLAIN SELECT/*+ HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ NO_HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_PROBING, NO_HASH_JOIN_PROBING

En una consulta de combinación hash, especifica si se va a utilizar la tabla especificada para el lado de sondeo de la combinación. La consulta comprueba si los valores de columna de la tabla de compilación existen en la tabla de sondeo, en lugar de leer todo el contenido de la tabla de sondeo. Puede utilizar `HASH_JOIN_PROBING` y `HASH_JOIN_BUILDING` para especificar cómo se procesan las consultas de combinación hash sin reordenar las tablas dentro del texto de la consulta. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores. No tiene efecto en Aurora MySQL versión 3.

En los siguientes ejemplos se muestra cómo usar esta sugerencia. Especificar la sugerencia `HASH_JOIN_PROBING` para la tabla `T2` tiene el mismo efecto que especificar `NO_HASH_JOIN_PROBING` para la tabla `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_PROBING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_PROBING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_BUILDING, NO_HASH_JOIN_BUILDING

En una consulta de combinación hash, especifica si se va a utilizar la tabla especificada para el lado de compilación de la combinación. La consulta procesa todas las filas de esta tabla para

crear la lista de valores de columna para hacer referencia cruzada con la otra tabla. Puede utilizar HASH_JOIN_PROBING y HASH_JOIN_BUILDING para especificar cómo se procesan las consultas de combinación hash sin reordenar las tablas dentro del texto de la consulta. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores. No tiene efecto en Aurora MySQL versión 3.

En el siguiente ejemplo se muestra cómo usar esta sugerencia. Especificar la sugerencia HASH_JOIN_BUILDING para la tabla T2 tiene el mismo efecto que especificar NO_HASH_JOIN_BUILDING para la tabla T1.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_BUILDING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_BUILDING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

JOIN_FIXED_ORDER

Especifica que las tablas de la consulta se unen en función del orden en que aparecen en la consulta. Es útil con consultas que afectan a tres o más tablas. Sirve de reemplazo para la sugerencia de MySQL STRAIGHT_JOIN y es equivalente a la sugerencia de MySQLJOIN_FIXED_ORDER. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores.

En el siguiente ejemplo se muestra cómo usar esta sugerencia.

```
EXPLAIN SELECT /*+ JOIN_FIXED_ORDER() */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_ORDER

Especifica el orden de combinación de las tablas de la consulta. Es útil con consultas que afectan a tres o más tablas. Es equivalente a la sugerencia de MySQL JOIN_ORDER. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores.

En el siguiente ejemplo se muestra cómo usar esta sugerencia.

```
EXPLAIN SELECT /*+ JOIN_ORDER (t4, t2, t1, t3) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_PREFIX

Especifica las tablas que se van a poner primero en el orden de combinación. Es útil con consultas que afectan a tres o más tablas. Es equivalente a la sugerencia de MySQL JOIN_PREFIX. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores.

En el siguiente ejemplo se muestra cómo usar esta sugerencia.

```
EXPLAIN SELECT /*+ JOIN_PREFIX (t4, t2) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_SUFFIX

Especifica las tablas que se van a poner en último lugar en el orden de combinación. Es útil con consultas que afectan a tres o más tablas. Es equivalente a la sugerencia de MySQL JOIN_SUFFIX. Esta sugerencia está disponible en Aurora MySQL versión 2.08 y versiones posteriores.

En el siguiente ejemplo se muestra cómo usar esta sugerencia.

```
EXPLAIN SELECT /*+ JOIN_SUFFIX (t1) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

Para obtener información sobre el uso de consultas de combinación hash, consulte Optimización de grandes consultas combinadas de Aurora MySQL con combinaciones hash.

Referencia de procedimientos almacenados en Aurora MySQL

Para administrar el clúster de base de datos de Aurora MySQL, invoque los procedimientos almacenados integrados.

Temas

- Recopilación y mantenimiento del historial de estado global
- Configuración, inicio y detención de la replicación del registro binario (binlog)
- Finalización de una sesión o una consulta
- Replicación de transacciones mediante GTID
- Rotación de los registros de consultas

- Establecimiento y muestra de la configuración del registro binario

Recopilación y mantenimiento del historial de estado global

Amazon RDS proporciona un conjunto de procedimientos que crean instantáneas de los valores de las variables de estado a lo largo del tiempo y los escriben en una tabla, junto con cualquier cambio desde la última instantánea. Esta infraestructura se denomina Historial de estado global. Para obtener más información, consulte Managing the Global Status History (Administrar el historial de estado global).

Los siguientes procedimientos almacenados administran la forma en que se recopila y mantiene el historial de estado global.

Temas

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

Toma una instantánea bajo demanda para el historial de estado global.

Sintaxis

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

Desactiva las instantáneas tomadas por el historial de estado global.

Sintaxis

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

Desactiva la rotación de la tabla `mysql.global_status_history`.

Sintaxis

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

Activa el historial de estado global para tomar instantáneas predeterminadas a los intervalos especificados por `rds_set_gsh_collector`.

Sintaxis

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

Activa la rotación del contenido de la tabla `mysql.global_status_history` a `mysql.global_status_history_old` a los intervalos especificados por `rds_set_gsh_rotation`.

Sintaxis

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

Rota el contenido de la tabla `mysql.global_status_history` a `mysql.global_status_history_old` a petición.

Sintaxis

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

Especifica el intervalo, en minutos, entre las instantáneas tomadas por el historial de estado global.

Sintaxis

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parámetros

intervalPeriod

El intervalo, en minutos, entre snapshots. El valor predeterminado es 5.

mysql.rds_set_gsh_rotation

Especifica el intervalo, en días, entre rotaciones de la tabla `mysql.global_status_history`.

Sintaxis

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parámetros

intervalPeriod

El intervalo, en días, entre rotaciones de la tabla. El valor predeterminado es 7.

Configuración, inicio y detención de la replicación del registro binario (binlog)

Puede invocar los siguientes procedimientos almacenados cuando esté conectado a la instancia principal en un clúster de Aurora MySQL. Estos procedimientos controlan la forma en la que se replican las transacciones desde una base de datos externa en Aurora MySQL o desde Aurora MySQL a una base de datos externa.

Temas

- mysql.rds_disable_session_binlog (Aurora MySQL versión 2)
- mysql.rds_enable_session_binlog (Aurora MySQL versión 2)
- mysql.rds_import_binlog_ssl_material
- mysql.rds_next_master_log (Aurora MySQL versión 2)
- mysql.rds_next_source_log (Aurora MySQL versión 3)
- mysql.rds_remove_binlog_ssl_material
- mysql.rds_reset_external_master (Aurora MySQL versión 2)
- mysql.rds_reset_external_source (Aurora MySQL versión 3)
- mysql.rds_set_binlog_source_ssl (Aurora MySQL versión 3)
- mysql.rds_set_external_master (Aurora MySQL versión 2)
- mysql.rds_set_external_source (Aurora MySQL versión 3)
- mysql.rds_set_external_master_with_auto_position (Aurora MySQL versión 2)
- mysql.rds_set_external_source_with_auto_position (Aurora MySQL versión 3)
- mysql.rds_set_master_auto_position (Aurora MySQL versión 2)
- mysql.rds_set_read_only (Aurora MySQL versión 3)
- mysql.rds_set_session_binlog_format (Aurora MySQL versión 2)
- mysql.rds_set_source_auto_position (Aurora MySQL versión 3)
- mysql.rds_skip_repl_error
- mysql.rds_start_replication
- mysql.rds_start_replication_until (versión 3 de Aurora MySQL)
- mysql.rds_stop_replication

mysql.rds_disable_session_binlog (Aurora MySQL versión 2)

Desactiva el registro binario de la sesión actual mediante la configuración de la variable `sql_log_bin` a `OFF`.

Sintaxis

```
CALL mysql.rds_disable_session_binlog;
```

Parámetros

Ninguno

Notas de uso

Para un clúster de bases de datos de Aurora MySQL, puede invocar este procedimiento almacenado cuando esté conectado a la instancia principal.

Para Aurora, este procedimiento se admite en la versión 2.12 de Aurora MySQL y versiones posteriores compatibles con MySQL 5.7.

> **Note**
>
> En la versión 3 de Aurora MySQL, puede usar el siguiente comando para deshabilitar el registro binario de la sesión actual si tiene el privilegio:`SESSION_VARIABLES_ADMIN`
>
> ```
> SET SESSION sql_log_bin = OFF;
> ```

mysql.rds_enable_session_binlog (Aurora MySQL versión 2)

Activa el registro binario de la sesión actual mediante la configuración de la variable `sql_log_bin` a `ON`.

Sintaxis

```
CALL mysql.rds_enable_session_binlog;
```

Parámetros

Ninguno

Notas de uso

Para un clúster de bases de datos de Aurora MySQL, puede invocar este procedimiento almacenado cuando esté conectado a la instancia principal.

Para Aurora, este procedimiento se admite en la versión 2.12 de Aurora MySQL y versiones posteriores compatibles con MySQL 5.7.

> **Note**
>
> En la versión 3 de Aurora MySQL, puede usar el siguiente comando para habilitar el registro binario de la sesión actual si tiene el privilegio:`SESSION_VARIABLES_ADMIN`
>
> ```
> SET SESSION sql_log_bin = ON;
> ```

mysql.rds_import_binlog_ssl_material

Importa el certificado de la entidad de certificación, el certificado de cliente y la clave de cliente a un clúster de base de datos de Aurora MySQL. La información es necesaria para la comunicación SSL y la replicación cifrada.

> **Note**
>
> Actualmente, este procedimiento es compatible con la versión 2: 2.09.2, 2.10.0, 2.10.1 y 2.11.0; y la versión 3: 3.01.1 y posteriores de Aurora MySQL.

Sintaxis

```
CALL mysql.rds_import_binlog_ssl_material (
  ssl_material
);
```

Parámetros

`ssl_material`

Carga JSON que incluye el contenido de los siguientes archivos con formato .pem para un cliente MySQL:

- "ssl_ca":"*certificado de la entidad de certificación*"
- "ssl_cert":"*certificado cliente*"
- "ssl_key":"*clave cliente*"

Notas de uso

Prepare la replicación cifrada antes de ejecutar este procedimiento:

- Si no tiene SSL habilitado en la instancia de base de datos de origen MySQL externa y no dispone de una clave cliente ni de un certificado cliente, habilite SSL en el servidor de base de datos de MySQL y genere la clave cliente y el certificado cliente necesarios.
- Si SSL está habilitado en la instancia de base de datos de origen externa, proporcione una clave cliente y un certificado cliente para el clúster de base de datos de Aurora MySQL. Si no los tiene, genere una nueva clave y certificado para el clúster de base de datos Aurora MySQL. Para firmar el certificado cliente, debe tener la clave de la entidad de certificación que usó para configurar SSL en la instancia de base de datos de origen MySQL externa.

Para obtener más información, consulte Creating SSL Certificates and Keys Using openssl en la documentación de MySQL.

> **Important**
>
> Después de preparar la replicación cifrada, use una conexión SSL para ejecutar este procedimiento. La clave cliente no se debe transferir en una conexión que no sea segura.

Este procedimiento importa la información de SSL de una base de datos MySQL externa a un clúster de base de datos Aurora MySQL. La información de SSL está en archivos con formato .pem que contienen la información de SSL del clúster de base de datos Aurora MySQL. Durante la replicación cifrada, el clúster de base de datos Aurora MySQL actúa como un cliente en el servidor de base de datos MySQL. Los certificados y las claves del cliente de Aurora MySQL son archivos con formato .pem.

Puede copiar la información de estos archivos en el parámetro `ssl_material` en la carga JSON correcta. Para permitir la replicación cifrada, importe esta información de SSL en el clúster de base de datos Aurora MySQL.

La carga JSON debe tener el siguiente formato.

```
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
ssl_ca_pem_body_code
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
ssl_cert_pem_body_code
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
ssl_key_pem_body_code
-----END RSA PRIVATE KEY-----\n"}'
```

Ejemplos

En el siguiente ejemplo, se importa la información de SSL en Aurora MySQL. En los archivos con formato .pem, el código del cuerpo suele ser más grande que el que se muestra en el ejemplo.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

mysql.rds_next_master_log (Aurora MySQL versión 2)

Cambia la posición del registro de instancia de base de datos de origen al inicio del siguiente registro binario en la instancia de base de datos de origen. Use este procedimiento únicamente si aparece el error de E/S de replicación 1236 en una réplica de lectura.

Sintaxis

```
CALL mysql.rds_next_master_log(
curr_master_log
);
```

Parámetros

curr_master_log

El índice del archivo de registro maestro actual. Por ejemplo, si el nombre del archivo actual es `mysql-bin-changelog.012345`, el índice es 12345. Para determinar el nombre del archivo de log maestro actual, ejecute el comando `SHOW REPLICA STATUS` y vea el campo `Master_Log_File`.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_next_master_log`.

> **Warning**
>
> Llame a `mysql.rds_next_master_log` solo si la replicación deja de funcionar tras una conmutación por error de una instancia de base de datos Multi-AZ que es el origen de la replicación y el campo `Last_IO_Errno` de `SHOW REPLICA STATUS` muestra el error de E/S 1236.
>
> La llamada a `mysql.rds_next_master_log` puede provocar una pérdida de datos en la réplica de lectura si las transacciones de la instancia de origen no se escribieron en el registro binario en el disco antes del evento de conmutación por error.

Ejemplos

Supongamos que la replicación falla en una réplica de lectura de Aurora MySQL. La ejecución de `SHOW REPLICA STATUS\G` en la réplica de lectura devuelve el siguiente resultado:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
```

```
                  Source_Port: 3306
                 Connect_Retry: 10
               Source_Log_File: mysql-bin-changelog.012345
           Read_Source_Log_Pos: 1219393
                Relay_Log_File: relaylog.012340
                 Relay_Log_Pos: 30223388
         Relay_Source_Log_File: mysql-bin-changelog.012345
            Replica_IO_Running: No
           Replica_SQL_Running: Yes
               Replicate_Do_DB:
           Replicate_Ignore_DB:
            Replicate_Do_Table:
        Replicate_Ignore_Table:
       Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

El campo `Last_IO_Errno` muestra que la instancia ha recibido el error de E/S 1236. El campo `Master_Log_File` muestra que el nombre de archivo es `mysql-bin-changelog.012345`, lo que significa que el índice del archivo de registro es `12345`. Para resolver el error, puede llamar a `mysql.rds_next_master_log` con el siguiente parámetro:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log (Aurora MySQL versión 3)

Cambia la posición del registro de instancia de base de datos de origen al inicio del siguiente registro binario en la instancia de base de datos de origen. Use este procedimiento únicamente si aparece el error de E/S de replicación 1236 en una réplica de lectura.

Sintaxis

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

Parámetros

curr_source_log

El índice del archivo de registro de origen actual. Por ejemplo, si el nombre del archivo actual es `mysql-bin-changelog.012345`, el índice es 12345. Para determinar el nombre del archivo de registro de origen actual, ejecute el comando `SHOW REPLICA STATUS` y vea el campo `Source_Log_File`.

Notas de uso

El usuario administrativo debe ejecutar el procedimiento `mysql.rds_next_source_log`.

> **Warning**
>
> Llame a `mysql.rds_next_source_log` solo si la replicación deja de funcionar tras una conmutación por error de una instancia de base de datos Multi-AZ que es el origen de la replicación y el campo `Last_IO_Errno` de `SHOW REPLICA STATUS` muestra el error de E/S 1236.

La llamada a `mysql.rds_next_source_log` puede provocar una pérdida de datos en la réplica de lectura si las transacciones de la instancia de origen no se escribieron en el registro binario en el disco antes del evento de conmutación por error. Puede reducir el riesgo de que esto ocurra configurando los parámetros de la instancia de origen `sync_binlog` y `innodb_support_xa` en `1`, aunque esto podría reducir el rendimiento.

Ejemplos

Supongamos que la replicación falla en una réplica de lectura de Aurora MySQL. La ejecución de `SHOW REPLICA STATUS\G` en la réplica de lectura devuelve el siguiente resultado:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
```

```
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

El campo `Last_IO_Errno` muestra que la instancia ha recibido el error de E/S 1236. El campo `Source_Log_File` muestra que el nombre de archivo es `mysql-bin-changelog.012345`, lo que significa que el índice del archivo de registro es `12345`. Para resolver el error, puede llamar a `mysql.rds_next_source_log` con el siguiente parámetro:

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_remove_binlog_ssl_material

Elimina el certificado de la entidad de certificación, el certificado cliente y la clave cliente para las comunicaciones SSL y la replicación cifrada. Esta información se importa mediante mysql.rds_import_binlog_ssl_material.

Sintaxis

```
CALL mysql.rds_remove_binlog_ssl_material;
```

mysql.rds_reset_external_master (Aurora MySQL versión 2)

Vuelve a configurar una instancia de base de datos de Aurora MySQL para que deje de ser una réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

> **Important**
>
> Para ejecutar este procedimiento, `autocommit` debe estar habilitado. Para habilitarlo, establezca el parámetro `autocommit` en 1. Para obtener información acerca de cómo modificar los parámetros, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Sintaxis

```
CALL mysql.rds_reset_external_master;
```

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_reset_external_master`. Este procedimiento se debe ejecutar en la instancia de base de datos de MySQL que se va a eliminar como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

> **Note**
>
> Ofrecemos estos procedimientos almacenados principalmente para habilitar la replicación con las instancias de MySQL que se ejecutan fuera de Amazon RDS. Recomendamos que utilice réplicas de Aurora para administrar la replicación dentro de un clúster de base de datos de Aurora MySQL siempre que sea posible. Para obtener información sobre la administración de la replicación en clústeres de base de datos de Aurora MySQL, consulte Uso de réplicas de Aurora.

Para obtener más información acerca del uso de la replicación para importar los datos desde una instancia de MySQL que se ejecuta fuera de Aurora MySQL, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

mysql.rds_reset_external_source (Aurora MySQL versión 3)

Vuelve a configurar una instancia de base de datos de Aurora MySQL para que deje de ser una réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

> **Important**
>
> Para ejecutar este procedimiento, `autocommit` debe estar habilitado. Para habilitarlo, establezca el parámetro `autocommit` en 1. Para obtener información acerca de cómo modificar los parámetros, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Sintaxis

```
CALL mysql.rds_reset_external_source;
```

Notas de uso

El usuario administrativo debe ejecutar el procedimiento `mysql.rds_reset_external_source`. Este procedimiento se debe ejecutar en la instancia de base de datos de MySQL que se va a eliminar como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

> **Note**
>
> Ofrecemos estos procedimientos almacenados principalmente para habilitar la replicación con las instancias de MySQL que se ejecutan fuera de Amazon RDS. Recomendamos que utilice réplicas de Aurora para administrar la replicación dentro de un clúster de base de datos de Aurora MySQL siempre que sea posible. Para obtener información sobre la administración de la replicación en clústeres de base de datos de Aurora MySQL, consulte Uso de réplicas de Aurora.

mysql.rds_set_binlog_source_ssl (Aurora MySQL versión 3)

Habilita el cifrado `SOURCE_SSL` para la replicación de binlog. Para obtener más información, consulte CHANGE REPLICATION SOURCE TO statement en la documentación de MySQL.

Sintaxis

```
CALL mysql.rds_set_binlog_source_ssl(mode);
```

Parámetros

mode

Valor que indica si está habilitado el cifrado:SOURCE_SSL

- 0: el cifrado SOURCE_SSL está deshabilitado. El valor predeterminado es 0.
- 1: el cifrado SOURCE_SSL está habilitado. Puede configurar el cifrado mediante SSL o TLS.

Notas de uso

Este procedimiento es compatible con la versión 3.06 y versiones posteriores de Aurora MySQL.

mysql.rds_set_external_master (Aurora MySQL versión 2)

Configura una instancia de base de datos de Aurora MySQL para que sea una réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

El procedimiento mysql.rds_set_external_master está en desuso y se eliminará en la próxima versión. En su lugar, use mysql.rds_set_external_source.

> **Important**
>
> Para ejecutar este procedimiento, autocommit debe estar habilitado. Para habilitarlo, establezca el parámetro autocommit en 1. Para obtener información acerca de cómo modificar los parámetros, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Sintaxis

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
```

```
);
```

Parámetros

host_name

El nombre de host o la dirección IP de la instancia de MySQL que se ejecuta fuera de Amazon RDS para convertirse en instancia de base de datos de origen.

host_port

El puerto usado por la instancia de MySQL que se ejecuta fuera de Amazon RDS que se configurará como instancia de base de datos de origen. Si la configuración de la red incluye la replicación de puertos SSH (Secure Shell) que convierte el número de puerto, especifique el número de puerto expuesto por SSH.

replication_user_name

El ID de un usuario con permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia de MySQL que se ejecuta fuera de Amazon RDS. Es recomendable que proporcione una cuenta que se use solo para la replicación con la instancia externa.

replication_user_password

La contraseña del ID de usuario especificado en `replication_user_name`.

mysql_binary_log_file_name

El nombre del registro binario de la instancia de base de datos de origen que contiene la información de replicación.

mysql_binary_log_file_location

La ubicación del registro binario `mysql_binary_log_file_name` en la que la replicación empieza a leer la información de la replicación.

Para determinar el nombre y la ubicación del archivo binlog, puede ejecutar `SHOW MASTER STATUS` en la instancia de base de datos de origen.

ssl_encryption

Valor que especifica si el cifrado de la capa de conexión segura (SSL) se usa en la conexión de reproducción. El 1 especifica que se usa el cifrado SSL; el 0 especifica que no se usa el cifrado. El valor predeterminado es 0.

> **Note**
>
> La opción `MASTER_SSL_VERIFY_SERVER_CERT` no es compatible. Esta opción se establece en 0, lo que significa que la conexión está cifrada, pero los certificados no se verifican.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_set_external_master`. Este procedimiento se debe ejecutar en la instancia de base de datos de MySQL que se va a configurar como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

Antes de ejecutar `mysql.rds_set_external_master`, debe configurar la instancia de MySQL que se ejecuta fuera de Amazon RDS como instancia de base de datos de origen. Para conectarse a la instancia de MySQL que se ejecuta fuera de Amazon RDS, debe especificar los valores de `replication_user_name` y `replication_user_password` que indican un usuario de replicación que tiene los permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia externa de MySQL.

Para configurar una instancia externa de MySQL como instancia de base de datos de origen

1. Con el cliente de MySQL que prefiera, conéctese a la instancia externa de MySQL y cree una cuenta de usuario que se usará para la replicación. A continuación se muestra un ejemplo.

   MySQL 5.7

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

   > **Note**
   >
   > Especifique una contraseña distinta de la que se muestra aquí como práctica recomendada de seguridad.

2. En la instancia externa de MySQL, conceda a `REPLICATION CLIENT` y a `REPLICATION SLAVE` privilegios para el usuario de replicación. En el siguiente ejemplo se conceden los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE` en todas las bases de datos al usuario “repl_user” de su dominio.

   MySQL 5.7

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Para utilizar la replicación cifrada, configure la instancia de base de datos de origen para que utilice conexiones SSL. Asimismo, importe el certificado de la entidad de certificación, el certificado cliente y la clave cliente en la instancia de base de datos o clúster de base de datos mediante el procedimiento mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Ofrecemos estos procedimientos almacenados principalmente para habilitar la replicación con las instancias de MySQL que se ejecutan fuera de Amazon RDS. Recomendamos que utilice réplicas de Aurora para administrar la replicación dentro de un clúster de base de datos de Aurora MySQL siempre que sea posible. Para obtener información sobre la administración de la replicación en clústeres de base de datos de Aurora MySQL, consulte Uso de réplicas de Aurora.

Después de llamar a `mysql.rds_set_external_master` para configurar una instancia de base de datos de Amazon RDS como réplica de lectura, puede llamar a mysql.rds_start_replication en la réplica de lectura para iniciar el proceso de replicación. Puede llamar a mysql.rds_reset_external_master (Aurora MySQL versión 2) para eliminar la configuración de la réplica de lectura.

Cuando se llama a `mysql.rds_set_external_master`, Amazon RDS registra la hora, el usuario y una acción `set master` en las tablas `mysql.rds_history` y `mysql.rds_replication_status`.

Ejemplos

Cuando se ejecuta en una instancia de base de datos de MySQL, el siguiente ejemplo configura la instancia de base de datos como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source (Aurora MySQL versión 3)

Configura una instancia de base de datos de Aurora MySQL para que sea una réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

> **Important**
>
> Para ejecutar este procedimiento, `autocommit` debe estar habilitado. Para habilitarlo, establezca el parámetro `autocommit` en 1. Para obtener información acerca de cómo modificar los parámetros, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Sintaxis

```
CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
```

```
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parámetros

host_name

El nombre de host o la dirección IP de la instancia de MySQL que se ejecuta fuera de Amazon RDS para convertirse en instancia de base de datos de origen.

host_port

El puerto usado por la instancia de MySQL que se ejecuta fuera de Amazon RDS que se configurará como instancia de base de datos de origen. Si la configuración de la red incluye la replicación de puertos SSH (Secure Shell) que convierte el número de puerto, especifique el número de puerto expuesto por SSH.

replication_user_name

El ID de un usuario con permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia de MySQL que se ejecuta fuera de Amazon RDS. Es recomendable que proporcione una cuenta que se use solo para la replicación con la instancia externa.

replication_user_password

La contraseña del ID de usuario especificado en `replication_user_name`.

mysql_binary_log_file_name

El nombre del registro binario de la instancia de base de datos de origen que contiene la información de replicación.

mysql_binary_log_file_location

La ubicación del registro binario `mysql_binary_log_file_name` en la que la replicación empieza a leer la información de la replicación.

Para determinar el nombre y la ubicación del archivo binlog, puede ejecutar `SHOW MASTER STATUS` en la instancia de base de datos de origen.

ssl_encryption

Valor que especifica si el cifrado de la capa de conexión segura (SSL) se usa en la conexión de reproducción. El 1 especifica que se usa el cifrado SSL; el 0 especifica que no se usa el cifrado. El valor predeterminado es 0.

> **Note**
>
> Debe haber importado un certificado SSL personalizado mediante mysql.rds_import_binlog_ssl_material para habilitar esta opción. Si no ha importado un certificado SSL personalizado, defina este parámetro en 0 y utilice mysql.rds_set_binlog_source_ssl (Aurora MySQL versión 3) para habilitar el SSL para la replicación de registros binarios.
> La opción SOURCE_SSL_VERIFY_SERVER_CERT no es compatible. Esta opción se establece en 0, lo que significa que la conexión está cifrada, pero los certificados no se verifican.

Notas de uso

El usuario administrativo debe ejecutar el procedimiento `mysql.rds_set_external_source`. Este procedimiento se debe ejecutar en la instancia de base de datos de Aurora MySQL que se va a configurar como réplica de lectura de una instancia de MySQL que se ejecute fuera de Amazon RDS.

Antes de ejecutar `mysql.rds_set_external_source`, debe configurar la instancia de MySQL que se ejecuta fuera de Amazon RDS como instancia de base de datos de origen. Para conectarse a la instancia de MySQL que se ejecuta fuera de Amazon RDS, debe especificar los valores de `replication_user_name` y `replication_user_password` que indican un usuario de replicación que tiene los permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia externa de MySQL.

Para configurar una instancia externa de MySQL como instancia de base de datos de origen

1. Con el cliente de MySQL que prefiera, conéctese a la instancia externa de MySQL y cree una cuenta de usuario que se usará para la replicación. A continuación se muestra un ejemplo.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> Especifique una contraseña distinta de la que se muestra aquí como práctica recomendada de seguridad.

2. En la instancia externa de MySQL, conceda a `REPLICATION CLIENT` y a `REPLICATION SLAVE` privilegios para el usuario de replicación. En el siguiente ejemplo se conceden los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE` en todas las bases de datos al usuario “repl_user” de su dominio.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

Para utilizar la replicación cifrada, configure la instancia de base de datos de origen para que utilice conexiones SSL. Asimismo, importe el certificado de la entidad de certificación, el certificado del cliente y la clave de cliente en la instancia de base de datos o clúster de base de datos mediante el procedimiento mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Ofrecemos estos procedimientos almacenados principalmente para habilitar la replicación con las instancias de MySQL que se ejecutan fuera de Amazon RDS. Recomendamos que utilice réplicas de Aurora para administrar la replicación dentro de un clúster de base de datos de Aurora MySQL siempre que sea posible. Para obtener información sobre la administración de la replicación en clústeres de base de datos de Aurora MySQL, consulte Uso de réplicas de Aurora.

Después de llamar a `mysql.rds_set_external_source` para configurar una instancia de base de datos de Aurora MySQL como réplica de lectura, puede llamar a mysql.rds_start_replication en la réplica de lectura para iniciar el proceso de replicación. Puede llamar a mysql.rds_reset_external_source (Aurora MySQL versión 3) para eliminar la configuración de la réplica de lectura.

Cuando se llama a `mysql.rds_set_external_source`, Amazon RDS registra la hora, el usuario y una acción `set master` en las tablas `mysql.rds_history` y `mysql.rds_replication_status`.

Ejemplos

Cuando se ejecuta en una instancia de base de datos de Aurora MySQL, el siguiente ejemplo configura la instancia de base de datos como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Amazon RDS.

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_master_with_auto_position (Aurora MySQL versión 2)

Permite configurar una instancia principal de Aurora MySQL para aceptar la replicación entrante desde una instancia MySQL externa. Este procedimiento también configura la replicación basada en identificadores de transacciones globales (GTID).

Este procedimiento no configura la replicación retardada, porque Aurora MySQL no admite este tipo de replicación.

Sintaxis

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parámetros

host_name

El nombre de host o la dirección IP de la instancia de MySQL que se ejecuta fuera de Aurora que se convertirá en el origen de replicación.

host_port

El puerto usado por la instancia de MySQL que se ejecuta fuera de Aurora que se configurará como origen de la replicación. Si la configuración de la red incluye la replicación de puertos Secure Shell (SSH) que convierte el número de puerto, especifique el número de puerto expuesto por SSH.

replication_user_name

ID de un usuario con permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia de MySQL que se ejecuta fuera de Aurora. Es recomendable que proporcione una cuenta que se use solo para la replicación con la instancia externa.

replication_user_password

La contraseña del ID de usuario especificado en `replication_user_name`.

ssl_encryption

Esta opción no está implementada actualmente. El valor predeterminado es 0.

Notas de uso

Para un clúster de bases de datos de Aurora MySQL, puede invocar este procedimiento almacenado cuando esté conectado a la instancia principal.

El usuario maestro debe ejecutar el procedimiento `mysql.rds_set_external_master_with_auto_position`. El usuario maestro ejecuta este procedimiento en la instancia principal de un clúster de bases de datos de Aurora MySQL que funciona como un destino de replicación. Este puede ser el destino de replicación de una instancia de MySQL externa o un clúster de bases de datos de Aurora MySQL.

Este procedimiento se admite para la versión 2 de Aurora MySQL. Para Aurora MySQL versión 3, utilice el procedimiento mysql.rds_set_external_source_with_auto_position (Aurora MySQL versión 3) en su lugar.

Antes de ejecutar `mysql.rds_set_external_master_with_auto_position`, configure la instancia de base de datos de MySQL para que sea un origen de replicación. Para conectar la instancia de MySQL externa, especifique valores para `replication_user_name` y `replication_user_password`. Estos valores deben indicar a un usuario de replicación que tiene permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia externa de MySQL.

Para configurar una instancia externa de MySQL como origen de replicación

1. Con el cliente de MySQL que prefiera, conéctese a la instancia externa de MySQL y cree una cuenta de usuario que se usará para la replicación. A continuación se muestra un ejemplo.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. En la instancia de MySQL externa, conceda a `REPLICATION CLIENT` y a `REPLICATION SLAVE` privilegios para el usuario de replicación. En el siguiente ejemplo se conceden los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE` en todas las bases de datos al usuario `'repl_user'` de su dominio.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

Cuando invoque `mysql.rds_set_external_master_with_auto_position`, Amazon RDS registra determinada información. Esta información es la hora, el usuario y una acción de `"set master"` en las tablas de `mysql.rds_history` y `mysql.rds_replication_status`.

Para omitir una transacción específica basada en GTID que se sabe que causa un problema, puede usar el procedimiento almacenado mysql.rds_skip_transaction_with_gtid (Aurora MySQL versión 2 y 3). Para obtener más información sobre el uso de la replicación basada en GTID, consulte Uso de la replicación basada en GTID.

Ejemplos

Cuando se ejecuta en una instancia principal de Aurora, el siguiente ejemplo configura el clúster de Aurora para que actúe como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Aurora.

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_external_source_with_auto_position (Aurora MySQL versión 3)

Permite configurar una instancia principal de Aurora MySQL para aceptar la replicación entrante desde una instancia MySQL externa. Este procedimiento también configura la replicación basada en identificadores de transacciones globales (GTID).

Sintaxis

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
```

```
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parámetros

host_name

El nombre de host o la dirección IP de la instancia de MySQL que se ejecuta fuera de Aurora que se convertirá en el origen de replicación.

host_port

El puerto usado por la instancia de MySQL que se ejecuta fuera de Aurora que se configurará como origen de la replicación. Si la configuración de la red incluye la replicación de puertos Secure Shell (SSH) que convierte el número de puerto, especifique el número de puerto expuesto por SSH.

replication_user_name

ID de un usuario con permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia de MySQL que se ejecuta fuera de Aurora. Es recomendable que proporcione una cuenta que se use solo para la replicación con la instancia externa.

replication_user_password

La contraseña del ID de usuario especificado en `replication_user_name`.

ssl_encryption

Esta opción no está implementada actualmente. El valor predeterminado es 0.

> **Note**
>
> Utilice mysql.rds_set_binlog_source_ssl (Aurora MySQL versión 3) para habilitar SSL para replicación de registros binarios.

Notas de uso

Para un clúster de bases de datos de Aurora MySQL, puede invocar este procedimiento almacenado cuando esté conectado a la instancia principal.

El usuario administrativo debe ejecutar el procedimiento `mysql.rds_set_external_source_with_auto_position`. El usuario administrativo ejecuta este procedimiento en la instancia principal de un clúster de bases de datos de Aurora MySQL que funciona como un destino de replicación. Este puede ser el destino de replicación de una instancia de MySQL externa o un clúster de bases de datos de Aurora MySQL.

Este procedimiento se admite para Aurora MySQL versión 3. Este procedimiento no configura la replicación retardada, porque Aurora MySQL no admite este tipo de replicación.

Antes de ejecutar `mysql.rds_set_external_source_with_auto_position`, configure la instancia de base de datos de MySQL para que sea un origen de replicación. Para conectar la instancia de MySQL externa, especifique valores para `replication_user_name` y `replication_user_password`. Estos valores deben indicar a un usuario de replicación que tiene permisos `REPLICATION CLIENT` y `REPLICATION SLAVE` en la instancia externa de MySQL.

Para configurar una instancia externa de MySQL como origen de replicación

1. Con el cliente de MySQL que prefiera, conéctese a la instancia externa de MySQL y cree una cuenta de usuario que se usará para la replicación. A continuación se muestra un ejemplo.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. En la instancia de MySQL externa, conceda a `REPLICATION CLIENT` y a `REPLICATION SLAVE` privilegios para el usuario de replicación. En el siguiente ejemplo se conceden los privilegios `REPLICATION CLIENT` y `REPLICATION SLAVE` en todas las bases de datos al usuario `'repl_user'` de su dominio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Cuando invoque `mysql.rds_set_external_source_with_auto_position`, Amazon RDS registra determinada información. Esta información es la hora, el usuario y una acción de `"set master"` en las tablas de `mysql.rds_history` y `mysql.rds_replication_status`.

Para omitir una transacción específica basada en GTID que se sabe que causa un problema, puede usar el procedimiento almacenado mysql.rds_skip_transaction_with_gtid (Aurora MySQL versión 2 y 3). Para obtener más información sobre el uso de la replicación basada en GTID, consulte Uso de la replicación basada en GTID.

Ejemplos

Cuando se ejecuta en una instancia principal de Aurora, el siguiente ejemplo configura el clúster de Aurora para que actúe como réplica de lectura de una instancia de MySQL que se ejecuta fuera de Aurora.

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_master_auto_position (Aurora MySQL versión 2)

Establece el modo de replicación en el que se debe basar ya sea en posiciones de archivos de registros binarios o en identificadores de transacciones globales (GTID).

Sintaxis

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

Parámetros

auto_position_mode

Un valor que indicar si debe usarse la replicación de posición de los archivos de registro o la replicación basada en GTID:

- `0`: utilice el método de replicación basado en la posición del archivo de registro binario. El valor predeterminado es `0`.
- 1: utilice el método de replicación basado en GTID.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_set_master_auto_position`.

Este procedimiento se admite para la versión 2 de Aurora MySQL.

mysql.rds_set_read_only (Aurora MySQL versión 3)

Activa o desactiva el modo `read_only` globalmente para la instancia de base de datos.

Sintaxis

```
CALL mysql.rds_set_read_only(mode);
```

Parámetros

mode

Valor que indica si el modo `read_only` está activado o desactivado globalmente para la instancia de base de datos:

- `0`: `OFF`. El valor predeterminado es `0`.
- `1` – `ON`

Notas de uso

El procedimiento almacenado `mysql.rds_set_read_only` modifica solo el parámetro `read_only`. El parámetro `innodb_read_only` no se puede cambiar en las instancias de bases de datos del lector.

El cambio del parámetro `read_only` no persiste al reiniciarse. Para realizar cambios permanentes en `read_only`, debe usar el parámetro de clúster de base de datos `read_only`.

Este procedimiento es compatible con la versión 3.06 y versiones posteriores de Aurora MySQL.

mysql.rds_set_session_binlog_format (Aurora MySQL versión 2)

Establece el formato de registro binario de la sesión actual.

Sintaxis

```
CALL mysql.rds_set_session_binlog_format(format);
```

Parámetros

format

Un valor que indica el formato de registro binario de la sesión actual:

- `STATEMENT`: el origen de la replicación escribe los eventos en el registro binario en función de instrucciones SQL.
- `ROW`: el origen de la replicación escribe los eventos en el registro binario que indican los cambios en las filas individuales de la tabla.
- `MIXED`: el registro se basa generalmente en instrucciones SQL, pero cambia a filas en determinadas condiciones. Para obtener más información, consulte Mixed Binary Logging Format en la documentación de MySQL.

Notas de uso

Para un clúster de bases de datos de Aurora MySQL, puede invocar este procedimiento almacenado cuando esté conectado a la instancia principal.

Para utilizar este procedimiento almacenado, debe tener configurado el registro binario para la sesión actual.

Para Aurora, este procedimiento se admite en la versión 2.12 y versiones posteriores compatibles con MySQL 5.7 de Aurora MySQL.

mysql.rds_set_source_auto_position (Aurora MySQL versión 3)

Establece el modo de replicación en el que se debe basar, ya sea en posiciones de archivos de registros binarios o en identificadores de transacciones globales (GTID).

Sintaxis

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

Parámetros

auto_position_mode

Un valor que indicar si debe usarse la replicación de posición de los archivos de registro o la replicación basada en GTID:

- `0`: utilice el método de replicación basado en la posición del archivo de registro binario. El valor predeterminado es `0`.
- 1: utilice el método de replicación basado en GTID.

Notas de uso

Para un clúster de bases de datos de Aurora MySQL, puede invocar este procedimiento almacenado cuando esté conectado a la instancia principal.

El usuario administrativo debe ejecutar el procedimiento `mysql.rds_set_source_auto_position`.

mysql.rds_skip_repl_error

Omite y elimina un error de replicación en una réplica de lectura de base de datos de MySQL.

Sintaxis

```
CALL mysql.rds_skip_repl_error;
```

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_skip_repl_error` en una réplica de lectura. Para obtener más información sobre este procedimiento, consulte Skipping the current replication error (Omitir el error de replicación actual).

Para determinar si hay errores, ejecute el comando `SHOW REPLICA STATUS\G` de MySQL. Si un error de replicación no es crítico, puede ejecutar `mysql.rds_skip_repl_error` para omitir el error. Si hay varios errores, `mysql.rds_skip_repl_error` elimina el primer error y advierte de que hay otros presentes. A continuación, puede usar `SHOW REPLICA STATUS\G` para determinar la acción correcta para el siguiente error. Para obtener información acerca de los valores devueltos, consulte la instrucción SHOW REPLICA STATUS en la documentación de MySQL.

Para obtener más información acerca de la resolución de problemas de replicación con Aurora MySQL, consulte Diagnóstico y solución de un error de replicación de lectura de MySQL.

Error de replicación detenida

Al llamar al procedimiento `mysql.rds_skip_repl_error`, es posible que reciba un mensaje de error en el que se indica que la réplica tiene un error o está deshabilitada.

Este mensaje de error aparece si ejecuta el procedimiento en la instancia principal en lugar de en la réplica de lectura. Debe ejecutar este procedimiento en la réplica de lectura para que funcione.

Este mensaje de error también puede aparecer si ejecuta el procedimiento en la réplica de lectura, pero la replicación no se puede reiniciar correctamente.

Si tiene que omitir un número de errores elevado, el retardo de réplica puede aumentar por encima del periodo de retención predeterminado para los archivos de log binarios (binlog). En este caso, puede producirse un error fatal porque los archivos binlog se están limpiando antes de reproducirse de nuevo en la réplica de lectura. Esta limpieza hace que la replicación se detenga y ya no se puede llamar al comando `mysql.rds_skip_repl_error` para omitir los errores de replicación.

Puede mitigar este problema incrementando el número de horas que los archivos binlog se retienen en la instancia de base de datos de origen. Después de incrementar el tiempo de retención de los archivos binlog, puede reiniciar la replicación y llamar al comando `mysql.rds_skip_repl_error` si es necesario.

Para definir el tiempo de retención de binlog, use el procedimiento mysql.rds_set_configuration y especifique un parámetro de configuración de `'binlog retention hours'` junto con el número de horas para retener los archivos binlog en el clúster de base de datos. El ejemplo siguiente define el período de retención de los archivos binlog en 48 horas.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

Inicia la replicación desde un clúster de base de datos de Aurora MySQL.

> **Note**
>
> Puede utilizar el procedimiento almacenado mysql.rds_start_replication_until (versión 3 de Aurora MySQL) o mysql.rds_start_replication_until_gtid (Aurora MySQL versión 3) para iniciar la replicación desde una instancia de base de datos de Aurora MySQL y detener la replicación en la ubicación del archivo de registro binario especificado.

Sintaxis

```
CALL mysql.rds_start_replication;
```

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_start_replication`.

Para importar datos desde una instancia de MySQL fuera de Amazon RDS, llame a `mysql.rds_start_replication` en la réplica de lectura para iniciar el proceso de replicación después de llamar a mysql.rds_set_external_master (Aurora MySQL versión 2) o mysql.rds_set_external_source (Aurora MySQL versión 3) para crear la configuración de replicación. Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Para exportar datos a una instancia de MySQL externa a Amazon RDS, llame a `mysql.rds_start_replication` y a `mysql.rds_stop_replication` en la réplica de lectura para controlar algunas acciones de replicación, como la purga de registros binarios. Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Puede llamar a `mysql.rds_start_replication` en la réplica de lectura para reiniciar cualquier proceso de replicación que haya detenido previamente llamando a `mysql.rds_stop_replication`. Para obtener más información, consulte Error de replicación detenida.

mysql.rds_start_replication_until (versión 3 de Aurora MySQL)

Inicia la replicación desde un clúster de base de datos de Aurora MySQL y detiene la replicación en la ubicación del archivo de registro binario especificado.

Sintaxis

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

Parámetros

replication_log_file

El nombre del registro binario de la instancia de base de datos de origen que contiene la información de replicación.

replication_stop_point

La ubicación del registro binario `replication_log_file` en la que la replicación se detendrá.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_start_replication_until`.

Este procedimiento es compatible con la versión 3.04 y versiones posteriores de Aurora MySQL.

El procedimiento almacenado `mysql.rds_start_replication_until` no es compatible con la replicación administrada, que incluye lo siguiente:

- Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS
- Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora

El nombre de archivo especificado para el parámetro `replication_log_file` debe coincidir con el nombre del archivo binlog de instancia de base de datos de origen.

Cuando el parámetro `replication_stop_point` especifica una ubicación de parada correspondiente al pasado, la replicación se detiene de inmediato.

Ejemplos

En el ejemplo siguiente se inicia la replicación y se replican los cambios hasta que alcanza la ubicación `120` del archivo registro binario `mysql-bin-changelog.000777`.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_stop_replication

Detiene la replicación desde una instancia de base de datos de MySQL.

Sintaxis

```
CALL mysql.rds_stop_replication;
```

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_stop_replication`.

Si desea configurar la replicación para importar datos desde una instancia de MySQL que se ejecuta fuera de Amazon RDS, llame a `mysql.rds_stop_replication` en la réplica de lectura para detener el proceso de replicación una vez completada la importación. Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

Si desea configurar la replicación para exportar datos a una instancia de MySQL externa a Amazon RDS, llame a `mysql.rds_start_replication` y a `mysql.rds_stop_replication` en la réplica de lectura para controlar algunas acciones de replicación, como la limpieza de registros binarios. Para obtener más información, consulte Replicación entre Aurora y MySQL o entre Aurora y otro clúster de base de datos de Aurora (replicación de registro binario).

El procedimiento almacenado `mysql.rds_stop_replication` no es compatible con la replicación administrada, que incluye lo siguiente:

- Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS
- Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora

Finalización de una sesión o una consulta

Los siguientes procedimientos almacenados finalizan una sesión o una consulta.

Temas

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

Finaliza una conexión al servidor de MySQL.

Sintaxis

```
CALL mysql.rds_kill(processID);
```

Parámetros

processID

La identidad del subproceso de conexión que se va a finalizar.

Notas de uso

Cada conexión al servidor de MySQL se ejecuta en un subproceso independiente. Para finalizar una conexión, use el procedimiento `mysql.rds_kill` y transfiera el ID de subproceso de esa conexión. Para obtener el ID del subproceso, use el comando SHOW PROCESSLIST de MySQL.

Ejemplos

El siguiente ejemplo finaliza una conexión con el ID de subproceso 4243:

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

Finaliza una consulta que se ejecuta en el servidor de MySQL.

Sintaxis

```
CALL mysql.rds_kill_query(processID);
```

Parámetros

processID

La identidad del proceso o subproceso que ejecuta la consulta que se va a finalizar.

Notas de uso

Para detener una consulta que se ejecuta en el servidor MySQL, utilice el procedimiento `mysql_rds_kill_query` y pase el ID de conexión del subproceso que está ejecutando la consulta. A continuación, el procedimiento finaliza la conexión.

Para obtener el ID, consulte la tabla INFORMATION_SCHEMA PROCESSLIST o utilice el comando MySQL SHOW PROCESSLIST. El valor de la columna ID de `SHOW PROCESSLIST` o `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST` es el *ID del proceso*.

Ejemplos

El siguiente ejemplo detiene una consulta con el ID de subproceso 230040:

```
CALL mysql.rds_kill_query(230040);
```

Replicación de transacciones mediante GTID

Los siguientes procedimientos almacenados controlan cómo se replican las transacciones mediante identificadores de transacciones globales (GTID) con Aurora MySQL. Para conocer cómo utilizar la replicación basada en GTID con Aurora MySQL, consulte Uso de la replicación basada en GTID.

Temas

- mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versión 3)
- mysql.rds_gtid_purged (Aurora MySQL versión 3)
- mysql.rds_skip_transaction_with_gtid (Aurora MySQL versión 2 y 3)
- mysql.rds_start_replication_until_gtid (Aurora MySQL versión 3)

mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versión 3)

Configura la opción `ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS` de la instrucción `CHANGE REPLICATION SOURCE TO`. Hace que el canal de replicación asigne un GTID a las transacciones replicadas que no tienen uno. De esta forma, puede realizar la replicación de registros binarios desde un origen que no utiliza la replicación basada en GTID en una réplica que sí. Para obtener más información, consulte la instrucción CHANGE REPLICATION SOURCE TO y Replicación desde un origen sin GTID a una réplica con GTID en el Manual de referencia de MySQL.

Sintaxis

```
CALL mysql.rds_assign_gtids_to_anonymous_transactions(gtid_option);
```

Parámetros

`gtid_option`

Valor de cadena. Los valores permitidos son `OFF`, `LOCAL` o un UUID especificado.

Notas de uso

Este procedimiento tiene el mismo efecto que la emisión de la instrucción `CHANGE REPLICATION SOURCE TO ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS =` *`gtid_option`* en la comunidad MySQL.

El GTID debe dirigirse a `ON` para *`gtid_option`* que se configurará en `LOCAL` o un UUID específico.

El valor predeterminado es `OFF`, lo que significa que la característica no se utiliza.

`LOCAL` asigna un GTID que incluye el UUID de la réplica (la configuración `server_uuid`).

Al pasar un parámetro que es UUID se asigna un GTID que incluye el UUID especificado, como la configuración `server_uuid` para el servidor de origen de replicación.

Ejemplos

Para desactivar esta característica:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('OFF');
+------------------------------------------------------------+
| Message  |
+------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: OFF |
+------------------------------------------------------------+
1 row in set (0.07 sec)
```

Para utilizar el UUID de la réplica:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('LOCAL');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: LOCAL |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Para utilizar un UUID especificado:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('317a4760-
f3dd-3b74-8e45-0615ed29de0e');
+-----------------------------------------------------------------------------------------------------
+
| Message |
+-----------------------------------------------------------------------------------------------------
+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: 317a4760-
f3dd-3b74-8e45-0615ed29de0e |
+-----------------------------------------------------------------------------------------------------
+
```

```
1 row in set (0.07 sec)
```

mysql.rds_gtid_purged (Aurora MySQL versión 3)

Establece el valor global de la variable de sistema `gtid_purged` a un conjunto de identificadores de transacciones globales (GTID) determinado. La variable de sistema `gtid_purged` es un conjunto de GTID que consta de los GTID de todas las transacciones que se han confirmado en el servidor, pero que no existen en ningún archivo de registro binario del servidor.

Hay dos maneras de establecer el valor de `gtid_purged` para que sea compatible con MySQL 8.0:

- Reemplace el valor de `gtid_purged` con el conjunto de GTID especificado.
- Anexe el conjunto de GTID especificado al conjunto de GTID que ya contiene `gtid_purged`.

Sintaxis

Para reemplazar el valor de `gtid_purged` con el conjunto de GTID especificado:

```
CALL mysql.rds_gtid_purged (gtid_set);
```

Para anexar el valor de `gtid_purged` al conjunto de GTID especificado:

```
CALL mysql.rds_gtid_purged (+gtid_set);
```

Parámetros

`gtid_set`

El valor de *`gtid_set`* debe ser un superconjunto del valor actual de `gtid_purged`, y no puede cruzarse con `gtid_subtract(gtid_executed,gtid_purged)`. Es decir, el nuevo conjunto de GTID debe incluir todos los GTID que ya estuvieron en `gtid_purged`, y no puede incluir ningún GTID en `gtid_executed` que aún no se haya purgado. El parámetro *`gtid_set`* tampoco puede incluir ningún GTID que esté en el conjunto `gtid_owned` global, los GTID de las transacciones que se están procesando actualmente en el servidor.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_gtid_purged`.

Este procedimiento es compatible con la versión 3.04 y versiones posteriores de Aurora MySQL.

Ejemplos

En el siguiente ejemplo, se asigna el GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23` a la variable global `gtid_purged`.

```
CALL mysql.rds_gtid_purged('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_skip_transaction_with_gtid (Aurora MySQL versión 2 y 3)

Omite la replicación de una transacción con el identificador de transacción global (GTID) especificado en una instancia principal de Aurora.

Puede utilizar este procedimiento para la recuperación ante desastres cuando se sabe que una transacción de GTID específica causa un problema. Utilice este procedimiento almacenado para omitir la transacción problemática. Entre los ejemplos de transacciones problemáticas se incluyen transacciones que inhabilitan la replicación, eliminan datos importantes o provocan que la instancia de base de datos no esté disponible.

Sintaxis

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

Parámetros

`gtid_to_skip`

El GTID de la transacción de replicación que se debe omitir.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_skip_transaction_with_gtid`.

Este procedimiento se admite para Aurora MySQL versión 2 y 3.

Ejemplos

En el ejemplo siguiente se omite la replicación de la transacción con el GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid (Aurora MySQL versión 3)

Inicia la replicación desde un clúster de base de datos de Aurora MySQL y detiene la replicación inmediatamente después del identificador de transacción global (GTID) especificado.

Sintaxis

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

Parámetros

gtid

El GTID después del cual debe detenerse la replicación.

Notas de uso

El usuario maestro debe ejecutar el procedimiento `mysql.rds_start_replication_until_gtid`.

Este procedimiento es compatible con la versión 3.04 y versiones posteriores de Aurora MySQL.

El procedimiento almacenado `mysql.rds_start_replication_until_gtid` no es compatible con la replicación administrada, que incluye lo siguiente:

- Reproducción de clústeres de base de datos de Amazon Aurora MySQL entre Regiones de AWS
- Migración de datos desde una instancia de base de datos de RDS para MySQL a un clúster de base de datos de Amazon Aurora MySQL con una réplica de lectura de Aurora

Cuando el parámetro `gtid` especifica una transacción que ya ha ejecutado la réplica, la replicación se detiene de inmediato.

Ejemplos

En el ejemplo siguiente se inicia la replicación y se replican los cambios hasta que alcanza el GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

Rotación de los registros de consultas

Los siguientes procedimientos almacenados rotan los registros de MySQL en tablas de copia de seguridad. Para obtener más información, consulte Archivos de registro de base de datos de Aurora MySQL.

Temas

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

Rota la tabla `mysql.general_log` a una tabla de copia de seguridad.

Sintaxis

```
CALL mysql.rds_rotate_general_log;
```

Notas de uso

Para rotar la tabla `mysql.general_log` a una tabla de copia de seguridad, llame al procedimiento `mysql.rds_rotate_general_log`. Cuando se rotan las tablas de registro, la tabla de registro actual se copia en una tabla de registro de copia de seguridad y las entradas de la tabla de registro actual se eliminan. Si ya existe una tabla de registro de copia de seguridad, se elimina antes de copiar la tabla del registro actual en el copia de seguridad. Puede consultar la tabla de registro de copia de seguridad si es necesaria. La tabla de registro de copia de seguridad para la tabla `mysql.general_log` se llama `mysql.general_log_backup`.

Puede ejecutar este procedimiento solo cuando el parámetro `log_output` se establezca en`TABLE`.

mysql.rds_rotate_slow_log

Rota la tabla `mysql.slow_log` a una tabla de copia de seguridad.

Sintaxis

```
CALL mysql.rds_rotate_slow_log;
```

Notas de uso

Para rotar la tabla `mysql.slow_log` a una tabla de copia de seguridad, llame al procedimiento `mysql.rds_rotate_slow_log`. Cuando se rotan las tablas de registro, la tabla de registro actual se copia en una tabla de registro de copia de seguridad y las entradas de la tabla de registro actual se eliminan. Si ya existe una tabla de registro de copia de seguridad, se elimina antes de copiar la tabla del registro actual en el copia de seguridad.

Puede consultar la tabla de registro de copia de seguridad si es necesaria. La tabla de registro de copia de seguridad para la tabla `mysql.slow_log` se llama `mysql.slow_log_backup`.

Establecimiento y muestra de la configuración del registro binario

Los siguientes procedimientos almacenados establecen y muestran los parámetros de configuración, por ejemplo, para la retención de archivos de registro binario.

Temas

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

Especifica el número de horas que se deben conservar los registros binarios o el número de segundos que se retrasará la replicación.

Sintaxis

```
CALL mysql.rds_set_configuration(name,value);
```

Parámetros

name

El nombre del parámetro de configuración que se va a definir.

value

El valor del parámetro de configuración.

Notas de uso

El procedimiento `mysql.rds_set_configuration` admite los parámetros de configuración siguientes:

- binlog retention hours

Los parámetros de configuración se almacenan de forma permanente y sobreviven a cualquier reinicio o conmutación por error de una instancia de base de datos.

binlog retention hours

El parámetro `binlog retention hours` se usa para especificar la cantidad de horas que se deben retener los archivos de registro binario. Por lo general, Amazon Aurora limpia un registro binario lo antes posible, pero el registro binario podría seguir siendo necesario para la replicación con una base de datos MySQL externa a Aurora.

El valor predeterminado de `binlog retention hours` es `NULL`. Para Aurora MySQL, `NULL` significa que los registros binarios se limpian en diferido. Los registros binarios de Aurora MySQL pueden permanecer en el sistema durante un cierto periodo, que generalmente no es más de un día.

Para especificar el número de horas que se deben retener los registros binarios en un clúster de base de datos, utilice el procedimiento almacenado `mysql.rds_set_configuration` y especifique un periodo lo bastante largo como para realizar la replicación, como se muestra en el siguiente ejemplo.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> No puede utilizar el valor `0` para `binlog retention hours`.

El valor máximo de `binlog retention hours` para los clústeres de bases de datos de las versiones 2.11.0 y superiores y 3 de Aurora MySQL es 2160 (90 días).

Una vez que haya definido el periodo de retención, monitorice el uso del almacenamiento para la instancia de base de datos con el fin de asegurarse de que los logs binarios conservados no consuman demasiado almacenamiento.

mysql.rds_show_configuration

El número de horas que se conservan los registros binarios.

Sintaxis

```
CALL mysql.rds_show_configuration;
```

Notas de uso

Para verificar el número de horas que Amazon RDS conserva los registros binarios, use el procedimiento almacenado `mysql.rds_show_configuration`.

Ejemplos

El ejemplo siguiente muestra el periodo de retención:

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

Aurora MySQL: tablas de information_schema específica

Aurora MySQL tiene ciertas tablas `information_schema` que son específicas de Aurora.

information_schema.aurora_global_db_instance_status

La tabla `information_schema.aurora_global_db_instance_status` contiene información sobre el estado de todas las instancias de base de datos en los clústeres de base de datos principales y secundarios de una base de datos global. La siguiente tabla muestra las columnas que puede utilizar. Las columnas restantes son solo para uso interno de Aurora.

> **Note**
>
> Esta tabla de esquema de información solo está disponible con bases de datos globales de la versión 3.04.0 de Aurora MySQL y versiones posteriores.

Columna	Tipo de datos	Descripción
SERVER_ID	varchar(100)	El identificador de la instancia de base de datos.
SESSION_ID	varchar(100)	Un identificador único de la sesión actual. Un valor `MASTER_SESSION_ID`

Columna	Tipo de datos	Descripción
		identifica la instancia de base de datos de Writer (principal).
AWS_REGION	varchar(100)	La Región de AWS en la que se ejecuta esta instancia de base de datos global. Para obtener una lista de regiones, consulte Disponibilidad por región.
DURABLE_LSN	bigint unsigned	El número de secuencia de registro (LSN) hecho duradero en el almacenamiento. Un número de secuencia de registro (LSN) es un número secuencial único que identific a un registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción posterior.
HIGHEST_LSN_RCVD	bigint unsigned	El LSN más alto recibido por la instancia de base de datos de la instancia de base de datos del escritor.
OLDEST_READ_VIEW_TRX_ID	bigint unsigned	El ID de la transacción más antigua a la que puede purgar la instancia de base de datos del escritor.

Columna	Tipo de datos	Descripción
OLDEST_READ_VIEW_LSN	bigint unsigned	El LSN más antiguo utilizado por la instancia de base de datos para leer desde el almacenamiento.
VISIBILITY_LAG_IN_MSEC	float(10,0) unsigned	Para los lectores del clúster de base de datos principal , cuánto se está retrasando esta instancia de base de datos con respecto a la instancia de base de datos del escritor en milisegundos. En el caso de los lectores de un clúster de base de datos secundario, cuánto se está retrasando esta instancia de base de datos respecto al volumen secundario en milisegundos.

information_schema.aurora_global_db_status

La tabla `information_schema.aurora_global_db_status` contiene información sobre varios aspectos del retraso de la base de datos global de Aurora, específicamente, el retraso del almacenamiento de Aurora subyacente (llamado retraso en la durabilidad) y el retraso entre el objetivo de punto de recuperación (RPO). La siguiente tabla muestra las columnas que puede utilizar. Las columnas restantes son solo para uso interno de Aurora.

> **Note**
>
> Esta tabla de esquema de información solo está disponible con bases de datos globales de la versión 3.04.0 de Aurora MySQL y versiones posteriores.

Columna	Tipo de datos	Descripción
AWS_REGION	varchar(100)	La Región de AWS en la que se ejecuta esta instancia de base de datos global. Para obtener una lista de regiones, consulte Disponibilidad por región.
HIGHEST_LSN_WRITTEN	bigint unsigned	El número de secuencia de registro (LSN) más alto que existe actualmente en este clúster de base de datos. Un número de secuencia de registro (LSN) es un número secuencial único que identific a un registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción posterior.
DURABILITY_LAG_IN_ MILLISECONDS	float(10,0) unsigned	La diferencia en los valores de marca temporal entre el `HIGHEST_LSN_WRITTE N` del clúster de base de datos secundario y el `HIGHEST_LSN_WRITTE N` del clúster de base de datos principal. Este valor es siempre 0 en el clúster de base de datos principal de la base de datos global de Aurora.

Columna	Tipo de datos	Descripción
RPO_LAG_IN_MILLISECONDS	float(10,0) unsigned	El retraso del objetivo de punto de recuperación (RPO). El retardo de RPO es el tiempo que tarda la transacción de usuario más reciente en almacenarse en un clúster de base de datos secundario después de almacenarse en el clúster de base de datos principal de una base de datos global de Aurora. Este valor es siempre 0 en el clúster de base de datos principal de la base de datos global de Aurora. En términos sencillos, esta métrica calcula el objetivo de punto de recuperación de cada clúster de base de datos de Aurora MySQL de una base de datos global de Aurora, es decir, cuántos datos podrían perderse si se produce una interrupción. Al igual que con el retraso, el RPO se mide en tiempo.

Columna	Tipo de datos	Descripción
LAST_LAG_CALCULATION_TIMESTAMP	datetime	La marca temporal que especifica cuándo se calcularon por última vez los valores para `DURABILITY_LAG_IN_MILLISECONDS` y `RPO_LAG_IN_MILLISECONDS`. Un valor temporal como `1970-01-01 00:00:00+00` significa que este es el clúster de base de datos principal.
OLDEST_READ_VIEW_TRX_ID	bigint unsigned	El ID de la transacción más antigua a la que puede purgar la instancia de base de datos del escritor.

information_schema.replica_host_status

La tabla `information_schema.replica_host_status` contiene información de replicación. Las columnas que puede utilizar se muestran en la tabla a continuación. Las columnas restantes son solo para uso interno de Aurora.

Columna	Tipo de datos	Descripción
CPU	double	El porcentaje de uso de la CPU del host de la réplica.
IS_CURRENT	tinyint	Si la réplica está actualizada.
LAST_UPDATE_TIMESTAMP	datetime(6)	Hora en la que se produjo la última actualización. Se usa para determinar si un registro está obsoleto.

Columna	Tipo de datos	Descripción
REPLICA_LAG_IN_MILLISECONDS	double	El retraso de réplica en milisegundos.
SERVER_ID	varchar(100)	El ID del servidor de base de datos.
SESSION_ID	varchar(100)	El ID de la sesión de la base de datos. Se utiliza para determinar si una instancia de base de datos es una instancia de escritor o de lectura.

> **Note**
>
> Cuando una instancia de réplica se retrasa, la información consultada en su tabla `information_schema.replica_host_status` puede estar desactualizada. En este caso, te recomendamos que consultes desde la instancia del escritor.
> Si bien la `mysql.ro_replica_status` tabla contiene información similar, no es recomendable utilizarla.

information_schema.aurora_forwarding_processlist

La tabla `information_schema.aurora_forwarding_processlist` contiene información sobre los procesos involucrados en el reenvío de escritura.

El contenido de esta tabla solo está visible en la instancia de base de datos del escritor de un clúster de base de datos que tiene activado el reenvío de escritura global o en el clúster. Se devuelve un conjunto de resultados vacío en las instancias de base de datos del lector.

Campo	Tipo de datos	Descripción
ID	bigint	El identificador de la conexión en la instancia de base de datos del escritor. Este identificador es el mismo valor que se muestra en la columna `Id` de la instrucción `SHOW PROCESSLIST` y que es devuelto por la función `CONNECTION_ID()` dentro del subproceso.
USER	varchar (32)	El usuario de MySQL que emitió la instrucción.
HOST	varchar (255)	El cliente MySQL que emitió la instrucción. En el caso de instrucciones reenviadas, este campo muestra la dirección host del cliente de la aplicación que estableció la conexión en la instancia de base de datos del lector de reenvío.
DB	varchar (64)	La base de datos predeterminada para el subproceso.
COMMAND	varchar (16)	El tipo de comando que el subproceso ejecuta en nombre del cliente, o `Sleep` si la sesión está inactiva. Para obtener descripciones de los comandos de los subprocesos, consulte Thread Command Values en la documentación de MySQL.
HORA	int	El tiempo en segundos que el subproceso ha estado en su estado actual.
STATE	varchar (64)	Una acción, evento o estado que indica lo que está haciendo el subproceso. Para obtener descripciones de los valores de estado, consulte General Thread States en la documentación de MySQL.
INFO	longtext	La instrucción que se está ejecutando el subproceso, o `NULL` si no está ejecutando una instrucción. La instrucción puede ser la que se envía al servidor o una instrucción más interna si ejecuta otras instrucciones.

Campo	Tipo de datos	Descripción
IS_FORWARDED	bigint	Indica si el subproceso se reenvía desde una instancia de base de datos del lector.
REPLICA_SESSION_ID	bigint	El identificador de conexión de la réplica de Aurora. Este identificador es el mismo valor que se muestra en la columna `Id` de la instrucción `SHOW PROCESSLIST` en la instancia de base de datos de Aurora Reader de reenvío.
REPLICA_INSTANCE_IDENTIFIER	varchar (64)	El identificador de la instancia de base de datos del subproceso de reenvío.
REPLICA_clúster_NAME	varchar (64)	El identificador del clúster de base de datos del subproceso de reenvío. Para el reenvío de escritura en el clúster, este identificador es el mismo clúster de base de datos que la instancia de base de datos del escritor.
REPLICA_REGION	varchar (64)	La Región de AWS desde donde se origina el subproceso de reenvío. Para el reenvío de escritura en el clúster, esta región es la misma Región de AWS que la instancia de base de datos del escritor.

Actualizaciones del motor de base de datos de Amazon Aurora MySQL

Amazon Aurora publica de forma periódica actualizaciones. Las actualizaciones se aplican a clústeres de base de datos Aurora durante los períodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende de la región y de la configuración del período de mantenimiento para el clúster de la base de datos, así como del tipo de actualización.

Las versiones de Amazon Aurora están disponibles para todas las regiones de AWS en el transcurso de varios días. Algunas regiones pueden mostrar temporalmente una versión de motor que aún no está disponible en otra región.

Las actualizaciones se aplican a todas las instancias en un clúster de bases de datos a la vez. Una actualización requiere un reinicio de la base de datos en todas las instancias en un clúster de bases de datos, por lo que se producirán entre 20 y 30 segundos de inactividad. Puede ver o cambiar la configuración del periodo de mantenimiento desde la Consola de administración de AWS.

Para obtener más información sobre las versiones de Aurora MySQL compatibles con Amazon Aurora, consulte las notas de la versión de Aurora MySQL.

A continuación, puede aprender a elegir la versión de Aurora MySQL correcta para el clúster, cómo especificar la versión al crear o actualizar un clúster y los procedimientos para actualizar un clúster de una versión a otra con una interrupción mínima.

Temas

- Comprobación de los números de versión de Aurora MySQL
- Versiones beta y de soporte a largo plazo (LTS) para Amazon Aurora MySQL
- Preparación para el final del soporte estándar de la versión 2 de Amazon Aurora MySQL-Compatible Edition
- Preparación para el final de la vida útil de la versión 1 de la Edición compatible con MySQL de Amazon Aurora
- Actualización de clústeres de base Amazon Aurora MySQL
- Actualizaciones y correcciones de motor de base de datos de Amazon Aurora MySQL

Comprobación de los números de versión de Aurora MySQL

Aunque Edición compatible con Aurora MySQL es compatible con los motores de base de datos MySQL, Aurora MySQL incluye características y correcciones de errores específicos para versiones de Aurora MySQL particulares. Los desarrolladores de aplicaciones pueden comprobar la versión de Aurora MySQL en sus aplicaciones mediante SQL. Los administradores de bases de datos pueden comprobar y especificar las versiones de Aurora MySQL al crear o actualizar clústeres de base de datos de Aurora MySQL e instancias de base de datos.

Temas

- Comprobación o especificación de versiones del motor de Aurora MySQL mediante AWS
- Comprobación de versiones de Aurora MySQL mediante SQL

Comprobación o especificación de versiones del motor de Aurora MySQL mediante AWS

Cuando se realizan tareas administrativas mediante la Consola de administración de AWS, la AWS CLI o la API de RDS, se especifica la versión de Aurora MySQL en un formato alfanumérico descriptivo.

A partir de la versión 2 de Aurora MySQL, las versiones del motor de Aurora tienen la siguiente sintaxis.

```
mysql-major-version.mysql_aurora.aurora-mysql-version
```

La porción *mysql-major-version-* es `5.7` o `8.0`. Este valor representa la versión del protocolo cliente y el nivel general de soporte de características MySQL para la versión de Aurora MySQL correspondiente.

El *aurora-mysql-version* es un valor punteado con tres partes: la versión principal de Aurora MySQL, la versión secundaria de Aurora MySQL y el nivel de parche. La versión principal es `2` o `3`. Esos valores representan la compatibilidad de Aurora MySQL con MySQL 5.7 u 8.0 respectivamente. La versión secundaria representa la versión de las características que hay en las series 2.x o 3.x. El nivel de parche comienza en `0` para cada versión secundaria y representa el conjunto de correcciones de errores posteriores que se aplican a la versión secundaria. Ocasionalmente, una nueva característica se incorpora en una versión secundaria, pero no se hace visible de inmediato. En estos casos, la característica se somete a un ajuste preciso y se hace pública en un nivel de parche posterior.

Todas las versiones del motor Aurora MySQL 2.x son compatibles por conexión con Community MySQL 5.7.12 o posterior. Todas las versiones del motor Aurora MySQL 3.x son compatibles por conexión con MySQL 8.0.23 o posterior. Puede consultar las notas de la versión 3.x específica para encontrar cuál es la versión compatible con MySQL correspondiente.

Por ejemplo, las versiones del motor para Aurora MySQL 3.04.0 y 2.11.2 son las siguientes.

```
8.0.mysql_aurora.3.04.0
5.7.mysql_aurora.2.11.2
```

> **Note**
>
> No hay correspondencia uno a uno entre las versiones de MySQL de la comunidad y las versiones 2.x de Aurora MySQL. Para Aurora MySQL versión 3, hay una asignación más directa. Para comprobar qué correcciones de errores y nuevas características se encuentran en una versión de Aurora MySQL concreta, consulte Database engine updates for Amazon Aurora MySQL versión 3 (Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 3) y Database engine updates for Amazon Aurora MySQL versión 2 (Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 2) en las Notas de la versión de Aurora MySQL. Para obtener una lista cronológica de las nuevas características y versiones, consulte Historial de revisión. Para comprobar la versión mínima necesaria para una corrección relacionada con la seguridad, consulte las vulnerabilidades de seguridad corregidas en Aurora MySQL en las notas de la versión de Aurora MySQL.

Puede especificar la versión del motor de Aurora MySQL en algunos comandos de la AWS CLI y en operaciones de la API de RDS. Por ejemplo, especifique la opción `--engine-version` al ejecutar los comandos de la AWS CLI create-db-clúster y modify-db-clúster. Especifique el parámetro `EngineVersion` al ejecutar las operaciones de la API de RDS CreateDBclúster y ModifyDBclúster.

En la versión 2 de Aurora MySQL y versiones posteriores, la versión del motor en la Consola de administración de AWS incluye también la versión de Aurora. La actualización del clúster cambia el valor mostrado. Este cambio le ayuda a especificar y comprobar las versiones de Aurora MySQL precisas, sin necesidad de conectarse al clúster ni ejecutar ningún comando SQL.

> **Tip**
>
> Para clústeres de Aurora administrados mediante CloudFormation, este cambio en la configuración `EngineVersion` puede desencadenar acciones en CloudFormation. Para obtener información sobre cómo CloudFormation trata los cambios en la configuración de `EngineVersion`, consulte la documentación de CloudFormation.

Comprobación de versiones de Aurora MySQL mediante SQL

Los números de versión Aurora que puede recuperar en la aplicación mediante consultas SQL utilizan el formato *<major version>*.*<minor version>*.*<patch version>*. Puede obtener este número de versión para cualquier instancia de base de datos del clúster de Aurora MySQL consultando la variable de sistema `AURORA_VERSION`. Para obtener este número de versión, utilice una de las siguientes consultas.

```
select aurora_version();
select @@aurora_version;
```

Esas consultas producen resultados similares a los siguientes.

```
mysql> select aurora_version(), @@aurora_version;
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

Los números de versión devueltos por la consola, el CLI y la API de RDS utilizando las técnicas descritas en Comprobación o especificación de versiones del motor de Aurora MySQL mediante AWS suelen ser más descriptivos.

Versiones beta y de soporte a largo plazo (LTS) para Amazon Aurora MySQL

Aurora MySQL ofrece versiones beta y de soporte a largo plazo (LTS) para algunas versiones del motor de Aurora MySQL.

Temas

- Versiones de soporte a largo plazo (LTS) de Aurora MySQL
- Versiones beta de Aurora MySQL

Versiones de soporte a largo plazo (LTS) de Aurora MySQL

Cada versión nueva de Aurora MySQL sigue estando disponible durante cierta cantidad de tiempo para que la utilice al crear o actualizar un clúster de bases de datos. Después de este período, debe actualizar los clústeres que utilicen dicha versión. Puede actualizar manualmente el clúster antes de que finalice el período de soporte o Aurora puede actualizarlo automáticamente cuando su versión Aurora MySQL deje de admitirse.

Aurora designa determinadas versiones de Aurora MySQL como versiones con "soporte a largo plazo" (LTS). Los clústeres de base de datos que utilizan versiones LTS pueden mantenerse en la misma versión durante más tiempo y se someten a menos ciclos de actualización que los clústeres que utilizan versiones que no son LTS. Los clústeres de base de datos que utilizan versiones LTS pueden mantenerse en la misma versión secundaria durante al menos tres años o hasta que finalice el soporte estándar para la versión principal, lo que ocurra primero. Cuando hay que actualizar un clúster de bases de datos que está en una versión LTS, Aurora lo actualiza a la siguiente versión de LTS. De este modo, no es necesario volver a actualizar el clúster durante mucho tiempo.

Durante la vida útil de una versión LTS de Aurora MySQL, los nuevos niveles de parche introducen correcciones para problemas importantes. Los niveles de parche no incluyen características nuevas. Puede elegir si desea aplicar dichos parches a clústeres de base de datos que ejecuten la versión LTS. Recomendamos a los clientes que ejecuten versiones LTS que actualicen al último parche de la versión secundaria de LTS al menos una vez al año para aprovechar las correcciones operativas y de seguridad de alta gravedad. Para determinadas correcciones críticas, Amazon podría llevar a cabo una actualización administrada a un nivel de parche dentro de la misma versión LTS. Dichas actualizaciones administradas se llevan a cabo de forma automática en el periodo de mantenimiento del clúster. Todas las versiones de Aurora MySQL (las versiones LTS y las que no lo son) se someten a exhaustivas pruebas de estabilidad y funcionamiento. Algunas versiones secundarias están diseñadas como versiones LTS para que los clientes puedan utilizar esas versiones secundarias durante más tiempo sin necesidad de actualizar a una versión secundaria más reciente.

Le recomendamos que actualice a la versión más reciente, en lugar de utilizar la versión LTS, para la mayoría de los clústeres de Aurora MySQL. Al hacerlo se aprovecha Aurora como servicio administrado y le ofrece acceso a las características y correcciones de errores más recientes. Las versiones LTS están destinadas a clústeres con las siguientes características:

- No puede permitirse períodos de inactividad en la aplicación Aurora MySQL para actualizaciones fuera de los raros casos para parches críticos.
- El ciclo de prueba del clúster y las aplicaciones asociadas tarda mucho tiempo para cada actualización al motor de base de datos de Aurora MySQL.
- La versión de base de datos para su clúster de Aurora MySQL tiene todas las características de motor de base de datos y correcciones de errores que necesita su aplicación.

Las versiones LTS actuales para Aurora MySQL son las siguientes:

- Aurora MySQL versión 3.10*.
- Aurora MySQL versión 3.04*.

Para obtener más información sobre la versión LTS, consulte la sección sobre las actualizaciones del motor de base de datos de la versión 3 de Amazon Aurora MySQL en las notas de la versión de Aurora MySQL.

> **Note**
>
> Le recomendamos que desactive las actualizaciones automáticas de versiones secundarias para las versiones LTS. Establezca el parámetro `AutoMinorVersionUpgrade` en `false` o desactive la casilla Habilitar actualización automática de versiones secundarias en la Consola de administración de AWS.
> Si no las desactiva, el clúster de base de datos podría actualizarse a una versión que no sea LTS.

Versiones beta de Aurora MySQL

Una versión beta de Aurora MySQL es una versión preliminar que solo incluye correcciones de seguridad en un número limitado de Regiones de AWS. Estas correcciones se implementarán posteriormente de forma más generalizada en todas las regiones con la próxima versión de parche.

La numeración de una versión beta es similar a la de una versión secundaria de Aurora MySQL, pero con un cuarto dígito adicional, por ejemplo, 2.12.0.1 o 3.05.0.1.

Para obtener más información, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 2 y Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 3 en las Notas de la versión de Aurora MySQL.

Preparación para el final del soporte estándar de la versión 2 de Amazon Aurora MySQL-Compatible Edition

El fin del soporte estándar para la versión 2 de Amazon Aurora MySQL-Compatible Edition (compatible con MySQL 5.7) está previsto para el 31 de octubre de 2024. Recomendamos actualizar todos los clústeres que ejecutan la versión 2 de Aurora MySQL a la versión 3 predeterminada (compatible con MySQL 8.0) o superior antes de que la versión 2 de Aurora MySQL llegue al final de su periodo de soporte estándar. El 31 de octubre de 2024, Amazon RDS inscribirá automáticamente sus bases de datos en el Soporte extendido de Amazon RDS. Si utiliza la versión 2 de Amazon Aurora MySQL (compatible con MySQL 5.7) en un clúster de Aurora Serverless versión 1, esto no es aplicable en su caso. Si desea actualizar los clústeres de la versión 1 de Aurora Serverless a la versión 3 de Aurora MySQL, consulte Ruta de actualización para clústeres de bases de datos de Aurora Serverless v1.

Puede encontrar las próximas fechas del fin de soporte de las versiones principales de Aurora MySQL en el Calendario de lanzamientos para las versiones principales de Aurora MySQL.

Si tiene clústeres que utilizan Aurora MySQL versión 2, recibirá avisos periódicos con la última información sobre el modo de actualizarlos a medida que se acerque la fecha del fin del soporte estándar. Esta página se actualizará periódicamente con la información más reciente.

Calendario del fin del soporte estándar

1. Desde ahora y hasta el 31 de octubre de 2024: puede actualizar clústeres de la versión 2 de Aurora MySQL (compatible con MySQL 5.7) a la versión 3 de Aurora MySQL (compatible con MySQL 8.0).
2. 31 de octubre de 2024: en esta fecha, finaliza el soporte estándar de Aurora MySQL versión 2 y Amazon RDS inscribe automáticamente los clústeres en el Soporte extendido de Amazon RDS.

Lo inscribiremos automáticamente en el Soporte extendido de RDS. Para obtener más información, consulte Soporte extendido de Amazon RDS con Amazon Aurora.

Encontrar clústeres afectados por este proceso de fin de vida útil

Para encontrar clústeres afectados por este proceso de fin de vida útil, siga los siguientes procedimientos.

> **Important**
>
> Asegúrese de seguir estas instrucciones en cada Región de AWS y por cada Cuenta de AWS donde haya recursos.

Consola

Para encontrar un clúster de la versión 2 de Aurora MySQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. En el cuadro Filter by databases (Filtrar por bases de datos), introduzca 5.7.
4. Busque Aurora MySQL en la columna del motor.

AWS CLI

Para buscar clústeres afectados por este proceso de fin de vida útil mediante la AWS CLI, llame al comando describe-db-clústers. Puede usar el siguiente script de muestra.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?(Engine==`aurora-
mysql` && contains(EngineVersion,`5.7.mysql_aurora`))].{EngineVersion:EngineVersion,
 DBClusterIdentifier:DBClusterIdentifier, EngineMode:EngineMode}' --output table
 --region us-east-1

       +-------------------------------------------------------------+
       |                      DescribeDBClusters                     |
       +---------------------+---------------+-------------------------+
       |         DBCI        |      EM       |          EV             |
       +---------------------+---------------+-------------------------+
       |    aurora-mysql2    |  provisioned  | 5.7.mysql_aurora.2.11.3 |
```

```
| aurora-serverlessv1 |   serverless  | 5.7.mysql_aurora.2.11.3 |
+---------------------+---------------+-------------------------+
```

API de RDS

Para encontrar clústeres de base de datos de Aurora MySQL que ejecuten la versión 2 de Aurora MySQL, utilice la operación de la API DescribeDBclústers de RDS con los siguientes parámetros obligatorios:

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nombre
      - engine
    - Values.Value.N
      - ['aurora']

Soporte extendido de Amazon RDS

Puede utilizar el Soporte extendido de Amazon RDS en lugar de MySQL 5.7 comunitario sin costo alguno hasta la fecha de finalización del soporte, que es el 31 de octubre de 2024. El 31 de octubre de 2024, Amazon RDS inscribirá automáticamente sus bases de datos en el Soporte extendido de RDS para la versión 2 de Aurora MySQL. El Soporte extendido de RDS para Aurora es un servicio de pago que proporciona hasta 28 meses adicionales de soporte para la versión 2 de Aurora MySQL, hasta el fin del Soporte extendido de RDS en febrero de 2027. El soporte extendido de RDS solo se ofrecerá para las versiones secundarias 2.11 y 2.12 de Aurora MySQL. Para usar la version 2 de Amazon Aurora MySQL tras el periodo de soporte estándar, debe planear que sus bases de datos se ejecuten en una de estas versiones secundarias antes del 31 de octubre de 2024.

Para obtener más información acerca del Soporte extendido de RDS, como los cargos y otros aspectos, consulte Soporte extendido de Amazon RDS con Amazon Aurora.

Aplicación de una actualización

La actualización entre versiones principales requiere una planificación y pruebas más extensas que para una versión secundaria. El proceso puede llevar mucho tiempo. La actualización puede considerarse como un proceso en tres pasos, con actividades ejecutadas antes, durante y después de la actualización.

Antes de la actualización:

Antes de la actualización, recomendamos comprobar la compatibilidad con las aplicaciones, el rendimiento, los procedimientos de mantenimiento y otras consideraciones similares para el clúster actualizado, a fin de comprobar que las aplicaciones funcionarán del modo esperado después de la actualización. Las siguientes son cinco recomendaciones que ayudarán a mejorar la experiencia de actualización.

- En primer lugar, es fundamental entender Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL.
- Luego, analice cuáles son las técnicas de actualización disponibles al Actualización de Aurora MySQL versión 2 a versión 3.
- Para ayudarle a decidir el momento y el método adecuados para la actualización, infórmese sobre las diferencias entre Aurora MySQL (versión 3) y su entorno actual con Comparación entre Aurora MySQL versión 2 y Aurora MySQL versión 3.
- Una vez que se haya decidido por la opción que resulte más práctica y que funcione mejor, intente simular una actualización local en un clúster clonado, con la ayuda de Planificación de una actualización de versión principal para un clúster Aurora MySQL.
- Consulte el Comprobaciones previas de actualización de versiones principales para Aurora MySQL. El comprobador previo de actualización se puede ejecutar para determinar si la base de datos se actualizará correctamente o si habrá algún problema de incompatibilidad de las aplicaciones después de la actualización, además de evaluar el rendimiento, los procedimientos de mantenimiento y otras consideraciones similares.
- No todos los tipos o versiones de clústeres Aurora MySQL pueden utilizar el mecanismo de actualización local. Para obtener más información, consulte Rutas de actualización de versión principal Aurora MySQL.

Si tiene alguna duda, el equipo de soporte de AWS está disponible en los foros comunitarios y el soporte Premium.

Ejecución de la actualización:

Puede utilizar una de las siguientes técnicas de actualización. El tiempo de inactividad que sufra el sistema dependerá de la técnica elegida.

- Implementaciones azul/verde: cuando la prioridad sea reducir el tiempo de inactividad de las aplicaciones, puede utilizar implementaciones azul/verde de Amazon RDS para realizar

la actualización de la versión principal en los clústeres de base de datos de Amazon Aurora aprovisionados. Una implementación azul/verde crea un área de almacenamiento provisional que copia el entorno de producción. Puede realizar algunos cambios en el clúster de base de datos de Aurora del entorno verde (transitorio) sin afectar a las cargas de trabajo de producción. La conmutación suele tardar menos de un minuto y no hay pérdida de datos. Para obtener más información, consulte Descripción general de las implementaciones azul/verde de Amazon Aurora. Esto minimiza el tiempo de inactividad, pero requiere el uso de recursos adicionales mientras se realiza la actualización.

- Actualizaciones locales: puede hacer una actualización local; Aurora ejecutará automáticamente un proceso de verificación previa, desconectará el clúster, hará una copia de seguridad, realizará la actualización y volverá a ponerlo en línea. La actualización local de una versión principal se puede realizar con unos pocos clics y no implica ninguna otra coordinación o conmutación por error a otros clústeres, pero conlleva un tiempo de inactividad. Para obtener más información, consulte Pasos para realizar una actualización local
- Restauración de instantáneas: puede actualizar un clúster de la versión 2 de Aurora MySQL restaurando una instantánea de la versión 2 de Aurora MySQL como un clúster de la versión 3 de Aurora MySQL. Para ello debe crear primero una instantánea y despues restaurarla. Ese proceso implica interrumpir la base de datos, ya que la restauración se hace a partir de una instantánea.

Después de la actualización

Tras la actualización, debe supervisar de cerca el sistema (aplicación y base de datos) y realizar ajustes precisos si es necesario. Si sigue de cerca los pasos previos a la actualización, se minimizarán los cambios necesarios. Para obtener más información, consulte Solución de problemas de rendimiento de las bases de datos Amazon Aurora MySQL.

Para obtener más información sobre los métodos, la planificación, las pruebas y la solución de problemas de las actualizaciones de la versión principal de Aurora MySQL, lea con atención Actualización de la versión principal de un clúster de base de datos de Amazon Aurora MySQL, lo que incluye Solución de problemas para la actualización Aurora MySQL en el lugar. Además, tenga en cuenta que algunos tipos de instancia no son compatibles con la versión 3 de Aurora MySQL. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.

Ruta de actualización para clústeres de bases de datos de Aurora Serverless v1

La actualización entre versiones principales requiere una planificación y pruebas más extensas que para una versión secundaria. El proceso puede llevar mucho tiempo. Debemos considerar la

actualización como un proceso en tres fases, con actividades previas a la actualización, durante la actualización y después de la actualización.

La versión 2 de Aurora MySQL (compatible con MySQL 5.7) seguirá recibiendo soporte estándar para clústeres de Aurora Serverless v1.

Si desea actualizar a Amazon Aurora MySQL 3 (compatible MySQL 8.0) y seguir utilizando Aurora Serverless, puede utilizar Amazon Aurora Serverless v2. Para entender las diferencias entre Aurora Serverless v1 y Aurora Serverless v2, consulte Comparación de Aurora Serverless v2 y Aurora Serverless v1.

Actualizar a:Aurora Serverless v2 puede actualizar un clúster de Aurora Serverless v1 a Aurora Serverless v2. Para obtener más información, consulte Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.

Preparación para el final de la vida útil de la versión 1 de la Edición compatible con MySQL de Amazon Aurora

Está previsto que la versión 1 de la Edición compatible con MySQL de Amazon Aurora (con compatibilidad con MySQL 5.6) llegue al final de su vida útil el 28 de febrero de 2023. Amazon recomienda actualizar todos los clústeres (aprovisionados y Aurora Serverless) que ejecutan la versión 1 de Aurora MySQL a la versión 2 de Aurora MySQL (con compatibilidad con MySQL 5.7) o la versión 3 de Aurora MySQL (con compatibilidad con MySQL 8.0). Hágalo antes de que la versión 1 de Aurora MySQL llegue al final de su periodo de soporte.

Para los clústeres de base de datos aprovisionados de Aurora, puede completar las actualizaciones la versión 1 de Aurora MySQL a la versión 2 de Aurora MySQL mediante varios métodos. Puede encontrar instrucciones para el mecanismo de actualización in situ en Pasos para realizar una actualización local. Otra forma de completar la actualización es tomar una instantánea de un clúster de la versión 1 de Aurora MySQL y restaurar la instantánea a un clúster de la versión 2 de Aurora MySQL. O puede seguir un proceso de varios pasos que ejecuta los clústeres antiguo y nuevo en paralelo. Para obtener más información sobre cada método, consulte Actualización de la versión principal de un clúster de base de datos de Amazon Aurora MySQL.

Para los clústeres de base de datos de Aurora Serverless v1, puede realizar una actualización in situ de la versión 1 a la 2 de Aurora MySQL. Para obtener más información sobre este método, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1.

Para los clústeres de base de datos aprovisionados de Aurora, puede completar las actualizaciones la versión 1 de Aurora MySQL a la versión 3 de Aurora MySQL mediante un proceso de actualización en dos etapas:

1. Actualice de la versión 1 de Aurora MySQL a la versión 2 de Aurora MySQL utilizando los métodos ya descritos.
2. Actualice de la versión 2 de Aurora MySQL a la versión 3 utilizando los mismos métodos que para actualizar de la versión 1 a la versión 2. Para obtener más información, consulte Actualización de Aurora MySQL versión 2 a versión 3. Anote el Diferencias de características entre las versiones 2 y 3 de Aurora MySQL.

Puede encontrar las próximas fechas de finalización de la vida útil de las versiones principales de Aurora en Versiones de Amazon Aurora. Amazon actualiza automáticamente cualquier clúster que no se actualice antes de la fecha de finalización de la vida útil. Después de la fecha de finalización de la vida útil, estas actualizaciones automáticas a la versión principal posterior se producen durante una ventana de mantenimiento programada para los clústeres.

Los siguientes son hitos adicionales para actualizar los clústeres de la versión 1 de Aurora MySQL (aprovisionados y Aurora Serverless) que están llegando al final de su vida útil. Para cada uno, la hora de inicio es 00:00 del tiempo universal coordinado (UTC).

1. Desde ahora hasta el 28 de febrero de 2023: en cualquier momento puede iniciar actualizaciones de clústeres de la versión 1 de Aurora MySQL (con compatibilidad con MySQL 5.6) a la versión 2 de Aurora MySQL (con compatibilidad con MySQL 5.7). Desde la versión 2 de Aurora MySQL, puede realizar una actualización adicional a la versión 3 de Aurora MySQL (con compatibilidad con MySQL 8.0) para clústeres de base de datos aprovisionados de Aurora.
2. 16 de enero de 2023: después de este tiempo, no puede crear nuevos clústeres o instancias de la versión 1 de Aurora MySQL desde la Consola de administración de AWS o la AWS Command Line Interface (AWS CLI). Tampoco puede agregar nuevas regiones secundarias a una base de datos global de Aurora. Esto podría afectar su capacidad para recuperarse de una interrupción no planificada como se describe en Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada, ya que no puede completar los pasos 5 y 6 después de este tiempo. Tampoco podrá crear una nueva réplica de lectura entre regiones que ejecute la versión 1 de Aurora MySQL. Todavía puede hacer lo siguiente para los clústeres existentes de la versión 1 de Aurora MySQL hasta el 28 de febrero de 2023:
   - Restaure una instantánea tomada de un clúster de la versión 1 de Aurora MySQL a la misma versión que tenía el clúster de la instantánea original.

- Agregue réplicas de lectura (no aplicable para clústeres de bases de datos Aurora Serverless).
- Cambiar la configuración de la instancia
- Realizar una restauración a un momento dado
- Crear clones de clústeres de la versión 1 existentes
- Cree una nueva réplica de lectura entre regiones que ejecute Aurora MySQL versión 2 o posterior.

3. 28 de febrero de 2023: después de este tiempo, planeamos actualizar automáticamente los clústeres de la versión 1 de Aurora MySQL a la versión 2 predeterminada de Aurora MySQL dentro de una ventana de mantenimiento programada a continuación. La restauración de instantáneas de base de datos de la versión 1 de Aurora MySQL da como resultado una actualización automática del clúster restaurado a la versión 2 predeterminada de Aurora MySQL en ese momento.

La actualización entre versiones principales requiere una planificación y pruebas más extensas que para una versión secundaria. El proceso puede llevar mucho tiempo.

Para situaciones en las que la principal prioridad sea reducir el tiempo de inactividad, también puede utilizar implementaciones azul/verde para realizar la actualización de la versión principal en los clústeres de base de datos de Amazon Aurora aprovisionados. Una implementación azul/verde crea un área de almacenamiento provisional que copia el entorno de producción. Puede realizar cambios en el clúster de base de datos de Aurora en un entorno verde (transitorio) sin que eso afecte a las cargas de trabajo de producción. La conmutación suele tardar menos de un minuto sin que se produzca una pérdida de datos y sin la necesidad de realizar cambios en la aplicación. Para obtener más información, consulte Descripción general de las implementaciones azul/verde de Amazon Aurora.

Una vez finalizada la actualización, es posible que también deba hacer trabajo de seguimiento. Por ejemplo, es posible que deba realizar un seguimiento debido a las diferencias en la compatibilidad de SQL, la forma en que funcionan ciertas funciones relacionadas con MySQL o la configuración de parámetros entre la versión anterior y la nueva.

Para obtener más información sobre los métodos, la planificación, las pruebas y la solución de problemas de las actualizaciones de la versión principal de Aurora MySQL, asegúrese de leer con atención Actualización de la versión principal de un clúster de base de datos de Amazon Aurora MySQL.

Encontrar clústeres afectados por este proceso de fin de vida útil

Para encontrar clústeres afectados por este proceso de fin de vida útil, siga los siguientes procedimientos.

> **Important**
>
> Asegúrese de seguir estas instrucciones en cada Región de AWS y por cada Cuenta de AWS donde haya recursos.

Consola

Para encontrar un clúster de la versión 1 de Aurora MySQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. En el cuadro Filter by databases (Filtrar por bases de datos), ingrese 5.6.
4. Busque Aurora MySQL en la columna del motor.

AWS CLI

Para buscar clústeres afectados por este proceso de fin de vida útil mediante la AWS CLI, llame al comando describe-db-clústers. Puede usar el siguiente script de muestra.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?Engine==`aurora`].
{EV:EngineVersion, DBCI:DBClusterIdentifier, EM:EngineMode}' --output table --region
 us-east-1

        +-----------------------------------------+
        |            DescribeDBClusters           |
        +---------------+--------------+----------+
        |     DBCI      |     EM       |    EV    |
        +---------------+--------------+----------+
        |  my-database-1|  serverless  |  5.6.10a |
        +---------------+--------------+----------+
```

API de RDS

Para encontrar clústeres de base de datos de Aurora MySQL que ejecuten la versión 1 de Aurora MySQL, utilice la operación de la API DescribeDBclústers de RDS con los siguientes parámetros obligatorios:

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nombre
      - engine
    - Values.Value.N
      - ['aurora']

Actualización de clústeres de base Amazon Aurora MySQL

Puede actualizar un clúster de bases de datos de Aurora MySQL para obtener correcciones de errores, nuevas características de Aurora MySQL o para cambiar a una versión completamente nueva del motor de base de datos subyacente. En las siguientes secciones se muestra cómo hacerlo.

> **Note**
>
> El tipo de actualización que realice depende de cuánto tiempo de inactividad puede permitirse para su clúster, cuántas pruebas de verificación planea realizar, cuán importantes son las correcciones de errores específicas o las nuevas características para su caso de uso, y si planea realizar actualizaciones pequeñas frecuentes o actualizaciones ocasionales que salten varias versiones intermedias. Para cada actualización, puede cambiar la versión principal, la versión secundaria y el nivel de parche del clúster. Si no está familiarizado con la distinción entre versiones Aurora MySQL principales, versiones secundarias y niveles de parche, puede leer la información de fondo en Comprobación de los números de versión de Aurora MySQL.

> **Tip**
>
> Puede minimizar el tiempo de inactividad necesario para la actualización de un clúster de base de datos mediante una implementación azul/verde. Para obtener más información,

consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Temas

- Actualización de la versión secundaria o el nivel de parche de un clúster de bases de datos Aurora MySQL
- Actualización de la versión principal de un clúster de base de datos de Amazon Aurora MySQL

Actualización de la versión secundaria o el nivel de parche de un clúster de bases de datos Aurora MySQL

Puede utilizar los métodos siguientes para actualizar la versión secundaria de un clúster de bases de datos o para aplicar parches a un clúster de bases de datos:

- Actualización de Aurora MySQL mediante la modificación de la versión del motor (para las versiones 2 y 3 de Aurora MySQL)
- Activación de actualizaciones automáticas entre versiones secundarias de Aurora MySQL

Para obtener información acerca de cómo la aplicación de parches sin tiempo de inactividad puede reducir las interrupciones durante el proceso de actualización, consulte Uso de parches sin tiempo de inactividad.

Para obtener información sobre cómo realizar una actualización de una versión secundaria del clúster de base de datos de Aurora MySQL, consulte los siguientes temas.

Temas

- Antes de realizar una actualización de versión secundaria
- Comprobaciones previas de actualización de versiones secundarias para Aurora MySQL
- Actualización de Aurora MySQL mediante la modificación de la versión del motor
- Activación de actualizaciones automáticas entre versiones secundarias de Aurora MySQL
- Uso de parches sin tiempo de inactividad
- Técnica alternativa de actualización azul/verde

Antes de realizar una actualización de versión secundaria

Le recomendamos que lleve a cabo las siguientes acciones para reducir el tiempo de inactividad durante una actualización de versión secundaria:

- El mantenimiento del clúster de base de datos Aurora debe realizarse durante un periodo de poco tráfico. Utilice Performance Insights para identificar estos periodos de tiempo y configurar correctamente los plazos de mantenimiento. Para obtener más información sobre Performance Insights, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon RDS. Para obtener más información sobre los periodos de mantenimiento de clústeres de base de datos, consulte Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos.
- Utilice los SDK de AWS que admitan fluctuaciones y retrocesos exponenciales como procedimiento recomendado. Para obtener más información, consulte Exponential Backoff And Jitter.

Comprobaciones previas de actualización de versiones secundarias para Aurora MySQL

Al iniciar una actualización de una versión secundaria, Amazon Aurora ejecuta comprobaciones previas automáticamente.

Estas comprobaciones previas son obligatorias. No tiene la opción de omitirlas. Las comprobaciones previas proporcionan las siguientes ventajas:

- Le permiten evitar tiempos de inactividad no planeados durante la actualización.
- Si hay incompatibilidades, Amazon Aurora impide la actualización y proporciona un registro para que se informe sobre ellas. A continuación, podrá usar el registro para preparar la base de datos para la actualización reduciendo así las incompatibilidades. Para obtener información detallada acerca de cómo eliminar incompatibilidades, consulte Preparing your installation for upgrade en la documentación de MySQL.

Las comprobaciones previas se ejecutan antes de detenerse la instancia de base de datos para la actualización, lo que quiere decir que no causan tiempos de inactividad al ejecutarse. Si las verificaciones previas encuentran una incompatibilidad, Aurora cancela automáticamente la actualización antes de detenerse la instancia de base de datos. Aurora también genera un evento por la incompatibilidad. Para obtener más información acerca de los eventos de Amazon Aurora, consulte Uso de notificaciones de eventos de Amazon RDS.

Aurora registra información detallada acerca de cada incompatibilidad en el archivo de registro `PrePatchCompatibility.log`. En la mayoría de los casos, la entrada de registro incluye un vínculo a la documentación de SQL para corregir la incompatibilidad. Para obtener más información acerca de cómo visualizar los archivos de registro, consulte Visualización y descripción de archivos de registro de base de datos.

Debido a la naturaleza de las comprobaciones previa, analizan objetos en su base de datos. Este análisis genera un consumo de recursos y aumenta el tiempo de ejecución de la actualización.

Actualización de Aurora MySQL mediante la modificación de la versión del motor

La actualización de la versión secundaria de un clúster de base de datos de Aurora MySQL aplica correcciones adicionales y nuevas características a un clúster existente.

Este tipo de actualización se aplica a los clústeres de Aurora MySQL en los que la versión original y la versión actualizada tienen la misma versión principal de Aurora MySQL, ya sea 2 o 3. El proceso es rápido y sencillo, ya que no implica ninguna conversión para los metadatos de Aurora MySQL o la reorganización de los datos de la tabla.

Para realizar este tipo de actualización modificando la versión del motor del clúster de bases de datos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS. Por ejemplo, si el clúster está ejecutando Aurora MySQL 3.x, elija una versión posterior a la 3.x.

Si está realizando una actualización secundaria en una base de datos global de Aurora, actualice todos los clústeres secundarios antes de actualizar el clúster principal.

> **Note**
>
> Para realizar una actualización de la versión secundaria a la versión 3.04.* o posterior, o a la versión 2.12.* de Aurora MySQL, utilice el siguiente proceso:
>
> 1. Elimine todas las regiones secundarias del clúster global. Siga los pasos de Eliminación de un clúster de una base de datos global de Amazon Aurora.
> 2. Actualice la versión del motor de la región principal a la versión 3.04.* o posterior, o a la versión 2.12.*, según corresponda. Siga los pasos de To modify the engine version of a DB cluster.
> 3. Añada regiones secundarias al clúster global. Siga los pasos de Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Para modificar la versión del motor de un clúster de bases de datos

- Mediante la consola: modifique las propiedades del clúster. En la ventana Modify DB clúster (Modificar clúster de bases de datos), cambie la versión del motor de Aurora MySQL en la casilla DB engine version (Versión del motor de base de datos). Si no está familiarizado con el procedimiento general para modificar un clúster, siga las instrucciones de Modificación del clúster de base de datos con la consola, CLI y API.
- Mediante la:AWS CLI llame al comando modify-db-clúster de la AWS CLI y especifique el nombre del clúster de bases de datos para la opción `--db-cluster-identifier` y la versión del motor para la opción `--engine-version`.

  Por ejemplo, para actualizar a la versión 3.04.1 de Aurora MySQL, establezca la opción `--engine-version` en `8.0.mysql_aurora.3.04.1`. Especifique la opción `--apply-immediately` para actualizar inmediatamente la versión del motor para su clúster de bases de datos.
- Mediante la API de RDS: llame a la operación ModifyDBclúster de la API y especifique el nombre de su clúster de bases de datos para el parámetro `DBClusterIdentifier` y la versión del motor para el parámetro `EngineVersion`. Establezca el parámetro `ApplyImmediately` en `true` para actualizar inmediatamente la versión del motor para el clúster de bases de datos.

Activación de actualizaciones automáticas entre versiones secundarias de Aurora MySQL

Para un clúster de bases de datos Amazon Aurora MySQL puede especificar que Aurora actualice automáticamente el clúster de base de datos a nuevas versiones secundarias. Para ello, establezca la propiedad `AutoMinorVersionUpgrade` (Actualización automática de la versión secundaria en la Consola de administración de AWS) del clúster de base de datos.

La actualización automática se produce durante el período de mantenimiento. Si las instancias de base de datos individuales del clúster de base de datos tienen períodos de mantenimiento diferentes a las de la ventana de mantenimiento del clúster, la ventana de mantenimiento del clúster tiene prioridad.

La actualización automática de versiones secundarias no se aplica a los siguientes tipos de clústeres de Aurora MySQL:

- Clústeres que forman parte de una base de datos de Aurora global
- Clústeres que tienen réplicas entre regiones

La duración de la interrupción varía en función de la carga de trabajo, el tamaño del clúster, la cantidad de datos de registro binarios y si Aurora puede utilizar la característica de parches de tiempo de inactividad cero (ZDP). Aurora reinicia el clúster de bases de datos, por lo que es posible que experimente un breve periodo de falta de disponibilidad antes de reanudar el uso del clúster. En concreto, la cantidad de datos de log binario afecta al tiempo de recuperación. La instancia de base de datos procesa los datos del registro binario durante la recuperación. Por lo tanto, un gran volumen de datos de registro binario aumenta el tiempo de recuperación.

> **Note**
>
> Aurora solo realiza actualizaciones automáticas si todas las instancias de base de datos del clúster tienen la configuración `AutoMinorVersionUpgrade` habilitada. Para obtener información sobre cómo configurarla y cómo funciona cuando se aplica a los niveles de clúster e instancia, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora.
>
> A continuación, si existe una ruta de actualización para las instancias del clúster de base de datos a una versión secundaria del motor de base de datos que tenga `AutoUpgrade` establecido en true, se realizará la actualización. La opción `AutoUpgrade` es dinámica y la establece RDS.
>
> Las actualizaciones de versiones secundarias automáticas se realizan a la versión secundaria predeterminada.

Puede utilizar un comando de la CLI como el siguiente para comprobar el estado de la configuración `AutoMinorVersionUpgrade` para todas las instancias de base de datos de los clústeres de Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

El resultado de este comando debería ser similar al siguiente:

```
[
  {
      "DBInstanceIdentifier": "db-t2-medium-instance",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": true
  },
```

```
  {
      "DBInstanceIdentifier": "db-t2-small-original-size",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "instance-2020-05-01-2332",
      "DBClusterIdentifier": "cluster-57-2020-05-01-4615",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

En este ejemplo, Habilitar actualización automática de versiones secundarias está desactivado para el clúster de base de datos `cluster-57-2020-06-03-6411`, porque está desactivado para una de las instancias de base de datos del clúster.

Uso de parches sin tiempo de inactividad

Realizar actualizaciones para clústeres de base de datos de Aurora MySQL implica la posibilidad de una interrupción cuando se cierra la base de datos y mientras se actualiza. De forma predeterminada, si comienza la actualización cuando la base de datos está ocupada, perderá todas las conexiones y transacciones que el clúster de bases de datos tiene en proceso. Si espera hasta que la base de datos esté inactiva para realizar la actualización, es posible que tenga que esperar mucho tiempo.

La característica de aplicación de parches sin tiempo de inactividad (ZDP) intenta, en la medida de lo posible, conservar las conexiones de cliente a través de una actualización de Aurora MySQL. Si la ZDP se completa correctamente, las sesiones de aplicación se conservan y el motor de base de datos se reinicia a la vez que la actualización está en curso. El reinicio del motor de base de datos puede provocar una disminución del rendimiento de unos segundos a aproximadamente un minuto.

La ZDP no se aplica a lo siguiente:

- Revisiones y actualizaciones del sistema operativo (OS)
- Actualizaciones de la versión principal

ZDP está disponible para todas las versiones de Aurora MySQL y todas las clases de instancia de base de datos admitidas.

ZDP no es compatible con Aurora Serverless v1 o las bases de datos globales de Aurora.

> **Note**
>
> Recomendamos que se usen solo las clases de instancia de base de datos T para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Utilización de clases de instancia T para el desarrollo y la prueba.

Puede ver métricas de atributos importantes durante la ZDP en el registro de errores de MySQL. También puede ver información sobre cuándo Aurora MySQL utiliza la ZDP o elige no usar la ZDP en la página Eventos en la Consola de administración de AWS.

En Aurora MySQL, Aurora puede realizar una revisión de tiempo de inactividad cero esté o no habilitada la replicación de registros binarios. Si la replicación de registros binarios está habilitada, Aurora MySQL elimina automáticamente la conexión al destino binlog durante una operación ZDP. Aurora MySQL se reconecta automáticamente al destino binlog y reanuda la reproducción después de que finalice el reinicio.

La ZDP también funciona en combinación con las mejoras de reinicio de Aurora MySQL. La aplicación de parches a la instancia de base de datos del escritor aplica parches de forma automática a los lectores a la vez. Después de aplicar el parche, Aurora restaura las conexiones tanto en las instancias de base de datos del escritor como del lector.

Es posible que la ZDP no se complete correctamente en las siguientes condiciones:

- Hay en curso consultas o transacciones de ejecución prolongada. Si Aurora puede llevar a cabo una ZDP en este caso, se cancelarán las transacciones abiertas, pero se mantendrán las conexiones.
- Se están utilizando tablas temporales, bloqueos de usuario o bloqueos de tabla; por ejemplo, mientras se ejecutan instrucciones de lenguaje de definición de datos (DDL). Aurora interrumpe estas conexiones.
- Existen cambios de parámetros pendientes.

Si no hubiera un período de tiempo apropiado para la realización de la ZDP debido a una o varias de estas condiciones, la aplicación de parches vuelve al comportamiento estándar.

Aunque las conexiones permanecen intactas tras una operación de ZDP correcta, se reinicializan algunas variables y características. Los siguientes tipos de información no se conservan durante un reinicio causado por una aplicación de parches sin tiempo de inactividad:

- Variables globales. Aurora restaura las variables de sesión, pero no restaura las variables globales después del reinicio.
- Variables de estado. En particular, el valor del tiempo de actividad que informa el estado del motor se restablece tras un reinicio que utiliza los mecanismos de ZDR o ZDP.
- `LAST_INSERT_ID`.
- Estado `auto_increment` en memoria para tablas. El estado de incremento automático en memoria se reinicializa. Para obtener más información sobre los valores de incremento automático, consulte el Manual de referencia de MySQL.
- Información de diagnóstico de las tablas `INFORMATION_SCHEMA` y `PERFORMANCE_SCHEMA`. Esta información de diagnóstico también aparece en la salida de comandos como `SHOW PROFILE` y `SHOW PROFILES`.

Las siguientes actividades relacionadas con el reinicio sin tiempo de inactividad se indican en la página Events (Eventos):

- Intento de actualizar la base de datos sin tiempo de inactividad.
- Intento de actualizar la base de datos sin tiempo de inactividad finalizado. El evento informa cuánto tiempo tardó el proceso. El evento también informa sobre cuántas conexiones se conservaron durante el reinicio y cuántas se han eliminado. Puede consultar el registro de errores de la base de datos para ver más detalles sobre lo que ocurrió durante el reinicio.

Técnica alternativa de actualización azul/verde

En algunas situaciones, su prioridad principal es realizar un cambio inmediato del clúster antiguo a uno actualizado. En tales situaciones, puede utilizar un proceso de varios pasos que ejecuta los clústeres antiguo y nuevo en paralelo. Aquí, replicará los datos del clúster anterior al nuevo hasta que esté listo para que el nuevo clúster asuma el control. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Actualización de la versión principal de un clúster de base de datos de Amazon Aurora MySQL

En un número de versión de Aurora MySQL, como 3.04.1, el 3 representa la versión principal. Aurora MySQL versión 2 es compatible MySQL 5.7 Aurora MySQL versión 3 es compatible MySQL 8.0.

La actualización entre versiones principales requiere una planificación y pruebas más extensas que para una versión secundaria. El proceso puede llevar mucho tiempo. Una vez finalizada la actualización, es posible que también deba hacer trabajo de seguimiento. Por ejemplo, esto puede ocurrir debido a las diferencias en la compatibilidad de SQL o la forma en que funcionan ciertas características relacionadas con MySQL. O puede ocurrir debido a la diferencia en la configuración de los parámetros entre la versión antigua y la nueva.

Contenido

- Actualización de Aurora MySQL versión 2 a versión 3
- Rutas de actualización de versión principal Aurora MySQL
- Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL
- Planificación de una actualización de versión principal para un clúster Aurora MySQL
  - Simulación de la actualización mediante la clonación del clúster de base de datos
  - Implementaciones azul/verde
- Comprobaciones previas de actualización de versiones principales para Aurora MySQL
  - Proceso de comprobación previa de Aurora MySQL
  - Formato de registro de comprobación previa de Aurora MySQL
  - Ejemplos de resultados del registro de comprobación previa de Aurora MySQL
  - Rendimiento de la comprobación previa de Aurora MySQL
  - Resumen de comprobaciones previas de actualizaciones de Community MySQL
  - Resumen de comprobaciones previas de actualizaciones de Aurora MySQL
  - Referencia de descripciones de comprobaciones previas de Aurora MySQL
    - Errores
      - Comprobaciones previas de MySQL que informan de errores
      - Comprobaciones previas de Aurora MySQL que informan de errores
    - Advertencias
      - Comprobaciones previas de MySQL que informan de advertencias
      - Comprobaciones previas de Aurora MySQL que informan de advertencias
    - Avisos
    - Errores, advertencias o avisos

- Pasos para realizar una actualización local
  - Cómo afectan las actualizaciones en el lugar a los grupos de parámetros de un clúster
  - Cambios en las propiedades del clúster entre versiones de Aurora MySQL
  - Actualizaciones mayores en el lugar para bases de datos globales
  - Consideraciones sobre el Backtrack
- Tutorial de actualización de Aurora MySQL en el lugar
- Determinación del motivo de los errores en una actualización de la versión principal de Aurora MySQL
- Solución de problemas para la actualización Aurora MySQL en el lugar
- Limpieza posterior a la actualización para Aurora MySQL versión 3
  - Índices espaciales

Actualización de Aurora MySQL versión 2 a versión 3

Si tiene un clúster compatible con MySQL 5.7 y desea actualizarlo a un clúster compatible con MySQL-8.0, puede hacerlo ejecutando un proceso de actualización en el propio clúster. Este tipo de actualización es una actualización en el lugar, en contraste con las actualizaciones que se realizan al crear un nuevo clúster. Esta técnica mantiene el mismo punto de enlace y otras características del clúster. La actualización es relativamente rápida ya que no requiere copiar todos los datos en un nuevo volumen de clúster. Esta estabilidad ayuda a minimizar cualquier cambio de configuración en las aplicaciones. También ayuda a reducir la cantidad de pruebas para el clúster actualizado. Esto se debe a que el número de instancias de base de datos y sus clases de instancia permanecen iguales.

El mecanismo de actualización en el lugar implica apagar el clúster de base de datos mientras se lleva a cabo la operación. Aurora realiza un cierre limpio y completa las operaciones pendientes, como la restauración de transacciones y la depuración de deshacer. Para obtener más información, consulte Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL.

El método de actualización in situ es práctico, ya que es fácil de realizar y minimiza los cambios de configuración en las aplicaciones asociadas. Por ejemplo, una actualización en el lugar conserva los puntos de enlace y el conjunto de instancias de base de datos del clúster. Sin embargo, el tiempo necesario para una actualización en el lugar puede variar en función de las propiedades del esquema y de cuán ocupado esté el clúster. Por lo tanto, según las necesidades de su clúster, puede elegir entre las técnicas de actualización:

- Actualización in situ

- Implementación azul/verde
- Restauración de instantáneas

> **Note**
>
> Si utiliza la AWS CLI o la API de RDS para el método de actualización de restauración de instantáneas, debe ejecutar una operación posterior para crear una instancia de base de datos de escritor en el clúster de base de datos restaurado.

Para obtener información general sobre Aurora MySQL versión 3 y sus nuevas características, consulte Aurora MySQL versión 3 compatible con MySQL 8.0.

Para obtener más información sobre la planificación de una actualización, consulte Planificación de una actualización de versión principal para un clúster Aurora MySQL y Pasos para realizar una actualización local.

Rutas de actualización de versión principal Aurora MySQL

No todos los tipos o versiones de clústeres Aurora MySQL pueden utilizar el mecanismo de actualización en el lugar. Puede aprender la ruta de actualización adecuada para cada clúster Aurora MySQL consultando la tabla siguiente.

Tipo de clúster de base de datos Aurora MySQL	¿Puede usar la actualización en el lugar?	Acción
Clúster aprovisionado de Aurora MySQL, versión 2	Sí	La actualización in situ está disponible para clústeres de Aurora MySQL compatibles con MySQL 5.7. Para obtener información acerca de la actualización a Aurora MySQL versión 3, consulte Planificación de una actualización de versión principal para un clúster Aurora MySQL y Pasos para realizar una actualización local.

Tipo de clúster de base de datos Aurora MySQL	¿Puede usar la actualización en el lugar?	Acción
Clúster aprovisionado de Aurora MySQL, versión 3	No aplicable	Utilice un procedimiento de actualización de versiones secundarias para actualizar entre las versiones 3 de Aurora MySQL.
Aurora Serverless v1Clúster de	No aplicable	Aurora Serverless v1 solo es compatible para la versión 2 de Aurora MySQL.
Aurora Serverless v2Clúster de	No aplicable	Aurora Serverless v2 solo es compatible para la versión 3 de Aurora MySQL.
Clúster en una base de datos global Aurora	Sí	Para actualizar la versión 2 de Aurora MySQL a la versión 3, siga el procedimiento para realizar una actualización local para clústeres en una base de datos global de Aurora. Realice la actualización en el clúster global. Aurora actualiza el clúster principal y todos los clústeres secundarios en la base de datos global al mismo tiempo. Si utiliza la API RDS o AWS CLI, llame al comando `modify-global-cluster` o la operación `ModifyGlobalCluster` en lugar de `modify-db-cluster` o `ModifyDBCluster` . Puede realizar una actualización local desde la versión 2 a la versión 3 de Aurora MySQL si establece el parámetro `lower_case_table_names` en predeterminado y reinicia la base de datos global. Para obtener más información, consulte Actualizaciones de la versión principal.

Tipo de clúster de base de datos Aurora MySQL	¿Puede usar la actualización en el lugar?	Acción
Clúster de consultas paralelas	Sí	Puede realizar una actualización local.
Clúster que es el destino de la replicación de registros binarios	Quizás	Si la replicación del registro binario se realiza desde un clúster de Aurora MySQL, puede realizar una actualización local. No puede realizar la actualización si la replicación del registro binario proviene de una instancia de base de datos de RDS para MySQL o de una instancia de base de datos de MySQL en las instalaciones. En ese caso, puede actualizar utilizando el mecanismo de restauración de instantáneas.
Cluster con cero instancias de base de datos	No	Mediante la AWS CLI o la API RDS, puede crear un clúster de Aurora MySQL sin ninguna instancia de base de datos adjunta. De la misma manera, también puede quitar todas las instancias de base de datos de un clúster Aurora MySQL mientras deja intactos los datos del volumen del clúster. Aunque un clúster tiene cero instancias de base de datos, no puede realizar una actualización en el lugar. El mecanismo de actualización requiere una instancia de escritor en el clúster para realizar conversiones en las tablas del sistema, archivos de datos, etc. En este caso, utilice AWS CLI o la API de RDS para crear una instancia de escritor para el clúster. Luego, puede realizar una actualización en el lugar.

Tipo de clúster de base de datos Aurora MySQL	¿Puede usar la actualización en el lugar?	Acción
Clúster con retroceso habilitado	Sí	Puede realizar una actualización in situ para un clúster Aurora MySQL que utilice la característica de retroceso . Sin embargo, después de la actualización, no puede realizar un retroceso del clúster hasta un momento anterior a la actualización.

Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL

Aurora MySQL realiza una actualización de versión principal como un proceso de varias etapas. Puede comprobar el estado actual de una actualización. Algunos de los pasos de actualización también proporcionan información sobre el progreso. A medida que comienza cada etapa, Aurora MySQL registra un evento. Puede examinar los eventos a medida que se producen en la página Eventos de la consola de RDS. Para obtener más información sobre el trabajo con eventos, consulte Uso de notificaciones de eventos de Amazon RDS.

> **Important**
>
> Una vez que el proceso comienza, se ejecuta hasta que la actualización se realiza correctamente o falla. No puede cancelar la actualización mientras está en marcha. Si la actualización falla, Aurora revierte todos los cambios y el clúster tiene la misma versión del motor, metadatos, etc. que antes.

El proceso de actualización consta de tres pasos:

1. Aurora realiza una serie de comprobaciones previas antes de comenzar el proceso de actualización. El clúster sigue ejecutándose mientras Aurora realiza estas comprobaciones. Por ejemplo, el clúster no puede tener transacciones XA en el estado preparado ni procesar ninguna declaración de lenguaje de definición de datos (DDL). Por ejemplo, es posible que deba cerrar las aplicaciones que están enviando ciertos tipos de instrucciones SQL. O puede simplemente

esperar hasta que se terminen ciertas instrucciones de larga duración. A continuación, intente la actualización de nuevo. Algunas comprobaciones prueban las condiciones que no impiden la actualización, pero pueden hacer que la actualización tarde mucho tiempo.

Si Aurora detecta que no se cumplen las condiciones requeridas, modifique las condiciones identificadas en los detalles del evento. Siga la guía de Solución de problemas para la actualización Aurora MySQL en el lugar. Si Aurora detecta condiciones que podrían provocar una actualización lenta, planee supervisar la actualización durante un período prolongado.

2. Aurora desconecta el clúster. Luego, Aurora realiza un conjunto similar de pruebas que en la etapa anterior, para confirmar que no surgieron nuevos problemas durante el proceso de apagado. Si Aurora detecta alguna condición en este punto que impida la actualización, Aurora cancela la actualización y vuelve a conectarla. En este caso, confirme cuándo ya no se aplican las condiciones e inicie la actualización nuevamente.
3. Aurora crea una instantánea del volumen del clúster. Supongamos que descubre problemas de compatibilidad u otros tipos de problemas una vez finalizada la actualización. O supongamos que desea realizar pruebas utilizando tanto los clústeres originales como los actualizados. En tales casos, puede restaurar a partir de esta instantánea para crear un nuevo clúster con la versión original del motor y los datos originales.

   > **Tip**
   >
   > Esta instantánea es una instantánea manual. Sin embargo, Aurora puede crearla y continuar con el proceso de actualización incluso si ha alcanzado la cuota de instantáneas manuales. Esta instantánea permanecerá permanentemente (en caso necesario) hasta que la elimine. Después de finalizar todas las pruebas posteriores a la actualización, puede eliminar esta instantánea para minimizar los cargos de almacenamiento.

4. Aurora clona el volumen del clúster. La clonación es una operación rápida que no implica copiar los datos reales de la tabla. Si Aurora encuentra un problema durante la actualización, se revierte a los datos originales del volumen del clúster clonado y vuelve a poner el clúster en línea. El volumen clonado temporal durante la actualización no está sujeto al límite habitual en el número de clones para un único volumen de clúster.
5. Aurora realiza un apagado limpio para la instancia de base de datos del escritor. Durante el apagado limpio, los eventos de progreso se registran cada 15 minutos para las siguientes operaciones. Puede examinar los eventos a medida que se producen en la página Eventos de la consola de RDS.

- Aurora purga los registros de deshacer de versiones antiguas de filas.
- Aurora revierte cualquier transacción no confirmada.

6. Aurora actualiza la versión del motor en la instancia de base de datos de escritor:
   - Aurora instala el binario para la nueva versión del motor en la instancia de base de datos del escritor.
   - Aurora utiliza la instancia de base de datos del escritor para actualizar sus datos al formato compatible con MySQL 5.7. Durante esta etapa, Aurora modifica las tablas del sistema y realiza otras conversiones que afectan a los datos del volumen del clúster. En particular, Aurora actualiza los metadatos de partición en las tablas del sistema para que sean compatibles con el formato de partición MySQL 5.7. Esta etapa puede tardar mucho tiempo si las tablas del clúster tienen un gran número de particiones.

     Si se producen errores durante esta etapa, puede encontrar los detalles en los registros de errores de MySQL. Una vez iniciada esta etapa, si el proceso de actualización falla por cualquier motivo, Aurora restaura los datos originales del volumen del clúster clonado.
7. Aurora actualiza la versión del motor en las instancias de base de datos del lector.
8. Se ha completado el proceso de actualización. Aurora registra un evento final para indicar que el proceso de actualización se completó correctamente. Ahora su clúster de base de datos está ejecutando la nueva versión principal.

Planificación de una actualización de versión principal para un clúster Aurora MySQL

Para ayudarle a decidir el momento y el método adecuados para la actualización, infórmese sobre las diferencias entre Aurora MySQL versión 3 y su entorno actual:

- Si va a convertir desde RDS para MySQL 8.0 o MySQL 8.0 Community Edition, consulte Comparación de Aurora MySQL versión 3 y MySQL 8.0 Community Edition.
- Si va a actualizar desde Aurora MySQL versión 2, RDS para MySQL 5.7 o la comunidad MySQL 5.7, consulte Comparación entre Aurora MySQL versión 2 y Aurora MySQL versión 3.
- Cree nuevas versiones compatibles con MySQL 8.0 de cualquier grupo de parámetros personalizados. Aplique los valores de parámetros personalizados necesarios a los nuevos grupos de parámetros. Consulte Cambios de parámetros para Aurora MySQL versión 3 para obtener más información sobre los cambios de parámetros.
- Revise las definiciones de objeto y el esquema de base de datos de Aurora MySQL versión 2 para ver el uso de las nuevas palabras clave reservadas introducidas en MySQL 8.0 Community Edition.

Hágalo antes de realizar la actualización. Para obtener más información, consulte la página sobre palabras clave y palabras reservadas de MySQL 8.0 en la documentación de MySQL.

También puede encontrar más consideraciones y sugerencias sobre la actualización específica de MySQL en Cambios en MySQL 8.0 en el Manual de referencia de MySQL. Por ejemplo, puede usar el comando `mysqlcheck --check-upgrade` para analizar las bases de datos Aurora MySQL existentes e identificar posibles problemas de actualización.

> **Note**
>
> Recomendamos usar clases de instancias de base de datos más grandes, al actualizar a Aurora MySQL versión 3 mediante la actualización in situ o la técnica de restauración de instantáneas. Algunos ejemplos son db.r5.24xlarge y db.r6g.16xlarge. Esto ayuda a que el proceso de actualización se complete más rápido al usar la mayor parte de la capacidad de CPU disponible en la instancia de base de datos. Puede cambiar a la clase de instancia de base de datos que desee una vez finalizada la actualización de la versión principal.

Una vez finalizada la actualización, puede seguir los procedimientos posteriores a la actualización en Limpieza posterior a la actualización para Aurora MySQL versión 3. Por último, pruebe la funcionalidad y el rendimiento de su aplicación.

Si va a convertir desde RDS para MySQL o la comunidad MySQL, siga el procedimiento de migración que se explica en Migración de datos a un clúster de base de datos de Amazon Aurora MySQL. En algunos casos, puede utilizar la replicación de registros binarios para sincronizar los datos con un clúster Aurora MySQL versión 3 como parte de la migración. Si es así, el sistema de origen debe ejecutar una versión compatible con su clúster de base de datos de destino.

Para asegurarse de que las aplicaciones y los procedimientos de administración funcionan sin problemas después de actualizar un clúster de una versión principal a otra, realice tareas de planificación y preparación con antelación. Para ver qué tipos de código de administración se deben actualizar para sus scripts de AWS CLI o aplicaciones basadas en la API de RDS, consulte Cómo afectan las actualizaciones en el lugar a los grupos de parámetros de un clúster. Consulte también Cambios en las propiedades del clúster entre versiones de Aurora MySQL.

Para obtener información sobre los problemas que pueden surgir durante la actualización, consulte Solución de problemas para la actualización Aurora MySQL en el lugar. En caso de problemas

que podrían hacer que la actualización tarde mucho tiempo, puede probar esas condiciones con antelación y corregirlas.

> **Note**
>
> Un mecanismo de actualización in situ implica apagar el clúster de base de datos mientras se lleva a cabo la operación. Aurora MySQL realiza un apagado limpio y completa las operaciones pendientes, como la depuración de deshacer. Una actualización puede tardar mucho tiempo si hay que depurar muchos registros de deshacer. Recomendamos realizar la actualización solo después de que la longitud de la lista de historial (HLL) sea baja. Un valor generalmente aceptable de HLL es de 100 000 o menos. Para obtener más información, consulte esta entrada del blog.

Simulación de la actualización mediante la clonación del clúster de base de datos

También puede comprobar los procedimientos de compatibilidad, rendimiento y mantenimiento de las aplicaciones y consideraciones similares para el clúster actualizado. Para ello, puede realizar una simulación de la actualización antes de realizar la actualización en sí. Esta técnica puede ser especialmente útil para clústeres de producción. Aquí, es importante minimizar el tiempo de inactividad y tener listo el clúster actualizado tan pronto como finalice la actualización.

Utilice los siguientes pasos:

1. Cree un clon del clúster original. Siga el procedimiento indicado en Clonación de un volumen de clúster de base de datos de Amazon Aurora.
2. Configure un conjunto similar de instancias de base de datos de escritor y lector como en el clúster original.
3. Realice una actualización en el lugar del clúster clonado. Siga el procedimiento indicado en Pasos para realizar una actualización local.

   Inicie la actualización inmediatamente después de crear el clon. De esta forma, el volumen del clúster sigue siendo idéntico al estado del clúster original. Si el clon se encuentra inactivo antes de realizar la actualización, Aurora realiza procesos de limpieza de la base de datos en segundo plano. En ese caso, la actualización del clon no es una simulación precisa de la actualización del clúster original.
4. Pruebe los procedimientos de compatibilidad, rendimiento y administración de las aplicaciones, etc., utilizando el clúster clonado.

5. Si encuentra algún problema, ajuste sus planes de actualización para que los tengan en cuenta. Por ejemplo, adapte cualquier código de aplicación para que sea compatible con el conjunto de características de la versión superior. Calcule cuánto tiempo puede tardar la actualización en función de la cantidad de datos del clúster. También puede programar la actualización para un momento en el que el clúster no esté ocupado.
6. Una vez que esté satisfecho con el funcionamiento de las aplicaciones y la carga de trabajo en el clúster de prueba, puede realizar la actualización en el lugar para el clúster de producción.
7. Esfuércese para minimizar el tiempo de inactividad total de su clúster durante una actualización de versión principal. Para ello, asegúrese de que la carga de trabajo del clúster sea baja o nula en el momento de la actualización. En particular, asegúrese de que no haya transacciones en curso durante mucho tiempo al iniciar la actualización.

Implementaciones azul/verde

En algunas situaciones, su prioridad principal es realizar un cambio inmediato del clúster antiguo a uno actualizado. En tales situaciones, puede utilizar un proceso de varios pasos que ejecuta los clústeres antiguo y nuevo en paralelo. Aquí, replicará los datos del clúster anterior al nuevo hasta que esté listo para que el nuevo clúster asuma el control. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Comprobaciones previas de actualización de versiones principales para Aurora MySQL

La actualización de MySQL de una versión principal a otra, como, por ejemplo, pasar de MySQL 5.7 a MySQL 8.0, implica algunos cambios arquitectónicos importantes que requieren una planificación y preparación minuciosas. A diferencia de las actualizaciones de versiones secundarias, que se centran principalmente en actualizar el software del motor de base de datos y, en algunos casos, las tablas del sistema, las actualizaciones principales de MySQL suelen introducir cambios fundamentales en la forma en que la base de datos almacena y administra sus metadatos.

Para ayudarle a identificar dichas incompatibilidades, al actualizar de la versión 2 de Aurora MySQL a la versión 3, Aurora ejecuta automáticamente comprobaciones de compatibilidad de actualizaciones (comprobaciones previas) para examinar los objetos del clúster de base de datos e identificar las incompatibilidades conocidas que pueden impedir que se lleve a cabo la actualización. Para obtener información detallada sobre las comprobaciones previas de Aurora MySQL, consulte Referencia de descripciones de comprobaciones previas de Aurora MySQL. Las comprobaciones previas de Aurora se ejecutan junto con las que lleva a cabo la utilidad del comprobador de actualización de Community MySQL.

Estas comprobaciones previas son obligatorias. No tiene la opción de omitirlas. Las comprobaciones previas proporcionan las siguientes ventajas:

- Pueden reducir la posibilidad de que se produzcan errores de actualización que puedan provocar un tiempo de inactividad prolongado.
- Si hay incompatibilidades, Amazon Aurora impedirá que se lleve a cabo la actualización y proporcionará un registro para que obtenga más información sobre ellas. Luego podrá usar el registro para preparar la base de datos para la actualización a la versión 3 y resolver así las incompatibilidades. Para obtener información detallada sobre cómo resolver incompatibilidades, consulte la sección Preparing your installation for upgrade en la documentación de MySQL y la sección Upgrading to MySQL 8.0? Here is what you need to know... en el blog de MySQL Server.

  Para obtener más información acerca de cómo actualizar a MySQL 8.0, consulte la sección sobre la actualización de MySQL en la documentación de MySQL.

Las comprobaciones previas se ejecutan antes de que el clúster de base de datos se desconecte para la actualización de la versión principal. Si las verificaciones previas encuentran una incompatibilidad, Aurora cancela automáticamente la actualización antes de detenerse la instancia de base de datos. Aurora también genera un evento por la incompatibilidad. Para obtener más

información acerca de los eventos de Amazon Aurora, consulte Uso de notificaciones de eventos de Amazon RDS.

Una vez completadas las comprobaciones previas, Aurora registra información detallada sobre cada incompatibilidad en el archivo `upgrade-prechecks.log`. En la mayoría de los casos, la entrada de registro incluye un vínculo a la documentación de SQL para corregir la incompatibilidad. Para obtener más información acerca de cómo visualizar los archivos de registro, consulte Visualización y descripción de archivos de registro de base de datos.

> **Note**
>
> Debido a la naturaleza de las comprobaciones previa, analizan objetos en su base de datos. Este análisis genera un consumo de recursos y aumenta el tiempo de ejecución de la actualización. Para obtener más información sobre los aspectos a tener en cuenta con respecto al rendimiento de las comprobaciones previas, consulte Proceso de comprobación previa de Aurora MySQL.

Contenido

- Proceso de comprobación previa de Aurora MySQL
- Formato de registro de comprobación previa de Aurora MySQL
- Ejemplos de resultados del registro de comprobación previa de Aurora MySQL
- Rendimiento de la comprobación previa de Aurora MySQL
- Resumen de comprobaciones previas de actualizaciones de Community MySQL
- Resumen de comprobaciones previas de actualizaciones de Aurora MySQL
- Referencia de descripciones de comprobaciones previas de Aurora MySQL
  - Errores
    - Comprobaciones previas de MySQL que informan de errores
    - Comprobaciones previas de Aurora MySQL que informan de errores
  - Advertencias
    - Comprobaciones previas de MySQL que informan de advertencias
    - Comprobaciones previas de Aurora MySQL que informan de advertencias
  - Avisos
  - Errores, advertencias o avisos

Proceso de comprobación previa de Aurora MySQL

Tal y como se ha indicado anteriormente, el proceso de actualización de Aurora MySQL implica la ejecución de comprobaciones de compatibilidad (comprobaciones previas) en la base de datos antes de continuar con la actualización de la versión principal.

En el caso de las actualizaciones locales, las comprobaciones previas se ejecutan en la instancia de base de datos de escritor mientras esté en línea. Si la comprobación previa se realiza correctamente, se llevará a cabo la actualización. Si se encuentran errores, se registrarán en el archivo `upgrade-prechecks.log` y se cancelará la actualización. Antes de volver a intentar la actualización, resuelva los errores que aparezcan en el archivo `upgrade-prechecks.log`.

En el caso de las actualizaciones de restauración de instantánea, la comprobación previa se ejecuta durante el proceso de restauración. Si se realiza correctamente, la base de datos se actualizará a la nueva versión de Aurora MySQL. Si se encuentran errores, se registrarán en el archivo `upgrade-prechecks.log` y se cancelará la actualización. Antes de volver a intentar la actualización, resuelva los errores que aparezcan en el archivo `upgrade-prechecks.log`.

Para obtener más información, consulte Determinación del motivo de los errores en una actualización de la versión principal de Aurora MySQL y Referencia de descripciones de comprobaciones previas de Aurora MySQL.

Para supervisar el estado de la comprobación previa, puede ver los siguientes eventos en el clúster de la base de datos.

Estado de la comprobación previa	Mensaje de evento	Acción
Started	Preparación de la actualización en curso: se están iniciando las comprobaciones previas de la actualización en línea.	Ninguna
Con error	El clúster de la base de datos tiene un estado que no se puede actualizar: se ha producido un error en las	Revise el archivo `upgrade-prechecks.log` por si tiene errores. Corrija los errores.

Estado de la comprobación previa	Mensaje de evento	Acción
	comprobaciones previas de la actualización. Para obtener más información, consulte el archivo upgrade-prechecks.log. Para obtener más información sobre cómo solucionar la causa del error de actualización, consulte https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Upgrading.Troubleshooting.html	Vuelva a intentar la actualización.
Succeeded	Preparación de la actualización en curso: se han completado las comprobaciones previas de la actualización en línea.	Se ha realizado correctamente la comprobación previa y no se ha devuelto ningún error. Revise el archivo `upgrade-prechecks.log` por si hay advertencias y avisos.

Para obtener más información sobre la visualización de eventos, consulte Consulta de eventos de Amazon RDS.

Formato de registro de comprobación previa de Aurora MySQL

Una vez finalizadas las comprobaciones de compatibilidad de las actualizaciones (comprobaciones previas), puede revisar el archivo `upgrade-prechecks.log`. En el archivo de registro se incluyen los resultados, los objetos afectados y la información de corrección de cada comprobación previa.

Los errores bloquean la actualización. Debe resolverlos antes de volver a intentar la actualización.

Las advertencias y los avisos son menos importantes, pero le recomendamos que los revise detenidamente para asegurarse de que no haya problemas de compatibilidad con la carga de trabajo de la aplicación. Resuelva de inmediato cualquier problema que se detecte.

El archivo de registro tiene el formato siguiente:

- `targetVersion`: la versión compatible con MySQL de la actualización de Aurora MySQL.
- `auroraServerVersion`: la versión de Aurora MySQL en la que se ha ejecutado la comprobación previa.
- `auroraTargetVersion`: la versión de Aurora MySQL a la que está actualizando.
- `checksPerformed`: incluye la lista de comprobaciones previas realizadas.
- `id`: el nombre de la comprobación previa que se está ejecutando.
- `title`: una descripción de la comprobación previa que se está ejecutando.
- `status`: esto no indica si la comprobación previa se ha realizado correctamente o si se ha producido un error en ella, pero muestra el estado de la consulta de comprobación previa:
  - `OK`: la consulta de comprobación previa se ha ejecutado y completado correctamente.
  - `ERROR`: no se ha podido ejecutar la consulta de comprobación previa. Esto puede ocurrir debido a problemas, como, por ejemplo, las limitaciones de recursos, los reinicios inesperados de la instancia o la interrupción de la consulta de comprobación previa de la compatibilidad.

    Para obtener más información, consulte este ejemplo.
- `description`: una descripción general de la incompatibilidad y cómo solucionar el problema.
- `documentationLink`: si procede, aquí encontrará un enlace a la documentación pertinente de Aurora MySQL o MySQL. Para obtener más información, consulte Referencia de descripciones de comprobaciones previas de Aurora MySQL.
- `detectedProblems`: si la comprobación previa devuelve un error, una advertencia o un aviso, se mostrarán los detalles de la incompatibilidad y los objetos incompatibles, si procede:
  - `level`: el nivel de incompatibilidad detectado por la comprobación previa. A continuación, se muestran los niveles válidos:
    - `Error`: no se puede continuar con la actualización hasta que se resuelva la incompatibilidad.
    - `Warning`: se puede continuar con la actualización, pero se ha detectado un objeto, una sintaxis o una configuración obsoletos. Revise detenidamente las advertencias y resuélvalas de inmediato para evitar problemas en futuras versiones.

    - `Notice`: se puede continuar con la actualización, pero se ha detectado un objeto, una sintaxis o una configuración obsoletos. Revise detenidamente los avisos y resuélvalos de inmediato para evitar problemas en futuras versiones.
  - `dbObject`: el nombre del objeto de la base de datos en el que se ha detectado la incompatibilidad.
  - `description`: una descripción detallada de la incompatibilidad y cómo solucionar el problema.
- `errorCount`: el número de errores de incompatibilidad detectados. Bloquean la actualización.
- `warningCount`: el número de advertencias de incompatibilidad detectadas. No bloquean la actualización, pero resuélvalas de inmediato para evitar problemas en futuras versiones.
- `noticeCount`: el número de avisos de incompatibilidad detectados. No bloquean la actualización; resuélvalos de inmediato para evitar problemas en futuras versiones.
- `Summary`: un resumen del número de errores, advertencias y avisos de compatibilidad de las comprobaciones previas.

Ejemplos de resultados del registro de comprobación previa de Aurora MySQL

En los siguientes ejemplos se muestra el resultado del registro de comprobación previa que puede ver. Para obtener más información sobre las comprobaciones previas que se ejecutan, consulte Referencia de descripciones de comprobaciones previas de Aurora MySQL.

Estado correcto de la comprobación previa; no se ha detectado ninguna incompatibilidad

La consulta de comprobación previa se ha completado correctamente. No se ha detectado ninguna incompatibilidad.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinytext",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny text columns",
  "status": "OK",
  "detectedProblems": []
},
```

Estado correcto de la comprobación previa; se ha detectado un error

La consulta de comprobación previa se ha completado correctamente. Se ha detectado un error.

```
{
```

```
 "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
 "title": "Check for geometry columns on prefix indexes",
 "status": "OK",
 "description": "Consider dropping the prefix indexes of geometry columns and
restart the upgrade.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test25.sbtest1",
       "description": "Table `test25`.`sbtest1` has an index `idx_t1` on geometry
column/s. Mysql 8.0 does not support this type of index on a geometry column
https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run 'DROP
INDEX `idx_t1` ON `test25`.`sbtest1`;"
     },
}
```

Estado correcto de la comprobación previa; se ha detectado una advertencia

Se pueden devolver advertencias cuando una comprobación previa se ha realizado correctamente o se ha producido un error en ella.

En este caso, la consulta de comprobación previa se ha completado correctamente. Se han detectado dos advertencias.

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
```

```
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
    ]
}
```

ERROR en el estado de la comprobación previa; no se ha registrado ninguna incompatibilidad

Se ha producido un error en la consulta de comprobación previa, por lo que no se han podido comprobar las incompatibilidades.

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "ERROR",
  "description": "Can't connect to MySQL server on 'localhost:3306' (111) at
 13/08/2024 12:22:20 UTC. This failure can occur due to low memory available on the
 instance for executing upgrade prechecks. Please check 'FreeableMemory' Cloudwatch
 metric to verify the available memory on the instance while executing prechecks. If
 instance ran out of memory, we recommend to retry the upgrade on a higher instance
 class."
}
```

Este error puede producirse debido a un reinicio inesperado de la instancia o a la interrupción de una consulta de comprobación previa de compatibilidad en la base de datos durante la ejecución. Por ejemplo, en clases de instancias de base de datos más pequeñas, es posible que esto se produzca cuando se agote la memoria disponible en la instancia.

Puede usar la métrica `FreeableMemory` de Amazon CloudWatch para verificar la memoria disponible en la instancia mientras se ejecutan las comprobaciones previas. Si la instancia se ha quedado sin memoria, le recomendamos que vuelva a intentar la actualización en una clase de instancia de base de datos más grande. En algunos casos, puede utilizar una implementación azul/verde, lo que permite que las comprobaciones previas y las actualizaciones se ejecuten en el clúster de base de datos “verde” con independencia de la carga de trabajo de producción, lo que también consume recursos del sistema.

Para obtener más información, consulte Solución de problemas de uso de memoria de bases de datos Aurora MySQL.

Resumen de la comprobación previa: se han detectado un error y tres advertencias

Las comprobaciones previas de compatibilidad también incluyen información sobre las versiones de Aurora MySQL de origen y destino, así como un resumen del número de errores, advertencias y avisos, al final del resultado de la comprobación previa.

Por ejemplo, en el siguiente resultado se indica que se ha intentado actualizar de Aurora MySQL 2.11.6 a Aurora MySQL 3.07.1. La actualización ha devuelto un error, tres advertencias y ningún aviso. Dado que las actualizaciones no pueden continuar cuando se devuelve un error, debe resolver el problema de compatibilidad routineSyntaxCheck y volver a intentar la actualización.

```
{
  "serverAddress": "/tmp%2Fmysql.sock",
  "serverVersion": "5.7.12 - MySQL Community Server (GPL)",
  "targetVersion": "8.0.36",
  "auroraServerVersion": "2.11.6",
  "auroraTargetVersion": "3.07.1",
  "outfilePath": "/rdsdbdata/tmp/PreChecker.log",
  "checksPerformed": [{
      ... output for each individual precheck ...
      .
      .
      {
        "id": "oldTemporalCheck",
        "title": "Usage of old temporal type",
        "status": "OK",
          "detectedProblems": []
      },
      {
        "id": "routinesSyntaxCheck",
        "title": "MySQL 8.0 syntax check for routine-like objects",
        "status": "OK",
        "description": "The following objects did not pass a syntax check with
 the latest MySQL 8.0 grammar. A common reason is that they reference names that
 conflict with new reserved keywords. You must update these routine definitions and
 `quote` any such references before upgrading.",
        "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
        "detectedProblems": [{
            "level": "Error",
            "dbObject": "test.select_res_word",
            "description": "at line 2,18: unexpected token 'except'"
        }]
```

```
      },
      .
      .
      .
      {
        "id": "zeroDatesCheck",
        "title": "Zero Date, Datetime, and Timestamp values",
        "status": "OK",
        "description": "Warning: By default zero date/datetime/timestamp values
 are no longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are
 included in SQL_MODE by default. These modes should be used with strict mode as
 they will be merged with strict mode in a future release. If you do not include
 these modes in your SQL_MODE setting, you are able to insert date/datetime/
timestamp values that contain zeros. It is strongly advised to replace zero values
 with valid ones, as they may not work correctly in the future.",
        "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
        "detectedProblems": [{
            "level": "Warning",
            "dbObject": "global.sql_mode",
            "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
            },
            {
            "level": "Warning",
            "dbObject": "session.sql_mode",
            "description": " of 8 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
            }
          ]
       },
       .
       .
       .
  }],
  "errorCount": 1,
  "warningCount": 3,
  "noticeCount": 0,
  "Summary": "1 errors were found. Please correct these issues before upgrading to
 avoid compatibility issues."
}
```

Rendimiento de la comprobación previa de Aurora MySQL

Las comprobaciones previas de compatibilidad se ejecutan antes de que se desconecte la instancia de base de datos para la actualización, de modo que, en circunstancias normales, no provocan tiempos de inactividad en la instancia de base de datos al ejecutarse. Sin embargo, pueden afectar a la carga de trabajo de la aplicación que se ejecuta en la instancia de base de datos del escritor. Las comprobaciones previas acceden al diccionario de datos a través de las tablas de information_schema, lo que puede ser lento si hay muchos objetos de base de datos. Tenga en cuenta los siguientes factores:

- La duración de la comprobación previa varía en función del número de objetos de la base de datos, como, por ejemplo, tablas, columnas, rutinas y limitaciones. Los clústeres de base de datos con un gran número de objetos pueden tardar más en ejecutarse.

  Por ejemplo, removedFunctionsCheck puede tardar más y utilizar más recursos en función del número de objetos almacenados.
- En el caso de las actualizaciones locales, el uso de una clase de instancia de base de datos mayor (por ejemplo, db.r5.24xlarge o db.r6g.16xlarge) puede ayudar a que la actualización se complete de forma más rápida al utilizar más CPU. Puede reducir el tamaño después de la actualización.
- Las consultas en `information_schema` en varias bases de datos pueden ser lentas, especialmente si hay muchos objetos y en instancias de bases de datos más pequeñas. En tales casos, plantéese la posibilidad de utilizar la clonación, la restauración de instantáneas o una implementación azul/verde para las actualizaciones.
- El uso de recursos de comprobación previa (CPU, memoria) puede aumentar con más objetos, lo que se traduce en tiempos de ejecución más prolongados en instancias de base de datos más pequeñas. En tales casos, plantéese la posibilidad de probar la clonación, la restauración de instantáneas o una implementación azul/verde para las actualizaciones.

  Si se produce un error en las comprobaciones previas por falta de recursos, puede detectarlo en el registro de comprobaciones previas mediante el resultado de estado:

```
"status": "ERROR",
```

Para obtener más información, consulte Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL y Planificación de una actualización de versión principal para un clúster Aurora MySQL.

Resumen de comprobaciones previas de actualizaciones de Community MySQL

A continuación, se muestra una lista general de incompatibilidades entre MySQL 5.7 y 8.0:

- Su clúster de base de datos compatible con MySQL 5.7 no puede usar características que no sean compatibles con MySQL 8.0.

  Para obtener más información, consulte la sección sobre características eliminadas en MySQL 8.0, en la documentación de MySQL.
- No debe haber ninguna infracción de la palabra clave ni de la palabra reservada. Es posible que en MySQL 8.0 haya algunas palabras clave reservadas que no estaban reservadas previamente.

  Para obtener más información, consulte la página sobre Palabras clave y palabras reservadas en la documentación de MySQL.
- Para mejorar la compatibilidad de Unicode, piense en convertir objetos que usen el conjunto de caracteres `utf8mb3` para usar el conjunto de caracteres `utf8mb4`. El conjunto de caracteres `utf8mb3` ha quedado obsoleto. Asimismo, piense en utilizar `utf8mb4` para referencias de conjuntos de caracteres, en vez de utilizar `utf8`, ya que actualmente `utf8` es un alias del conjunto de caracteres `utf8mb3`.

  Para obtener más información, consulte la sección sobre el conjunto de caracteres utf8mb3 (codificación Unicode UTF-8 de 3 bytes) en la documentación de MySQL.
- No debe haber tablas InnoDB con un formato de fila que no sea el predeterminado.
- No debe haber atributos de tipo de longitud `ZEROFILL` o `display`.
- Ninguna tabla particionada debe usar un motor de almacenamiento que no tenga soporte de particiones nativo.
- No tiene que haber tablas en la base de datos del sistema `mysql` de MySQL 5.7 que tengan el mismo nombre que una tabla usada por el diccionario de datos de MySQL 8.0.
- Ninguna tabla debe usar tipos o funciones de datos obsoletos.
- No tiene que haber nombres de restricción de clave externa que superen los 64 caracteres.
- No tiene que haber modos SQL obsoletos en su configuración de variable del sistema `sql_mode`.
- No tiene que haber tablas ni procedimientos almacenados que tengan elementos de columna `ENUM` o `SET` individuales que superen los 255 caracteres de longitud.
- Ninguna partición de tabla debe residir en espacios de tablas InnoDB compartidos.
- No debe haber referencias circulares en las rutas de los archivos de datos de los espacios de tablas.

- No tiene que haber consultas ni definiciones de programas almacenadas que usen calificadores `ASC` o `DESC` para cláusulas `GROUP BY`.
- No debe haber ninguna variable de sistema eliminada y las variables de sistema deben usar los nuevos valores predeterminados para MySQL 8.0.
- No debe haber valores de fecha, fecha y hora o marca temporal que sean cero (`0`).
- No debe haber inconsistencias en el esquema causadas por la eliminación o la corrupción de un archivo.
- No debe haber nombres de tablas que contengan la cadena de caracteres `FTS`.
- No debe haber tablas InnoDB que pertenezcan a un motor diferente.
- No debe haber nombres de tablas o esquemas que no sean válidos para MySQL 5.7.

Para obtener más información sobre las comprobaciones previas que se ejecutan, consulte Referencia de descripciones de comprobaciones previas de Aurora MySQL.

Para obtener más información acerca de cómo actualizar a MySQL 8.0, consulte la sección sobre la actualización de MySQL en la documentación de MySQL. Para obtener una descripción general de los cambios en MySQL 8.0, consulte la sección What is new in MySQL 8.0 en la documentación de MySQL.

Resumen de comprobaciones previas de actualizaciones de Aurora MySQL

Aurora MySQL tiene sus propios requisitos específicos al actualizar de la versión 2 a la versión 3, entre los que se incluyen los siguientes:

- No debe haber ninguna sintaxis SQL obsoleta, como `SQL_CACHE`, `SQL_NO_CACHE` y `QUERY_CACHE`, en las vistas, las rutinas, los disparadores y los eventos.
- No debe haber ninguna columna `FTS_DOC_ID` en ninguna tabla sin el índice `FTS`.
- No debe haber ninguna discrepancia en la definición de columna entre el diccionario de datos de InnoDB y la definición real de la tabla.
- Todos los nombres de bases de datos y tablas deben estar en minúscula cuando el parámetro `lower_case_table_names` esté configurado como `1`.
- No debe haber un definidor vacío ni faltar un definidor en los eventos y disparadores, y el contexto de creación de los disparadores tiene que ser válido.
- Todos los nombres de desencadenadores de una base de datos deben ser únicos.

- La recuperación de DDL y la DDL rápida no son compatibles con la versión 3 de Aurora MySQL. No debe haber artefactos en las bases de datos relacionados con estas características.
- Las tablas con el formato de fila `REDUNDANT` o `COMPACT` no pueden tener índices de más de 767 bytes.
- La longitud del prefijo de los índices definidos en las columnas de texto `tiny` no puede superar los 255 bytes. Con el conjunto de caracteres `utf8mb4` configurado, esto limita la longitud de prefijo admitida a 63 caracteres.

  Se permitía una longitud de prefijo mayor en MySQL 5.7 mediante el parámetro `innodb_large_prefix`. Este parámetro ha quedado obsoleto en MySQL 8.0.
- No debe haber ninguna incoherencia en los metadatos de InnoDB en la tabla `mysql.host`.
- No debe haber ninguna discrepancia en los tipos de datos de las columnas en las tablas del sistema.
- No debe haber transacciones XA en el estado `prepared`.
- Los nombres de las columnas de las vistas no pueden tener más de 64 caracteres.
- Los caracteres especiales de los procedimientos almacenados no pueden ser incoherentes.
- Las tablas no pueden tener incoherencias en las rutas de los archivos de datos.

Para obtener más información sobre las comprobaciones previas que se ejecutan, consulte Referencia de descripciones de comprobaciones previas de Aurora MySQL.

Referencia de descripciones de comprobaciones previas de Aurora MySQL

A continuación, se describen en detalle las comprobaciones previas para la actualización de Aurora MySQL.

Contenido

- Errores
  - Comprobaciones previas de MySQL que informan de errores
  - Comprobaciones previas de Aurora MySQL que informan de errores
- Advertencias
  - Comprobaciones previas de MySQL que informan de advertencias
  - Comprobaciones previas de Aurora MySQL que informan de advertencias
- Avisos

- Errores, advertencias o avisos

Errores

Las siguientes comprobaciones previas generan errores cuando fallan y no se puede continuar con la actualización.

Temas

- Comprobaciones previas de MySQL que informan de errores
- Comprobaciones previas de Aurora MySQL que informan de errores

Comprobaciones previas de MySQL que informan de errores

Las siguientes comprobaciones previas proceden de Community MySQL:

- checkTableMysqlSchema
- circularDirectoryCheck
- columnsWhichCannotHaveDefaultsCheck
- depreciatedSyntaxCheck
- engineMixupCheck
- enumSetElementLengthCheck
- foreignKeyLengthCheck
- getDuplicateTriggers
- getEventsWithNullDefiner
- getMismatchedMetadata
- getTriggersWithNullDefiner
- getValueOfVariablelower_case_table_names
- groupByAscSyntaxCheck
- mysqlEmptyDotTableSyntaxCheck
- mysqlIndexTooLargeCheck
- mysqlInvalid57NamesCheck
- mysqlOrphanedRoutinesCheck

- mysqlSchemaCheck
- nonNativePartitioningCheck
- oldTemporalCheck
- partitionedTablesInSharedTablespaceCheck
- removedFunctionsCheck
- routineSyntaxCheck
- schemaInconsistencyCheck

checkTableMysqlSchema

Nivel de comprobación previa: error

Problemas notificados por el comando **`check table x for upgrade`** del esquema de **`mysql`**

Antes de iniciar la actualización a la versión 3 de Aurora MySQL, se ejecuta `check table for upgrade` en cada tabla del esquema de `mysql` de la instancia de la base de datos. El comando `check table for upgrade` examina las tablas para detectar posibles problemas que puedan surgir durante una actualización a una versión más reciente de MySQL. Ejecutar este comando antes de intentar una actualización puede ayudar a identificar y resolver cualquier incompatibilidad con antelación, lo que facilita el proceso de actualización propiamente dicho.

Este comando realiza varias comprobaciones en cada tabla, como, por ejemplo:

- Verifica que los metadatos y la estructura de la tabla sean compatibles con la versión de MySQL de destino
- Comprueba si hay alguna característica obsoleta o eliminada que se utilice en la tabla
- Garantiza que la tabla se pueda actualizar correctamente sin que se pierdan datos

Para obtener más información, consulte la sección CHECK TABLE statement en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "checkTableMysqlSchema",
  "title": "Issues reported by 'check table x for upgrade' command for mysql
 schema.",
```

```
  "status": "OK",
  "detectedProblems": []
}
```

El resultado de esta comprobación previa depende del error detectado y de cuándo se produzca, ya que `check table for upgrade` realiza varias comprobaciones.

Si detecta algún error con esta comprobación previa, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

circularDirectoryCheck

Nivel de comprobación previa: error

Referencias circulares a directorios en las rutas de los archivos de datos de los espacios de tablas

A partir de la versión 8.0.17 de MySQL, la cláusula `CREATE TABLESPACE ... ADD DATAFILE` ya no permite referencias circulares a directorios. Para evitar problemas de actualización, elimine cualquier referencia circular a directorios de las rutas de los archivos de datos de los espacios de tablas antes de actualizar a la versión 3 de Aurora MySQL.

Ejemplo de salida:

```
{
  "id": "circularDirectory",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "description": "Error: Following tablespaces contain circular directory references
 (e.g. the reference '/../') in data file paths which as of MySQL 8.0.17 are not
 permitted by the CREATE TABLESPACE ... ADD DATAFILE clause. An exception to the
 restriction exists on Linux, where a circular directory reference is permitted if
 the preceding directory is a symbolic link. To avoid upgrade issues, remove any
 circular directory references from tablespace data file paths before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-innodb-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "ts2",
```

```
        "description": "circular reference in datafile path: '/home/ec2-user/dbdata/
mysql_5_7_44/../ts2.ibd'",
        "dbObjectType": "Tablespace"
      }
  ]
}
```

Si recibe este error, vuelva a crear las tablas con un espacio de tabla file-per-table. Utilice las rutas de archivos predeterminadas para todas las definiciones de tablas y espacios de tablas.

> **Note**
>
> Aurora MySQL no admite los comandos `CREATE TABLESPACE` ni los espacios de tablas. Antes de volver a crear los espacios de tablas, consulte la sección Online DDL operations en la documentación de MySQL para conocer los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

Una vez que los haya creado de nuevo, se aprueba la comprobación previa, lo que permite continuar con la actualización.

```
{
  "id": "circularDirectoryCheck",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "detectedProblems": []
},
```

columnsWhichCannotHaveDefaultsCheck

Nivel de comprobación previa: error

Columnas que no pueden tener valores predeterminados

En las versiones anteriores a MySQL 8.0.13, las columnas `BLOB`, `TEXT`, `GEOMETRY` y `JSON` no pueden tener valores predeterminados. Elimine las cláusulas predeterminadas en estas columnas antes de actualizar a la versión 3 de Aurora MySQL. Para obtener más información sobre los cambios en la gestión predeterminada de estos tipos de datos, consulte Data type default values en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "description": "Error: The following columns are defined as either BLOB, TEXT,
 GEOMETRY or JSON and have a default value set. These data types cannot have default
 values in MySQL versions prior to 8.0.13, while starting with 8.0.13, the default
 value must be specified as an expression. In order to fix this issue, please use
 the ALTER TABLE ... ALTER COLUMN ... DROP DEFAULT statement.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/data-type-
defaults.html#data-type-defaults-explicit",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test_blob_default.geo_col",
        "description": "geometry"
      }
  ]
},
```

La comprobación previa devuelve un error porque la columna `geo_col` de la tabla `test.test_blob_default` utiliza un tipo de datos `BLOB`, `TEXT`, `GEOMETRY` o `JSON` con un valor predeterminado especificado.

Si observamos la definición de la tabla, podemos ver que la columna `geo_col` se ha definido como `geo_col geometry NOT NULL default ''`.

```
mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Es necesario eliminar esta cláusula predeterminada para permitir que se apruebe la comprobación previa.

> **Note**
>
> Antes de ejecutar instrucciones `ALTER TABLE` o volver a crear espacios de tablas, consulte la sección Online DDL operations en la documentación de MySQL para

comprender los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

```
mysql> ALTER TABLE test_blob_default modify COLUMN geo_col geometry NOT NULL;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

La comprobación previa se aprueba y puede volver a intentar la actualización.

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "detectedProblems": []
},
```

depreciatedSyntaxCheck

Nivel de comprobación previa: error

Uso de palabras clave obsoletas en la definición

MySQL 8.0 ha eliminado la caché de consultas. Como resultado, se ha eliminado parte de la sintaxis de SQL específica de la caché de consultas. Si alguno de los objetos de la base de datos incluye las palabras clave `QUERY CACHE`, `SQL_CACHE` o `SQL_NO_CACHE`, se devolverá un error de comprobación previa. Para resolver este problema, vuelva a crear estos objetos. Para ello, elimine las palabras clave mencionadas.

Ejemplo de salida:

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
```

```
  "status": "OK",
  "description": "Error: The following DB objects contain keywords like 'QUERY
 CACHE', 'SQL_CACHE', 'SQL_NO_CACHE' which are not supported in major version 8.0.
 It is recommended to drop these DB objects or rebuild without any of the above
 keywords before upgrade.",
  "detectedProblems": [
      {
"level": "Error",
"dbObject": "test.no_query_cache_check",
"description": "PROCEDURE uses depreciated words in definition"
      }
  ]
}
```

La comprobación previa indica que el procedimiento almacenado de `test.no_query_cache_check` utiliza una de las palabras clave eliminadas. Al observar la definición del procedimiento, podemos ver que utiliza `SQL_NO_CACHE`.

```
mysql> show create procedure test.no_query_cache_check\G
*************************** 1. row ***************************
           Procedure: no_query_cache_check
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `no_query_cache_check`()
BEGIN
    SELECT SQL_NO_CACHE k from sbtest1 where id > 10 and id < 20 group by k asc;
END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Elimine la palabra clave.

```
mysql> drop procedure test.no_query_cache_check;
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter //

mysql> CREATE DEFINER=`reinvent`@`%` PROCEDURE `no_query_cache_check`() BEGIN
 SELECT k from sbtest1 where id > 10 and id < 20 group by k asc; END//
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> delimiter ;
```

Tras eliminar la palabra clave, la comprobación previa se completa correctamente.

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "detectedProblems": []
}
```

engineMixupCheck

Nivel de comprobación previa: error

Tablas reconocidas por InnoDB que pertenecen a otro motor

Al igual que en schemaInconsistencyCheck, esta comprobación previa verifica que los metadatos de la tabla de MySQL son coherentes antes de continuar con la actualización.

Si detecta algún error con esta comprobación previa, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

Ejemplo de salida:

```
{
  "id": "engineMixupCheck",
  "title": "Tables recognized by InnoDB that belong to a different engine",
  "status": "OK",
  "description": "Error: Following tables are recognized by InnoDB engine while the
 SQL layer believes they belong to a different engine. Such situation may happen
 when one removes InnoDB table files manually from the disk and creates e.g. a
 MyISAM table with the same name.\n\nA possible way to solve this situation is to
 e.g. in case of MyISAM table:\n\n1. Rename the MyISAM table to a temporary name
 (RENAME TABLE).\n2. Create some dummy InnoDB table (its definition does not need to
 match), then copy (copy, not move) and rename the dummy .frm and .ibd files to the
 orphan name using OS file commands.\n3. The orphan table can be then dropped (DROP
 TABLE), as well as the dummy table.\n4. Finally the MyISAM table can be renamed
 back to its original name.",
  "detectedProblems": [
```

```
      {
        "level": "Error",
        "dbObject": "mysql.general_log_backup",
        "description": "recognized by the InnoDB engine but belongs to CSV"
      }
  ]
}
```

enumSetElementLengthCheck

Nivel de comprobación previa: error

Definiciones de las columnas **`ENUM`** y **`SET`** que incluyen elementos que tienen más de 255 caracteres

Las tablas y los procedimientos almacenados no deben tener elementos de columna `ENUM` o `SET` que superen los 255 caracteres o 1020 bytes. En las versiones anteriores a MySQL 8.0, la longitud máxima combinada era de 64 K, pero la versión 8.0 limita los elementos individuales a 255 caracteres o 1020 bytes (en el caso de admitir caracteres multibyte). Si se produce un error en la comprobación previa para `enumSetElementLengthCheck`, modifique los elementos que superen estos nuevos límites antes de volver a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "enumSetElementLengthCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "description": "Error: The following columns are defined as either ENUM or SET and
 contain at least one element longer that 255 characters. They need to be altered so
 that all elements fit into the 255 characters limit.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/string-type-
overview.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.large_set.s",
        "description": "SET contains element longer than 255 characters"
      }
  ]
},
```

La comprobación previa indica un error porque la columna `s` de la tabla `test.large_set` incluye un elemento SET de más de 255 caracteres.

Tras reducir el tamaño de SET de esta columna, se aprueba la comprobación previa, lo que permite continuar con la actualización.

```
{
  "id": "enumSetElementLenghtCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "detectedProblems": []
},
```

foreignKeyLengthCheck

Nivel de comprobación previa: error

Nombres de restricción de clave externa que superen los 64 caracteres

En MySQL, la longitud de los identificadores está limitada a 64 caracteres, tal y como se describe en la documentación de MySQL. Debido a problemas detectados en los que la longitud de las claves externas podía ser igual o superior a este valor, lo que provocaba errores en la actualización, se ha implementado esta comprobación previa. Si detecta errores con esta comprobación previa, debe modificar la restricción o cambiarle el nombre para que tenga menos de 64 caracteres antes de volver a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "foreignKeyLength",
  "title": "Foreign key constraint names longer than 64 characters",
  "status": "OK",
  "detectedProblems": []
}
```

getDuplicateTriggers

Nivel de comprobación previa: error

Todos los nombres de desencadenadores de una base de datos deben ser únicos.

Debido a los cambios en la implementación del diccionario de datos, MySQL 8.0 no admite desencadenadores que distingan entre mayúsculas y minúsculas en una base de datos. Esta comprobación previa valida que el clúster de base de datos no tenga una o varias bases de datos que incluyan desencadenadores duplicados. Para obtener más información, consulte la sección Identifier case sensitivity en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "description": "Error: You have one or more database containing duplicate
 triggers. Mysql 8.0 does not support case sensitive triggers within a database
 https://dev.mysql.com/doc/refman/8.0/en/identifier-case-sensitivity.html. To
 upgrade to MySQL 8.0, drop the triggers with case-insensitive duplicate names and
 recreate with distinct names.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_product"
      },
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_PRODUCT"
      }
  ]
}
```

La comprobación previa indica que hay un error en el clúster de base de datos, ya que tiene dos desencadenadores con el mismo nombre, pero se diferencian en el uso de mayúsculas y minúsculas: `test.before_insert_product` y `test.before_insert_PRODUCT`.

Antes de realizar la actualización, cambie el nombre de los desencadenadores o elimínelos y vuelva a crearlos con un nombre nuevo.

Tras cambiarle el nombre de `test.before_insert_PRODUCT` a `test.before_insert_product_2`, la comprobación previa se realiza correctamente.

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "detectedProblems": []
}
```

getEventsWithNullDefiner

Nivel de comprobación previa: error

La columna de definidor de **`mysql.event`** no puede ser nula ni estar vacía.

El atributo `DEFINER` especifica la cuenta de MySQL que posee una definición de objeto almacenado, como, por ejemplo, un desencadenador, un procedimiento almacenado o un evento. Este atributo es particularmente útil en situaciones en las que se desea controlar el contexto de seguridad en el que se ejecuta el objeto almacenado. Al crear un objeto almacenado, si no se especifica un atributo `DEFINER`, el valor predeterminado será el usuario que ha creado el objeto.

Al actualizar a MySQL 8.0, no puede tener ningún objeto almacenado que tenga un definidor `null` o que esté vacío en el diccionario de datos de MySQL. Si tiene esos objetos almacenados, se generará un error de comprobación previa. Debe solucionarlo antes de continuar con la actualización.

Ejemplo de error:

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "description": "Error: Set definer column in mysql.event to a valid non-null
 definer.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.get_version",
        "description": "Set definer for event get_version in Schema test"
      }
  ]
}
```

La comprobación previa devuelve un error para el evento `test.get_version`, ya que tiene un definidor `null`.

Para resolver este problema, puede comprobar la definición del evento. Como puede ver, el definidor es `null` o que esté vacío.

```
mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+---------+
| db   | name        | definer |
+------+-------------+---------+
| test | get_version |         |
+------+-------------+---------+
1 row in set (0.00 sec)
```

Elimine o vuelva a crear el evento con un definidor válido.

> **Note**
>
> Antes de eliminar o volver a definir un `DEFINER`, revise y compruebe detenidamente los requisitos de aplicación y privilegios. Para obtener más información, consulte la sección Stored object access control en la documentación de MySQL.

```
mysql> drop event test.get_version;
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> delimiter $$
mysql> CREATE EVENT get_version
    ->     ON SCHEDULE
    ->       EVERY 1 DAY
    ->     DO
    ->      ///DO SOMETHING //
    -> $$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+------------+
| db   | name        | definer    |
```

```
+------+-------------+------------+
| test | get_version | reinvent@% |
+------+-------------+------------+
1 row in set (0.00 sec)
```

Ahora se aprueba la comprobación previa.

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "detectedProblems": []},
```

getMismatchedMetadata

Nivel de comprobación previa: error

Discrepancia en la definición de columna entre el diccionario de datos de InnoDB y la definición real de la tabla

Al igual que en schemaInconsistencyCheck, esta comprobación previa verifica que los metadatos de la tabla de MySQL son coherentes antes de continuar con la actualización. En este caso, la comprobación previa verifica que las definiciones de columna coincidan entre el diccionario de datos de InnoDB y la definición de tabla de MySQL. Si se detecta una discrepancia, no se continuará con la actualización.

Si detecta algún error con esta comprobación previa, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

Ejemplo de salida:

```
{
  "id": "getMismatchedMetadata",
  "title": "Column definition mismatch between InnoDB Data Dictionary and actual
 table definition.",
  "status": "OK",
  "description": "Error: Your database has mismatched metadata. The upgrade to mysql
 8.0 will not succeed until this is fixed.",
  "detectedProblems": [
```

```
      {
        "level": "Error",
        "dbObject": "test.mismatchTable",
        "description": "Table `test/mismatchTable` column names mismatch with InnoDb
 dictionary column names: iD <> id"
      }
  ]
}
```

La comprobación previa indica que hay una discrepancia en los metadatos de la columna `id` de la tabla `test.mismatchTable`. En concreto, los metadatos de MySQL tienen el nombre de la columna como `iD`, mientras que InnoDB lo tiene como `id`.

getTriggersWithNullDefiner

Nivel de comprobación previa: error

La columna de definidor de **`information_schema.triggers`** no puede ser **`null`** ni estar vacía.

La comprobación previa valida que la base de datos no tenga desencadenadores definidos con definidores `null` o vacíos. Para obtener más información sobre los requisitos del definidor para los objetos almacenados, consulte getEventsWithNullDefiner.

Ejemplo de salida:

```
{
  "id": "getTriggersWithNullDefiner",
  "title": "The definer column for information_schema.triggers cannot be null or
 blank.",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.example_trigger",
        "description": "Set definer for trigger example_trigger in Schema test"
      }
  ]
}
```

La comprobación previa devuelve un error porque el desencadenador `example_trigger` del esquema `test` tiene un definidor `null`. Para solucionar este problema, corrija el definidor

volviendo a crear el desencadenador con un usuario válido o elimínelo. Para obtener más información, consulte el ejemplo en getEventsWithNullDefiner.

> **Note**
>
> Antes de eliminar o volver a definir un `DEFINER`, revise y compruebe detenidamente los requisitos de aplicación y privilegios. Para obtener más información, consulte la sección Stored object access control en la documentación de MySQL.

getValueOfVariablelower_case_table_names

Nivel de comprobación previa: error

Todos los nombres de bases de datos y tablas deben estar en minúscula cuando el parámetro **`lower_case_table_names`** esté establecido en **`1`**.

En las versiones anteriores a MySQL 8.0, los nombres de bases de datos, los nombres de tablas y otros objetos correspondían a los archivos del directorio de datos, como, por ejemplo, los metadatos basados en archivos (.frm). La variable de sistema lower_case_table_names permite a los usuarios controlar la forma en que el servidor distingue entre mayúsculas y minúsculas en los identificadores de los objetos de la base de datos y el almacenamiento de dichos objetos de metadatos. Este parámetro se puede cambiar en un servidor ya inicializado tras un reinicio.

Sin embargo, en MySQL 8.0, aunque este parámetro sigue controlando la forma en que el servidor distingue entre mayúsculas y minúsculas en los identificadores, no se puede cambiar una vez inicializado el diccionario de datos. Al actualizar o crear una base de datos MySQL 8.0, el valor que se establezca para `lower_case_table_names` la primera vez que se inicie el diccionario de datos en MySQL será el que se utilice durante toda la vida útil de dicha base de datos. Esta restricción se ha establecido como parte de la implementación del Atomic Data Dictionary, en la que los objetos de la base de datos se migran de los metadatos basados en archivos a las tablas internas de InnoDB del esquema `mysql`.

Para obtener más información, consulte la sección Data dictionary changes en la documentación de MySQL.

Para evitar problemas a la hora de actualizar los metadatos basados en archivos al nuevo Atomic Data Dictionary, esta comprobación previa valida que, si está establecida la variable `lower_case_table_names = 1`, todas las tablas se almacenen en el disco en minúsculas. En

caso contrario, aparecerá un error de comprobación previa, y tendrá que corregir los metadatos antes de continuar con la actualización.

Ejemplo de salida:

```
{
  "id": "getValueOfVariablelower_case_table_names",
  "title": "MySQL pre-checks that all database or table names are lowercase when the
 lower_case_table_names parameter is set to 1.",
  "status": "OK",
  "description": "Error: You have one or more databases or tables with uppercase
 letters in the names, but the lower_case_table_names parameter is set to 1. To
 upgrade to MySQL 8.0, either change all database or table names to lowercase, or
 set the parameter to 0.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.TEST",
        "description": "Table test.TEST contains one or more capital letters in name
 while lower_case_table_names = 1"
      }
  ]
}
```

Se devuelve un error porque la tabla `test.TEST` incluye letras mayúsculas, pero la variable `lower_case_table_names` está establecida en 1.

Para resolver este problema, puede cambiar el nombre de la tabla a minúsculas o modificar el parámetro `lower_case_table_names` del clúster de base de datos antes de iniciar la actualización.

> **Note**
>
> Pruebe y revise detenidamente la documentación sobre la distinción entre mayúsculas y minúsculas en MySQL y cómo estos cambios podrían afectar a la aplicación.
> Consulte también la documentación de MySQL 8.0 sobre cómo se gestionan las variables lower_case_table_names de forma diferente en MySQL 8.0.

groupByAscSyntaxCheck

Nivel de comprobación previa: error

Uso de la sintaxis de **`GROUP BY ASC/DESC`** eliminada

A partir de la versión 8.0.13 de MySQL, se ha eliminado la sintaxis obsoleta de `ASC` o `DESC` para las cláusulas `GROUP BY`. Las consultas que se basan en la clasificación `GROUP BY` ahora pueden producir resultados diferentes. Para obtener un orden de clasificación específico, utilice una cláusula `ORDER BY`. Si existe algún objeto en la base de datos que utilice esta sintaxis, debe volver a crearlo mediante una cláusula `ORDER BY` antes de volver a intentar la actualización. Para obtener más información, consulte la sección SQL changes en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "groupbyAscSyntaxCheck",
  "title": "Usage of removed GROUP BY ASC/DESC syntax",
  "status": "OK",
  "description": "Error: The following DB objects use removed GROUP BY ASC/DESC
 syntax. They need to be altered so that ASC/DESC keyword is removed from GROUP BY
 clause and placed in appropriate ORDER BY clause.",
  "documentationLink": "https://dev.mysql.com/doc/relnotes/mysql/8.0/en/
news-8-0-13.html#mysqld-8-0-13-sql-syntax",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.groupbyasc",
        "description": "PROCEDURE uses removed GROUP BY ASC syntax",
        "dbObjectType": "Routine"
      }
  ]
}
```

mysqlEmptyDotTableSyntaxCheck

Nivel de comprobación previa: error

Comprobación de si la sintaxis de **`.<table>`** utilizada en las rutinas está obsoleta.

En MySQL 8.0, las rutinas ya no pueden incluir la sintaxis de identificador obsoleta (`\".<table>\"`). Si alguna de las rutinas o desencadenadores almacenados incluye dichos identificadores, se producirá un error en la actualización. Por ejemplo, ya no se permite la siguiente referencia de `.dot_table`:

```
mysql> show create procedure incorrect_procedure\G
```

```
*************************** 1. row ***************************
           Procedure: incorrect_procedure
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `incorrect_procedure`()
BEGIN delete FROM .dot_table; select * from .dot_table where 1=1; END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Tras volver a crear las rutinas y los desencadenadores para utilizar la sintaxis de identificador correcta y la secuencia de escape, se aprueba la comprobación previa y se puede continuar con la actualización. Para obtener más información sobre los identificadores, consulte la sección Schema object names en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "mysqlEmptyDotTableSyntaxCheck",
  "title": "Check for deprecated '.<table>' syntax used in routines.",
  "status": "OK",
  "description": "Error: The following routines contain identifiers in deprecated
 identifier syntax (\".<table>\"), and should be corrected before upgrade:\n",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.incorrect_procedure",
        "description": " routine body contains deprecated identifiers."
      }
  ]
}
```

La comprobación previa devuelve un error para la rutina `incorrect_procedure` de la base de datos `test`, ya que incluye una sintaxis obsoleta.

Tras corregir la rutina, la comprobación previa se realiza correctamente y puede volver a intentar la actualización.

mysqlIndexTooLargeCheck

Nivel de comprobación previa: error

Comprobación de si los índices son demasiado grandes para funcionar en las versiones de MySQL posteriores a 5.7

En el caso de formatos de fila compactos o redundantes, no debería ser posible crear un índice con un prefijo superior a 767 bytes. Sin embargo, antes de la versión 5.7.35 de MySQL esto era posible. Para obtener más información, consulte las notas de la versión de MySQL 5.7.35.

No se podrá acceder a los índices afectados por este error tras la actualización a MySQL 8.0. Esta comprobación previa identifica los índices problemáticos que deben volver a crearse antes de continuar con la actualización.

```
{
 "id": "mysqlIndexTooLargeCheck",
 "title": "Check for indexes that are too large to work on higher versions of MySQL
Server than 5.7",
 "status": "OK",
 "description": "Error: The following indexes ware made too large for their format
in an older version of MySQL (older than 5.7.34). Normally those indexes within
tables with compact or redundant row formats shouldn't be larger than 767 bytes. To
fix this problem those indexes should be dropped before upgrading or those tables
will be inaccessible.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test.table_with_large_idx",
       "description": "IDX_2"
     }
 ]
}
```

La comprobación previa devuelve un error porque la tabla `test.table_with_large_idx` incluye un índice en una tabla que utiliza un formato de fila compacto o redundante de más de 767 bytes. Ya no se podrá acceder a estar tablas después de actualizar a MySQL 8.0. Antes de continuar con la actualización, realice una de las siguientes acciones:

- Elimine el índice mencionado en la comprobación previa.
- Añada un índice mencionado en la comprobación previa.
- Cambie el formato de fila utilizado en la tabla.

En este caso, volvemos a crear la tabla para resolver el error de comprobación previa. Antes de volver a crear la tabla, asegúrese de que innodb_file_format esté establecido en

`Barracuda` y que innodb_default_row_format esté establecido en `dynamic`. Estos son los valores predeterminados en MySQL 5.7. Para obtener más información, consulte las secciones InnoDB row formats y InnoDB file-format management en la documentación de MySQL.

> **Note**
>
> Antes de volver a crear los espacios de tablas, consulte la sección Online DDL operations en la documentación de MySQL para conocer los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

```
mysql > select @@innodb_file_format,@@innodb_default_row_format;
+----------------------+-----------------------------+
| @@innodb_file_format | @@innodb_default_row_format |
+----------------------+-----------------------------+
| Barracuda            | dynamic                     |
+----------------------+-----------------------------+
1 row in set (0.00 sec)

mysql> optimize table table_with_large_idx;
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| Table                     | Op       | Msg_type | Msg_text
                            |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| test.table_with_large_idx | optimize | note     | Table does not support optimize,
 doing recreate + analyze instead |
| test.table_with_large_idx | optimize | status   | OK
                            |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.02 sec)

# Verify FILE_FORMAT and ROW_FORMAT
mysql>  select * from information_schema.innodb_sys_tables where name like 'test/
table_with_large_idx';
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
| TABLE_ID | NAME                      | FLAG | N_COLS | SPACE | FILE_FORMAT |
 ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
```

```
+----------+---------------------------+------+--------+-------+-------------
+------------+--------------+------------+
|       43 | test/table_with_large_idx |   33 |      4 |    26 | Barracuda   |
 Dynamic    |            0 | Single     |
+----------+---------------------------+------+--------+-------+-------------
+------------+--------------+------------+
1 row in set (0.00 sec)
```

Tras volver a crear la tabla, se aprueba la comprobación previa y se puede continuar con la actualización.

```
{
  "id": "mysqlIndexTooLargeCheck",
  "title": "Check for indexes that are too large to work on higher versions of MySQL
 Server than 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlInvalid57NamesCheck

Nivel de comprobación previa: error

Compruebe si los nombres de tablas y esquemas utilizados en MySQL 5.7 no son válidos

Al migrar al nuevo diccionario de datos de MySQL 8.0, la instancia de base de datos no puede incluir esquemas ni tablas con el prefijo `#mysql50#`. Si existe alguno de estos objetos, se produce un error en la actualización. Para resolver este problema, ejecute mysqlcheck con los esquemas y tablas devueltos.

> **Note**
>
> Asegúrese de utilizar la versión MySQL 5.7 de `mysqlcheck`, ya que --fix-db-names y --fix-table-names se han eliminado de MySQL 8.0.

Ejemplo de salida:

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
```

```
  "description": "The following tables and/or schemas have invalid names. In order
 to fix them use the mysqlcheck utility as follows:\n\n  $ mysqlcheck --check-
upgrade --all-databases\n  $ mysqlcheck --fix-db-names --fix-table-names --all-
databases\n\nOR via mysql client, for eg:\n\n  ALTER DATABASE `#mysql50#lost+found`
 UPGRADE DATA DIRECTORY NAME;",
  "documentationLink": "https://dev.mysql.com/doc/refman/5.7/en/identifier-
mapping.html https://dev.mysql.com/doc/refman/5.7/en/alter-database.html https://
dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "#mysql50#fix_db_names",
        "description": "Schema name"
      }
  ]
}
```

La comprobación previa indica que el esquema `#mysql50#fix_db_names` tiene un nombre no válido.

Tras corregir el nombre del esquema, se aprueba la comprobación previa, lo que permite continuar con la actualización.

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlOrphanedRoutinesCheck

Nivel de comprobación previa: error

Comprobación de si hay rutinas huérfanas en la versión 5.7

Al migrar al nuevo diccionario de datos de MySQL 8.0, si hay algún procedimiento almacenado en la base de datos donde el esquema ya no existe, se producirá un error en la actualización. Esta comprobación previa comprueba que todos los esquemas a los que se hace referencia en los procedimientos almacenados de la instancia de base de datos siguen existiendo. Para poder continuar con la actualización, elimine estos procedimientos almacenados.

Ejemplo de salida:

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "description": "Error: The following routines have been orphaned. Schemas that
 they are referencing no longer exists.\nThey have to be cleaned up or the upgrade
 will fail.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "dropped_db.get_version",
        "description": "is orphaned"
      }
  ]
},
```

La comprobación previa indica que el procedimiento almacenado `get_version` de la base de datos `dropped_db` está huérfano.

Para eliminar este procedimiento, puede volver a crear el esquema que falta.

```
mysql> create database dropped_db;
Query OK, 1 row affected (0.01 sec)
```

Una vez que se haya vuelto a crear el esquema, puede eliminar el procedimiento para permitir que continúe la actualización.

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlSchemaCheck

Nivel de comprobación previa: error

Los nombres de las tablas del esquema de **`mysql`** entran en conflicto con las nuevas tablas de MySQL 8.0

El nuevo Atomic Data Dictionary introducido en MySQL 8.0 almacena todos los metadatos en un conjunto de tablas internas de InnoDB en el esquema de `mysql`. Durante la actualización, las nuevas tablas del diccionario de datos interno se crean en el esquema de `mysql`. Para evitar colisiones de nombres, que podrían provocar errores en la actualización, la comprobación previa examina todos los nombres de las tablas del esquema de `mysql` para garantizar que ninguno de los nuevos nombres de tabla esté ya en uso. Si lo están, aparece un error y no se permite continuar con la actualización.

Ejemplo de salida:

```
{
  "id": "mysqlSchema",
  "title": "Table names in the mysql schema conflicting with new tables in the
 latest MySQL.",
  "status": "OK",
  "description": "Error: The following tables in mysql schema have names that will
 conflict with the ones introduced in the latest version. They must be renamed or
 removed before upgrading (use RENAME TABLE command). This may also entail changes
 to applications that use the affected tables.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrade-before-you-
begin.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.tablespaces",
        "description": "Table name used in mysql schema.",
        "dbObjectType": "Table"
      }
  ]
}
```

Se devuelve un error porque hay una tabla con el nombre `tablespaces` en el esquema de `mysql`. Este es uno de los nuevos nombres de tablas de diccionarios de datos internos de MySQL 8.0. Debe cambiar el nombre de dichas tablas o eliminarlas antes de realizar la actualización mediante el comando `RENAME TABLE`.

nonNativePartitioningCheck

Nivel de comprobación previa: error

Tablas particionadas que utilizan motores con particionamiento no nativo

Según la documentación de MySQL 8.0, actualmente hay dos motores de almacenamiento que ofrecen soporte de particionamiento nativo: InnoDB y NDB. De estos, solo InnoDB se admite en la versión 3 de Aurora MySQL, compatible con MySQL 8.0. Se producirá un error al intentar crear tablas particionadas en MySQL 8.0 con cualquier otro motor de almacenamiento. Esta comprobación previa busca tablas en el clúster de la base de datos que utilicen particiones no nativas. Si se devuelve alguna, debe eliminar la partición o convertir el motor de almacenamiento a InnoDB.

Ejemplo de salida:

```
{
  "id": "nonNativePartitioning",
  "title": "Partitioned tables using engines with non native partitioning",
  "status": "OK",
  "description": "Error: In the latest MySQL storage engine is responsible for
 providing its own partitioning handler, and the MySQL server no longer provides
 generic partitioning support. InnoDB and NDB are the only storage engines that
 provide a native partitioning handler that is supported in the latest MySQL. A
 partitioned table using any other storage engine must be altered—either to convert
 it to InnoDB or NDB, or to remove its partitioning—before upgrading the server,
 else it cannot be used afterwards.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-configuration-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partMyisamTable",
         "description": "MyISAM engine does not support native partitioning",
         "dbObjectType": "Table"
      }
  ]
}
```

En este caso, una tabla MyISAM utiliza la partición, lo que requiere una acción antes de que la actualización pueda continuar.

oldTemporalCheck

Nivel de comprobación previa: error

Uso de un tipo temporal antiguo

Los “Tipos temporales antiguos” hacen referencia a las columnas de tipo temporal (como `TIMESTAMP` y `DATETIME`) creadas en las versiones 5.5 y anteriores de MySQL. En MySQL 8.0, se ha eliminado la compatibilidad con estos tipos de datos temporales antiguos, lo que significa que las actualizaciones locales de MySQL 5.7 a 8.0 no son posibles si la base de datos los incluye. Para solucionar este problema, debe volver a crear todas las tablas que incluyan estos tipos de fechas temporales antiguos antes de continuar con la actualización.

Para obtener más información sobre la obsolescencia de los tipos de datos temporales antiguos en MySQL 5.7, consulte este blog. Para obtener más información sobre la eliminación de los tipos de datos temporales antiguos en MySQL 8.0, consulte este blog.

> **Note**
>
> Antes de volver a crear los espacios de tablas, consulte la sección Online DDL operations en la documentación de MySQL para conocer los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

Ejemplo de salida:

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "description": "Error: Following table columns use a deprecated and no longer
 supported temporal disk storage format. They must be converted to the new format
 before upgrading. It can by done by rebuilding the table using 'ALTER TABLE
 <table_name> FORCE' command",
  "documentationLink": "https://dev.mysql.com/blog-archive/mysql-8-0-removing-
support-for-old-temporal-datatypes/",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.55_temporal_table.timestamp_column",
        "description": "timestamp /* 5.5 binary format */",
        "dbObjectType": "Column"
      }
  ]
},
```

Se informa de un error en la columna `timestamp_column` de la tabla `test.55_temporal_table`, ya que utiliza un antiguo formato de almacenamiento en disco temporal que ya no se admite.

Para resolver este problema y permitir que se continúe con la actualización, vuelva a crear la tabla para convertir el antiguo formato de almacenamiento en disco temporal al nuevo introducido en MySQL 5.6. Para obtener más información y conocer los requisitos previos antes de hacerlo, consulte la sección Converting between 3-byte and 4-byte Unicode character sets en la documentación de MySQL.

Al ejecutar el siguiente comando para volver a crear las tablas mencionadas en esta comprobación previa, se convierten los tipos temporales antiguos al formato más nuevo con una precisión de fracción de segundo.

```
ALTER TABLE ... ENGINE=InnoDB;
```

Para obtener más información sobre cómo volver a crear tablas, consulte ALTER TABLE statement en la documentación de MySQL.

Tras volver a crear la tabla en cuestión y reiniciar la actualización, se aprueba la comprobación de compatibilidad, lo que permite continuar con la actualización.

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "detectedProblems": []
}
```

partitionedTablesInSharedTablespaceCheck

Nivel de comprobación previa: error

Uso de tablas particionadas en espacios de tablas compartidos

A partir de la versión 8.0.13 de MySQL, se elimina la compatibilidad con la colocación de particiones de tablas en espacios de tabla compartidos. Antes de realizar la actualización, mueva dichas tablas de los espacios de tabla compartidos a los espacios de tabla file-per-table.

> **Note**
>
> Antes de volver a crear los espacios de tablas, consulte la sección Partitioning operations en la documentación de MySQL para comprender los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

Ejemplo de salida:

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "description": "Error: The following tables have partitions in shared tablespaces.
 They need to be moved to file-per-table tablespace before upgrading. You can do
 this by running query like 'ALTER TABLE table_name REORGANIZE PARTITION X INTO
 (PARTITION X VALUES LESS THAN (30) TABLESPACE=innodb_file_per_table);'",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partInnoDBTable",
        "description": "Partition p1 is in shared tablespace innodb",
        "dbObjectType": "Table"
      }
  ]
}
```

La comprobación previa no se realiza correctamente porque la partición p1 de la tabla `test.partInnoDBTable` se encuentra en el espacio de tablas del sistema.

Para resolver este problema, vuelva a crear la tabla `test.partInnodbTable` y coloque la partición problemática p1 en un espacio de tablas file-per-table.

```
mysql > ALTER TABLE partInnodbTable REORGANIZE PARTITION p1
    ->   INTO (PARTITION p1 VALUES LESS THAN ('2014-01-01')
 TABLESPACE=innodb_file_per_table);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Después de hacerlo, se aprueba la comprobación previa.

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "detectedProblems": []
}
```

removedFunctionsCheck

Nivel de comprobación previa: error

Uso de funciones que se eliminaron de la última versión de MySQL

En MySQL 8.0, se han eliminado varias funciones integradas. Esta comprobación previa examina la base de datos en busca de objetos que puedan utilizar estas funciones. En el caso de detectarse, se devuelve un error. Debe resolver los problemas antes de volver a intentar la actualización.

La mayoría de las funciones eliminadas son funciones espaciales, que se han sustituido por funciones ST_* equivalentes. En estos casos, modifique los objetos de la base de datos para utilizar la nueva nomenclatura del procedimiento. Para obtener más información, consulte la sección sobre características eliminadas en MySQL 8.0, en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "description": "Error: The following DB objects use functions that were removed
 in the latest MySQL version. Please make sure to update them to use supported
 alternatives before upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.GetLocationsInPolygon",
        "description": "PROCEDURE uses removed function POLYGONFROMTEXT (consider
 using ST_POLYGONFROMTEXT instead)",
```

```
        "dbObjectType": "Routine"
      },
      {
        "level": "Error",
        "dbObject": "test.InsertLocation",
        "description": "PROCEDURE uses removed function POINTFROMTEXT (consider
 using ST_POINTFROMTEXT instead)",
        "dbObjectType": "Routine"
      }
  ]
},
```

La comprobación previa indica que el procedimiento almacenado `test.GetLocationsInPolygon` utiliza dos funciones eliminadas: POLYGONFROMTEXT y POINTFROMTEXT. También sugiere que utilice las nuevas funciones ST_POLYGONFROMTEXT y ST_POINTFROMTEXT como sustitutas. Tras volver a crear el procedimiento con las sugerencias, la comprobación previa se completará correctamente.

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "detectedProblems": []
},
```

> **Note**
>
> Aunque en la mayoría de los casos las funciones obsoletas se sustituyen directamente, asegúrese de probar la aplicación y revisar la documentación para comprobar si se ha producido algún cambio de comportamiento como consecuencia del cambio.

routineSyntaxCheck

Nivel de comprobación previa: error

Comprobación de sintaxis de MySQL para objetos de tipo rutinario

MySQL 8.0 ha introducido palabras clave reservadas que no estaban reservadas anteriormente. El comprobador previo de actualización evalúa el uso de palabras clave reservadas en los nombres de los objetos de la base de datos, así como en sus definiciones y cuerpo. Si

detecta que se utilizan palabras clave reservadas en los objetos de la base de datos, como procedimientos almacenados, funciones, eventos y desencadenadores, la actualización no se realizará correctamente y se publicará un error en el archivo `upgrade-prechecks.log`. Para resolver el problema, debe actualizar estas definiciones de objetos y escribir dichas referencias entre comillas simples (') antes de realizar la actualización. Para obtener más información sobre cómo aplicar caracteres de escape a las palabras reservadas en MySQL, consulte la sección String literals en la documentación de MySQL.

También puede cambiar el nombre por uno diferente, lo que puede requerir cambios en la aplicación.

Ejemplo de salida:

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "description": "The following objects did not pass a syntax check with the latest
 MySQL grammar. A common reason is that they reference names that conflict with new
 reserved keywords. You must update these routine definitions and `quote` any such
 references before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
         "level": "Error",
         "dbObject": "test.select_res_word",
         "description": "at line 2,18: unexpected token 'except'",
         "dbObjectType": "Routine"
      }
  ]
}
```

Para resolver este problema, compruebe la definición de rutina.

```
SHOW CREATE PROCEDURE test.select_res_word\G

*************************** 1. row ***************************
           Procedure: select_res_word
            sql_mode:
 ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_Z
    Create Procedure: CREATE PROCEDURE 'select_res_word'()
BEGIN
```

```
    SELECT * FROM except;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

El procedimiento utiliza una tabla denominada `except`, que es una nueva palabra clave en MySQL 8.0. Aplique caracteres de escape al literal de cadena para volver a crear el procedimiento.

```
> drop procedure if exists select_res_word;
Query OK, 0 rows affected (0.00 sec)

> DELIMITER $$
 > CREATE PROCEDURE select_res_word()
    -> BEGIN
    ->     SELECT * FROM 'except';
    -> END$$
Query OK, 0 rows affected (0.00 sec)

 > DELIMITER ;
```

Ahora se aprueba la comprobación previa.

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "detectedProblems": []
}
```

schemaInconsistencyCheck

Nivel de comprobación previa: error

Incoherencias en el esquema provocadas por la eliminación o la corrupción de un archivo

Tal y como se ha descrito anteriormente, MySQL 8.0 ha introducido el Atomic Data Dictionary, que almacena todos los metadatos en un conjunto de tablas internas de InnoDB en el esquema de `mysql`. Esta nueva arquitectura proporciona una forma transaccional y compatible con ACID para administrar los metadatos de las bases de datos, lo que resuelve el problema de DDL

atómico que planteaba el antiguo enfoque basado en archivos. En las versiones anteriores a MySQL 8.0, era posible que los objetos del esquema quedaran huérfanos si una operación de DDL se interrumpía de forma inesperada. La migración de los metadatos basados en archivos a las nuevas tablas del Atomic Data Dictionary durante la actualización garantiza que no haya objetos de esquema huérfanos en la instancia de la base de datos. Si se encuentra algún objeto huérfano, se producirá un error en la actualización.

Con el fin de ayudar a detectar estos objetos huérfanos antes de iniciar la actualización, se realiza una comprobación previa `schemaInconsistencyCheck` para garantizar que todos los objetos de metadatos del diccionario de datos estén sincronizados. Si se detecta algún objeto de metadatos huérfano, no se continuará con la actualización. Para continuar con la actualización, elimine estos objetos de metadatos huérfanos.

Si detecta algún error con esta comprobación previa, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

Ejemplo de salida:

```
{
  "id": "schemaInconsistencyCheck",
  "title": "Schema inconsistencies resulting from file removal or corruption",
  "status": "OK",
  "description": "Error: Following tables show signs that either table datadir
 directory or frm file was removed/corrupted. Please check server logs, examine
 datadir to detect the issue and fix it before upgrade",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.schemaInconsistencyCheck_failure",
        "description": "present in INFORMATION_SCHEMA's INNODB_SYS_TABLES table but
 missing from TABLES table"
      }
  ]
}
```

La comprobación previa indica que la tabla `test.schemaInconsistencyCheck_failure` tiene metadatos incoherentes. En este caso, la tabla existe en los metadatos del motor de almacenamiento de InnoDB (`information_schema.INNODB_SYS_TABLES`), pero no en los metadatos de MySQL (`information_schema.TABLES`).

Comprobaciones previas de Aurora MySQL que informan de errores

Las siguientes comprobaciones previas son específicas de Aurora MySQL:

- auroraCheckDDLRecovery
- auroraCheckRdsUpgradePrechecksTable
- auroraFODUpgradeCheck
- auroraGetDanglingFulltextIndex
- auroraUpgradeCheckForDatafilePathInconsistency
- auroraUpgradeCheckForFtsSpaceIdZero
- auroraUpgradeCheckForIncompleteXATransactions
- auroraUpgradeCheckForInstanceLimit
- auroraUpgradeCheckForInternalUsers
- auroraUpgradeCheckForMasterUser
- auroraUpgradeCheckForPrefixIndexOnGeometryColumns
- auroraUpgradeCheckForSpecialCharactersInProcedures
- auroraUpgradeCheckForSysSchemaObjectTypeMismatch
- auroraUpgradeCheckForViewColumnNameLength
- auroraUpgradeCheckIndexLengthLimitOnTinyColumns
- auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable
- auroraUpgradeCheckForInvalidUtf8mb3ColumnComments

auroraCheckDDLRecovery

Nivel de comprobación previa: error

Comprobación de si hay artefactos relacionados con la característica de recuperación de Aurora DDL

Como parte de la implementación de recuperación del lenguaje de definición de datos (DDL) en Aurora MySQL, los metadatos de las instrucciones de DDL en curso se mantienen en las tablas `ddl_log_md_table` y `ddl_log_table` del esquema de `mysql`. La implementación de recuperación de DDL de Aurora no es compatible con la versión 3 y versiones posteriores,

ya que la funcionalidad forma parte de la nueva implementación de Atomic Data Dictionary en MySQL 8.0. Si se está ejecutando alguna instrucción de DDL durante las comprobaciones de compatibilidad, es posible que se produzca un error en esta comprobación previa. Le recomendamos que intente actualizar cuando no haya en ejecución ninguna instrucción de DDL.

Si se produce un error en esta comprobación previa sin que se estén ejecutando instrucciones de DDL, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

Si se está ejecutando alguna instrucción de DDL, el resultado de la comprobación previa mostrará el siguiente mensaje:

```
“There are DDL statements in process. Please allow DDL statements to finish before
 upgrading.”
```

Ejemplo de salida:

```
{
  "id": "auroraCheckDDLRecovery",
  "title": "Check for artifacts related to Aurora DDL recovery feature",
  "status": "OK",
  "description": "Aurora implementation of DDL recovery is not supported from 3.x
 onwards. This check verifies that the database do not have artifacts realted to the
 feature",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_md_table",
        "description": "Table mysql.ddl_log_md_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_table",
        "description": "Table mysql.ddl_log_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "information_schema.processlist",
```

```
        "description": "There are DDL statements in process. Please allow DDL
 statements to finish before upgrading."
      }
  ]
}
```

La comprobación previa ha devuelto un error debido a que una DDL en curso se estaba ejecutando simultáneamente con las comprobaciones de compatibilidad. Le recomendamos que intente de nuevo realizar la actualización sin que se esté ejecutando ninguna DDL.

auroraCheckRdsUpgradePrechecksTable

Nivel de comprobación previa: error

Comprobación de la existencia de la tabla **`mysql.rds_upgrade_prechecks`**

Se trata de una comprobación previa exclusivamente interna llevada a cabo por el servicio de RDS. Los errores se gestionarán automáticamente en la actualización y pueden omitirse de forma segura.

Si detecta algún error con esta comprobación previa, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

```
{
  "id": "auroraCheckRdsUpgradePrechecksTable",
  "title": "Check existence of mysql.rds_upgrade_prechecks table",
  "status": "OK",
  "detectedProblems": []
}
```

auroraFODUpgradeCheck

Nivel de comprobación previa: error

Comprobación de si hay artefactos relacionados con la característica DDL rápido de Aurora

La optimización de DDL rápido se ha introducido en el modo lab en la versión 2 de Aurora MySQL con el fin de mejorar la eficiencia de algunas operaciones de DDL. En la versión 3 de Aurora MySQL, se ha eliminado el modo lab y la implementación de DDL rápido se ha sustituido por la característica de MySQL 8.0 denominada DDL instantáneo.

Antes de actualizar a la versión 3 de Aurora MySQL, deberá volverse a crear cualquier tabla que utilice DDL rápido en modo laboratorio. Para ello, es necesario ejecutar el comando `OPTIMIZE TABLE` o `ALTER TABLE ... ENGINE=InnoDB` para garantizar la compatibilidad con la versión 3 de Aurora MySQL.

Esta comprobación previa devuelve una lista de estas tablas. Una vez se han vuelto a crear las tablas devueltas, puede volver a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "description": "Aurora fast DDL is not supported from 3.x onwards. This check
 verifies that the database does not have artifacts related to the feature",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/AuroraMySQL.Managing.FastDDL.html#AuroraMySQL.Managing.FastDDL-v2",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test",
        "description": "Your table has pending Aurora fast DDL operations. Run
 'OPTIMIZE TABLE <table name>' for the table to apply all the pending DDL updates.
 Then try the upgrade again."
      }
  ]
}
```

La comprobación previa indica que la tabla `test.test` tiene operaciones de DDL rápido pendientes.

Para poder continuar con la actualización, puede volver a crear la tabla y, a continuación, volver a intentar la actualización.

> **Note**
>
> Antes de volver a crear los espacios de tablas, consulte la sección Online DDL operations en la documentación de MySQL para conocer los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

```
mysql> optimize table test.test;
+-----------+----------+----------
+-------------------------------------------------------------------+
| Table     | Op       | Msg_type | Msg_text
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
| test.test | optimize | note     | Table does not support optimize, doing recreate
 + analyze instead |
| test.test | optimize | status   | OK
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.04 sec)
```

Después de volver a crear la tabla, la comprobación previa se realiza correctamente.

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "detectedProblems": []
}
```

auroraGetDanglingFulltextIndex

Nivel de comprobación previa: error

Tablas con referencia de índice **`FULLTEXT`** colgante

En las versiones anteriores a MySQL 5.6.26, era posible que, tras eliminar un índice de búsqueda de texto completo, las columnas ocultas `FTS_DOC_ID` y `FTS_DOC_ID_INDEX` quedaran huérfanas. Para obtener más información, consulte Bug #76012.

Si ha creado tablas en versiones anteriores de MySQL en las que esto haya ocurrido, puede provocar un error en las actualizaciones a la versión 3 de Aurora MySQL. Esta comprobación previa verifica que no existan índices de texto completo huérfanos o “colgantes” en el clúster de base de datos antes de actualizar a MySQL 8.0. Si se produce un error en esta comprobación previa, vuelva a crear todas las tablas que incluyan esos índices de texto completo colgantes.

Ejemplo de salida:

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "description": "Error: The following tables contain dangling FULLTEXT index which
 is not supported. It is recommended to rebuild the table before upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.table_with_fts_index",
        "description": "Table `test.table_with_fts_index` contains dangling FULLTEXT
 index. Kindly recreate the table before upgrade."
      }
  ]
},
```

La comprobación previa indica que hay un error en la tabla `test.table_with_fts_index` porque incluye un índice de texto completo colgante. Para permitir que se continúe con la actualización, vuelva a crear la tabla para eliminar las tablas auxiliares del índice de texto completo. Utilice `OPTIMIZE TABLE test.table_with_fts_index` o `ALTER TABLE test.table_with_fts_index, ENGINE=INNODB`.

Después de volver a crear la tabla, se aprueba la comprobación previa.

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForDatafilePathInconsistency

Nivel de comprobación previa: error

Comprobación de si hay incoherencias relacionadas con la ruta de archivo **`ibd`**

Esta comprobación previa se aplica únicamente a la versión 3.03.0 y anteriores de Aurora MySQL. Si detecta un error con esta comprobación previa, actualice a la versión 3.04 o versiones posteriores de Aurora MySQL.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForFtsSpaceIdZero

Nivel de comprobación previa: error

Comprobación del índice de texto completo con el ID de espacio como cero

En MySQL, al añadir un índice de texto completo a una tabla de InnoDB, se crean varios espacios de tabla de índices auxiliares. Debido a un error en las versiones anteriores de MySQL, que se ha corregido en la versión 5.6.20, era posible que estas tablas de índices auxiliares se crearan en el espacio de tablas del sistema, en lugar de hacerlo en su propio espacio de tablas de InnoDB.

Si existe alguno de estos espacios de tablas auxiliares, se producirá un error en la actualización. Vuelva a crear los índices de texto completo mencionados en el error de comprobación previa y, a continuación, vuelva a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForFtsSpaceIdZero",
  "title": "Check for fulltext index with space id as zero",
  "status": "OK",
  "description": "The auxiliary tables of FTS indexes on the table are created
 in system table-space. Due to this DDL queries executed on MySQL8.0 shall cause
 database unavailability. To avoid that, drop and recreate all the FTS indexes on
 the table or rebuild the table using ALTER TABLE query before the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.fts_space_zero_check",
        "description": " The auxiliary tables of FTS indexes on the table
 'test.fts_space_zero_check' are created in system table-space due to https://
bugs.mysql.com/bug.php?id=72132. In MySQL8.0, DDL queries executed on this table
 shall cause database unavailability. To avoid that, drop and recreate all the
 FTS indexes on the table or rebuild the table using ALTER TABLE query before the
 upgrade."
      }
```

```
  ]
},
```

La comprobación previa indica que hay un error en la tabla `test.fts_space_zero_check`, ya que tiene tablas auxiliares de búsqueda de texto completo (FTS) en el espacio de tablas del sistema.

Después de eliminar y volver a crear los índices de FTS asociados a esta tabla, la comprobación previa se realizará correctamente.

> **Note**
>
> Antes de volver a crear los espacios de tablas, consulte la sección Partitioning operations en la documentación de MySQL para comprender los efectos del bloqueo y el movimiento de datos en las transacciones en primer plano.

```
{
 "id": "auroraUpgradeCheckForFtsSpaceIdZero",
 "title": "Check for fulltext index with space id as zero",
 "status": "OK",
 "detectedProblems": []
}
```

auroraUpgradeCheckForIncompleteXATransactions

Nivel de comprobación previa: error

Comprobación de si las transacciones de XA están preparadas

Mientras se ejecuta el proceso de actualización de la versión principal, es esencial que la instancia de base de datos de la versión 2 de Aurora MySQL se cierre por completo. Esto garantiza que todas las transacciones se confirmen o se reviertan y que InnoDB haya purgado todos los registros de deshacer. Como la reversión de las transacciones es necesaria, si la base de datos tiene alguna transacción de XA preparada, puede impedir que se cierre por completo. Por este motivo, si se detecta alguna transacción de XA preparada, no se podrá continuar con la actualización hasta que se tomen medidas para confirmarla o revertirla.

Para obtener más información sobre cómo detectar transacciones de XA preparadas con `XA RECOVER`, consulte la sección XA Transaction SQL Statements en la documentación de MySQL.

Para obtener más información sobre los estados de las transacciones de XA, consulte la sección XA Transaction States en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "description": "Your cluster currently has XA transactions in the prepared state.
 To proceed with the upgrade, commit or rollback these transactions.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your cluster currently has XA transactions in the prepared
 state. To proceed with the upgrade, commit or rollback these transactions."
      }
  ]
}
```

Esta comprobación previa indica que hay un error porque hay transacciones preparadas que deben confirmarse o revertirse.

Tras iniciar sesión en la base de datos, puede consultar la tabla information_schema.innodb_trx y el resultado `XA RECOVER` para obtener más información.

> **Important**
>
> Antes de confirmar o revertir una transacción, le recomendamos que revise la documentación de MySQL y los requisitos de la aplicación.

```
mysql> select trx_started,
    trx_mysql_thread_id,
    trx_id,trx_state,
    trx_operation_state,
    trx_rows_modified,
    trx_rows_locked
from
    information_schema.innodb_trx;
```

```
+--------------------+--------------------+---------+-----------
+--------------------+------------------+-----------------+
| trx_started         | trx_mysql_thread_id | trx_id  | trx_state |
 trx_operation_state | trx_rows_modified | trx_rows_locked |
+--------------------+--------------------+---------+-----------
+--------------------+------------------+-----------------+
| 2024-08-12 01:09:39 |                   0 | 2849470 | RUNNING   | NULL
    |                 1 |               0 |
+--------------------+--------------------+---------+-----------
+--------------------+------------------+-----------------+
1 row in set (0.00 sec)

mysql> xa recover;
+----------+--------------+--------------+--------+
| formatID | gtrid_length | bqual_length | data   |
+----------+--------------+--------------+--------+
|        1 |            6 |            0 | xatest |
+----------+--------------+--------------+--------+
1 row in set (0.00 sec)
```

En este caso, revertimos la transacción preparada.

```
mysql> XA ROLLBACK 'xatest';
Query OK, 0 rows affected (0.00 sec)
v
mysql> xa recover;
Empty set (0.00 sec)
```

Una vez que se ha revertido la transacción de XA, la comprobación previa se realizará correctamente.

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInstanceLimit

Nivel de comprobación previa: error

Comprobación de si la clase de instancia actual admite la actualización

Por el momento, no se puede ejecutar una actualización local desde la versión 2.12.0 o 2.12.1 de Aurora MySQL, donde la clase de instancia de base de datos de escritor sea db.r6i.32xlarge. En este caso, la comprobación previa devuelve un error. Para poder continuar con la actualización, puede cambiar la clase de la instancia de base de datos a db.r6i.24xlarge o a una inferior. También puede actualizar a la versión 2.12.2 de Aurora MySQL, o una versión posterior, donde db.r6i.32xlarge admite la actualización local a la versión 3 de Aurora MySQL.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForInstanceLimit",
  "title": "Checks if upgrade is supported on the current instance class",
  "status": "OK",
  "description": "Upgrade from Aurora Version 2.12.0 and 2.12.1 may fail for 32.xl
 and above instance class.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Upgrade is not supported on this instance size for Aurora
 MySql Version 2.12.1. Before upgrading to Aurora MySql 3, please consider either:
 1. Changing the instance class to 24.xl or lower. -or- 2. Upgrading to patch
 version 2.12.2 or higher."
      }
  ]
},
```

La comprobación previa devuelve un error porque la instancia de base de datos de escritor utiliza la clase de instancia db.r6i.32xlarge y se ejecuta en la versión 2.12.1 de Aurora MySQL.

auroraUpgradeCheckForInternalUsers

Nivel de comprobación previa: error

Comprobación de si hay usuarios internos de la versión 8.0

Esta comprobación previa se aplica únicamente a la versión 3.03.0 y anteriores de Aurora MySQL. Si detecta un error con esta comprobación previa, actualice a la versión 3.04 o versiones posteriores de Aurora MySQL.

Ejemplo de salida:

```
{
```

```
  "id": "auroraUpgradeCheckForInternalUsers",
  "title": "Check for 8.0 internal users.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForMasterUser

Nivel de comprobación previa: error

Comprobación de si existe el usuario maestro de RDS

MySQL 8 ha añadido un nuevo modelo de privilegios con soporte de rol y privilegios dinámicos para hacer que la administración de privilegios sea más fácil y detallada. Como parte de este cambio, Aurora MySQL ha introducido el nuevo `rds_superuser_role`, que se otorga automáticamente al usuario maestro de la base de datos al actualizar de la versión 2 a la versión 3 de Aurora MySQL.

Para obtener más información sobre los roles y privilegios asignados al usuario maestro en Aurora MySQL, consulte Privilegios de la cuenta de usuario maestro. Para obtener más información sobre el modelo de privilegios basado en roles de la versión 3 de Aurora MySQL, consulte Modelo de privilegios basado en roles.

Esta comprobación previa verifica que el usuario maestro existe en la base de datos. Si el usuario maestro no existe, se producirá un error en la comprobación previa. Para poder continuar con la actualización, vuelva a crear el usuario maestro. Para ello, restablezca la contraseña del usuario maestro o cree manualmente el usuario. A continuación, vuelva a intentar la actualización. Para obtener más información sobre cómo restablecer la contraseña del usuario maestro, consulte Cambio de la contraseña del usuario maestro de la base de datos.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
  "description": "Throws error if master user has been dropped!",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/UsingWithRDS.MasterAccounts.html",
  "detectedProblems": [
      {
```

```
        "level": "Error",
        "dbObject": "all",
        "description": "Your Master User on host '%' has been dropped. To proceed
 with the upgrade, recreate the master user `reinvent` on default host '%'"
      }
  ]
}
```

Tras restablecer la contraseña del usuario maestro, se aprobará la comprobación previa y podrá volver a intentar la actualización.

En el siguiente ejemplo se utiliza la AWS CLI para restablecer la contraseña. Los cambios de contraseña se aplican inmediatamente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --master-user-password my-new-password
```

A continuación, la comprobación previa se realiza correctamente.

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  title": "Check if master user exists",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForPrefixIndexOnGeometryColumns

Nivel de comprobación previa: error

Comprobación de si hay columnas geométricas en los índices de prefijos

A partir de la versión 8.0.12 de MySQL, ya no se puede crear un índice con prefijo en una columna con el tipo de datos GEOMETRY. Para obtener más información, consulte WL#11808.

Si existe alguno de estos índices, se producirá un error en la actualización. Para resolver el problema, elimine los índices con prefijo de `GEOMETRY` en las tablas mencionadas en el error de la comprobación previa.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "description": "Consider dropping the prefix Indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.geom_index_prefix",
        "description": "Table `test`.`geom_index_prefix` has an index `LatLon` on
 geometry column/s. Mysql 8.0 does not support this type of index on a geometry
 column https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run
 'DROP INDEX `LatLon` ON `test`.`geom_index_prefix`;"
      }
  ]
}
```

La comprobación previa indica que hay un error porque la tabla `test.geom_index_prefix` incluye un índice con un prefijo en una columna `GEOMETRY`.

Tras eliminar este índice, la comprobación previa se realizará correctamente.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForSpecialCharactersInProcedures

Nivel de comprobación previa: error

Comprobación de si hay incoherencias relacionadas con los caracteres especiales en los procedimientos almacenados

En las versiones anteriores a MySQL 8.0, los nombres de bases de datos, los nombres de tablas y otros objetos correspondían a los archivos del directorio de datos, como, por ejemplo, los metadatos basados en archivos. Como parte de la actualización a MySQL 8.0, todos los objetos de la base de datos se migran a las nuevas tablas del diccionario de datos interno que

se almacenan en el esquema de `mysql` para admitir el Atomic Data Dictionary recientemente implementado. Como parte de la migración de los procedimientos almacenados, la definición y el cuerpo del procedimiento se validan a medida que se van incorporando al nuevo diccionario de datos.

En las versiones anteriores a MySQL 8, era posible, en algunos casos, crear rutinas almacenadas o insertar directamente en la tabla `mysql.proc` procedimientos que incluían caracteres especiales. Por ejemplo, se podía crear un procedimiento almacenado que incluyera un comentario con el carácter de espacio no divisible y no compatible `\xa0`. Si se detecta alguno de estos procedimientos, se producirá un error en la actualización.

Esta comprobación previa valida que los cuerpos y las definiciones de los procedimientos almacenados no incluyan dichos caracteres. Para poder continuar con la actualización, vuelva a crear estos procedimientos almacenados sin ningún carácter oculto ni especial.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "description": "Following procedure(s) has special characters inconsistency.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "information_schema.routines",
        "description": "Data Dictionary Metadata is inconsistent for the procedure
 `get_version_proc` in the database `test` due to usage of special characters in
 procedure body. To avoid that, drop and recreate the procedure without any special
 characters before proceeding with the Upgrade."
      }
  ]
}
```

La comprobación previa indica que el clúster de base de datos incluye un procedimiento denominado `get_version_proc` en la base de datos de `test` que incluye caracteres especiales en el cuerpo del procedimiento.

Tras eliminar y volver a crear el procedimiento almacenado, la comprobación previa se realiza correctamente, lo que permite continuar con la actualización.

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForSysSchemaObjectTypeMismatch

Nivel de comprobación previa: error

Comprobación de si el tipo de objeto no coincide con el esquema **`sys`**

El esquema sys es un conjunto de objetos y vistas en una base de datos de MySQL que ayuda a los usuarios a solucionar problemas, optimizar y supervisar las instancias de base de datos. Al realizar una actualización de la versión principal de la versión 2 a la versión 3 de Aurora MySQL, las vistas del esquema `sys` se vuelven a crear y actualizar a las nuevas definiciones de la versión 3 de Aurora MySQL.

Como parte de la actualización, si algún objeto del esquema `sys` se define mediante motores de almacenamiento (`sys_config/BASE TABLE` en INFORMATION_SCHEMA.TABLES) y no mediante vistas, se producirá un error en la actualización. Estas tablas pueden encontrarse en la tabla `information_schema.tables`. Este no es el comportamiento esperado, pero en algunos casos puede ocurrir debido a un error del usuario.

Esta comprobación previa valida todos los objetos del esquema `sys` para garantizar que utilizan las definiciones de tabla correctas y que las vistas no se definen erróneamente como tablas de InnoDB o MyISAM. Para resolver el problema, corrija manualmente los objetos devueltos cambiándoles el nombre o eliminándolos. A continuación, vuelva a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "description": "Database contains objects with type mismatch for sys schema.",
  "detectedProblems": [
      {
```

```
        "level": "Error",
        "dbObject": "sys.waits_global_by_latency",
        "description": "Your object sys.waits_global_by_latency has a type mismatch.
 To fix the inconsistency we recommend to rename or remove the object before
 upgrading (use RENAME TABLE command). "
      }
  ]
}
```

La comprobación previa indica que la vista sys.waits_global_by_latency del esquema `sys` presenta una discrepancia de tipos que impide que se continúe con la actualización.

Tras iniciar sesión en la instancia de base de datos, puede ver que este objeto se define como una tabla de InnoDB, cuando debería ser una vista.

```
mysql> show create table sys.waits_global_by_latency\G
*************************** 1. row ***************************
       Table: waits_global_by_latency
Create Table: CREATE TABLE `waits_global_by_latency` (
  `events` varchar(128) DEFAULT NULL,
  `total` bigint(20) unsigned DEFAULT NULL,
  `total_latency` text,
  `avg_latency` text,
  `max_latency` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Para resolver este problema, podemos eliminar y volver a crear la vista con la definición correcta o cambiarle el nombre. Durante el proceso de actualización, se creará automáticamente con la definición de tabla correcta.

```
mysql> RENAME TABLE sys.waits_global_by_latency to sys.waits_global_by_latency_old;
Query OK, 0 rows affected (0.01 sec)
```

Una vez hecho esto, se aprueba la comprobación previa.

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "detectedProblems": []
```

```
}
```

auroraUpgradeCheckForViewColumnNameLength

Nivel de comprobación previa: error

Comprobación del límite superior del nombre de columna en la vista

La longitud máxima permitida de un nombre de columna en MySQL es de 64 characters. En las versiones anteriores a MySQL 8.0, en algunos casos era posible crear una vista con un nombre de columna de más de 64 caracteres. Si existe alguna de estas vistas en la instancia de la base de datos, se mostrará un error de comprobación previa y no se realizará correctamente la actualización. Para poder continuar con la actualización, debe volver a crear las vistas en cuestión, asegurándose de que la longitud de sus columnas sea inferior a 64 caracteres. A continuación, vuelva a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "description": "Following view(s) has column(s) with length greater than 64.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject":
 "test.colname_view_test.col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pa
        "description": "View `test`.`colname_view_test`has column
 `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` with
 invalid column name length. To avoid Upgrade errors, view should be altered by
 renaming the column name so that its length is not 0 and doesn't exceed 64."
      }
  ]
}
```

La comprobación previa indica que la vista `test.colname_view_test` incluye una columna `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` que supera la longitud máxima permitida de 64 caracteres.

Si observamos la definición de la vista, podemos ver la columna problemática.

```
mysql> desc `test`.`colname_view_test`;
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| Field                                                             | Type        |
 Null | Key | Default | Extra |
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| col1                                                              | varchar(20) |
 YES  |     | NULL    |       |
| col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad | int(11)     |
 YES  |     | NULL    |       |
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Para poder continuar con la actualización, vuelva a crear la vista y asegúrese de que la longitud de la columna no supere los 64 caracteres.

```
mysql> drop view `test`.`colname_view_test`;
Query OK, 0 rows affected (0.01 sec)

mysql> create view `test`.`colname_view_test`(col1, col2_nopad) as select inf,
 fodcol from test;
Query OK, 0 rows affected (0.01 sec)

mysql> desc `test`.`colname_view_test`;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| col1       | varchar(20) | YES  |     | NULL    |       |
| col2_nopad | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Una vez hecho esto, la comprobación previa se realizará correctamente.

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckIndexLengthLimitOnTinyColumns

Nivel de comprobación previa: error

Comprobación de si hay tablas con índices definidos con una longitud de prefijo superior a 255 bytes en columnas pequeñas

Al crear un índice en una columna con un tipo de datos binario en MySQL, debe añadir una longitud de prefijo en la definición del índice. En las versiones anteriores a MySQL 8.0, en algunos casos era posible especificar una longitud de prefijo superior al tamaño máximo permitido para estos tipos de datos. Un ejemplo son las columnas `TINYTEXT` y `TINYBLOB`, en las que el tamaño máximo de datos permitido es de 255 bytes, pero se permitían prefijos de índice mayores. Para obtener más información, consulte la sección InnoDB limits en la documentación de MySQL.

Si esta comprobación previa no se realiza correctamente, elimine el índice problemático o reduzca la longitud del prefijo de las columnas `TINYTEXT` y `TINYBLOB` del índice a menos de 255 bytes. A continuación, vuelva a intentar la actualización.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "description": "Prefix length of the indexes defined on tiny columns cannot
 exceed 255 bytes. With utf8mb4 char set, this limits the prefix length supported
 upto 63 characters only. A larger prefix length was allowed in MySQL5.7 using
 innodb_large_prefix parameter. This parameter is deprecated in MySQL 8.0.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/innodb-limits.html,
 https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.tintxt_prefixed_index.col1",
        "description": "Index 'PRIMARY' on tinytext/tinyblob column `col1` of table
 `test.tintxt_prefixed_index` is defined with prefix length exceeding 255 bytes.
 Reduce the prefix length to <=255 bytes depending on character set used. For
 utf8mb4, it should be <=63."
      }
  ]
}
```

La comprobación previa indica que hay un error en la tabla `test.tintxt_prefixed_index`, ya que tiene un índice `PRIMARY` con un prefijo superior a 255 bytes en una columna TINYTEXT o TINYBLOB.

Si observamos la definición de esta tabla, podemos ver que la clave principal tiene el prefijo 65 en la columna `TINYTEXT col1`. Dado que la tabla se define mediante el conjunto de caracteres `utf8mb4`, que almacena 4 bytes por carácter, el prefijo supera el límite de 255 bytes.

```
mysql> show create table `test`.`tintxt_prefixed_index`\G
*************************** 1. row ***************************
       Table: tintxt_prefixed_index
Create Table: CREATE TABLE `tintxt_prefixed_index` (
  `col1` tinytext NOT NULL,
  `col2` tinytext,
  `col_id` tinytext,
  PRIMARY KEY (`col1`(65))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=DYNAMIC
1 row in set (0.00 sec)
```

Si se modifica el prefijo del índice a 63 mientras se utiliza el conjunto de caracteres `utf8mb4`, se podrá continuar con la actualización.

```
mysql> alter table `test`.`tintxt_prefixed_index` drop PRIMARY KEY, ADD  PRIMARY KEY
 (`col1`(63));
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Una vez hecho esto, la comprobación previa se realizará correctamente.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

Nivel de comprobación previa: error

Comprobación de la incoherencia de los metadatos de InnoDB que faltan en la tabla **`mysql.host`**

Se trata de una comprobación previa exclusivamente interna llevada a cabo por el servicio de RDS. Los errores se gestionarán automáticamente en la actualización y pueden omitirse de forma segura.

Si detecta algún error con esta comprobación previa, abra un caso con AWS Support para solicitar que se resuelva la incoherencia de los metadatos. También puede volver a intentar la actualización. Para ello, debe realizar un volcado lógico y, a continuación, restaurar a un nuevo clúster de base de datos que ejecute la versión 3 de Aurora MySQL.

auroraUpgradeCheckForInvalidUtf8mb3ColumnComments

Nivel de comprobación previa: error

Comprobación de si hay comentarios de columnas utf8mb3 no válidos

Esta comprobación previa identifica las tablas que contienen comentarios de columnas con una codificación de caracteres utf8mb3 no válida. En MySQL 8.0, se aplica una validación más estricta a la codificación de caracteres en los metadatos, incluidos los comentarios de las columnas. Si algún comentario de columna contiene caracteres que no son válidos en el conjunto de caracteres utf8mb3, la actualización producirá un error.

La causa más común de este problema es la presencia de caracteres que no pertenecen a BMP (Plano Básico Multilingüe) en los comentarios de las columnas. Los caracteres que no pertenecen al BMP requieren codificación UTF-8 de 4 bytes (utf8mb4), pero utf8mb3 solo admite secuencias de 3 bytes, que no pueden representar estos caracteres.

Para resolver este problema, debe modificar los comentarios de las columnas para eliminar o reemplazar los caracteres que no pertenezcan al BMP antes de intentar la actualización. Puede usar la instrucción `ALTER TABLE` para actualizar los comentarios de las columnas.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "description": "Following table(s) has/have invalid utf8mb3 comments on the
 column/columns.",
  "detectedProblems": [
```

```
      {
        "level": "Error",
        "dbObject": "test.t2",
        "description": "Table test.t2 has invalid utf8mb3 comments in it's column/
columns. This is due to non-BMP characters in the comment field. To fix the
 inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
      }
  ]
}
```

La comprobación previa indica que la tabla `test.t2` contiene caracteres utf8mb3 no válidos en uno o más comentarios de columna, específicamente debido a la presencia de caracteres que no son del BMP.

Para resolver este problema, puede identificar las columnas problemáticas y actualizar sus comentarios. Primero, examine la estructura de la tabla para identificar las columnas con comentarios:

```
mysql> SHOW CREATE TABLE test.t2\G
```

Una vez que haya identificado las columnas con comentarios problemáticos, actualícelas mediante la instrucción `ALTER TABLE`. Por ejemplo:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT 'Updated
 comment without non-BMP characters';
```

Alternativamente, puede eliminar el comentario por completo:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT '';
```

Tras actualizar todos los comentarios problemáticos de las columnas, se aprobará la comprobación previa y la actualización podrá continuar:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "detectedProblems": []
}
```

> **Note**
>
> Antes de modificar los comentarios de las columnas, asegúrese de que cualquier código de aplicación o documentación que se base en estos comentarios se actualice en consecuencia. Considere la posibilidad de migrar al conjunto de caracteres utf8mb4 para mejorar la compatibilidad con Unicode si la aplicación requiere caracteres que no sean del BMP.

Advertencias

Las siguientes comprobaciones previas generan advertencias cuando se produce un error en la comprobación previa, pero se puede continuar con la actualización.

Temas

- Comprobaciones previas de MySQL que informan de advertencias
- Comprobaciones previas de Aurora MySQL que informan de advertencias

Comprobaciones previas de MySQL que informan de advertencias

Las siguientes comprobaciones previas proceden de Community MySQL:

- defaultAuthenticationPlugin
- maxdbFlagCheck
- mysqlDollarSignNameCheck
- reservedKeywordsCheck
- utf8mb3Check
- zeroDatesCheck

defaultAuthenticationPlugin

Nivel de comprobación previa: advertencia

Aspectos a tener en cuenta sobre el nuevo complemento de autenticación predeterminado

En MySQL 8.0, se ha introducido el complemento de autenticación `caching_sha2_password`, que proporciona un cifrado de contraseñas más seguro y un mejor rendimiento que el

complemento obsoleto `mysql_native_password`. En el caso de la versión 3 de Aurora MySQL, el complemento de autenticación predeterminado que se usa para los usuarios de bases de datos es `mysql_native_password`.

Esta comprobación previa advierte que este complemento se eliminará y se cambiará el predeterminado en una futura versión principal. Plantéese la posibilidad de evaluar la compatibilidad de los clientes y usuarios de la aplicación antes de realizar este cambio.

Para obtener más información, consulte la sección caching_sha2_password Compatibility Issues and Solutions en la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "defaultAuthenticationPlugin",
  "title": "New default authentication plugin considerations",
  "description": "Warning: The new default authentication plugin
 'caching_sha2_password' offers more secure password hashing than previously
 used 'mysql_native_password' (and consequent improved client connection
 authentication). However, it also has compatibility implications that
 may affect existing MySQL installations. If your MySQL installation
 must serve pre-8.0 clients and you encounter compatibility issues after
 upgrading, the simplest way to address those issues is to reconfigure
 the server to revert to the previous default authentication plugin
 (mysql_native_password). For example, use these lines in the server option file:
\n\n[mysqld]\ndefault_authentication_plugin=mysql_native_password\n\nHowever, the
 setting should be viewed as temporary, not as a long term or permanent solution,
 because it causes new accounts created with the setting in effect to forego the
 improved authentication security.\nIf you are using replication please take time to
 understand how the authentication plugin changes may impact you.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-caching-sha2-password-compatibility-issues\nhttps://
dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-
sha2-password-replication"
},
```

maxdbFlagCheck

Nivel de comprobación previa: advertencia

Uso de un indicador de **`MAXDB`** obsoleto de **`sql_mode`**

En MySQL 8.0, se han eliminado una serie de opciones de variables del sistema sql_mode; una de ellos era MAXDB. Esta comprobación previa examina todas las sesiones actualmente conectadas, junto con las rutinas y los desencadenadores, para garantizar que ninguna de ellas tenga el parámetro sql_mode establecido en ninguna combinación que incluya MAXDB.

Ejemplo de salida:

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "description": "Warning: The following DB objects have the obsolete MAXDB
 option persisted for sql_mode, which will be cleared during the upgrade. It can
 potentially change the datatype DATETIME into TIMESTAMP if it is used inside
 object's definition, and this in turn can change the behavior in case of dates
 earlier than 1970 or later than 2037. If this is a concern, please redefine these
 objects so that they do not rely on the MAXDB flag before running the upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.maxdb_stored_routine",
        "description": "PROCEDURE uses obsolete MAXDB sql_mode",
        "dbObjectType": "Routine"
      }
  ]
}
```

La comprobación previa indica que la rutina test.maxdb_stored_routine incluye una opción sql_mode no compatible.

Tras iniciar sesión en la base de datos, podrá ver la definición de rutina en la que sql_mode incluye MAXDB.

```
 > SHOW CREATE PROCEDURE test.maxdb_stored_routine\G

*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode:
 PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,MAXDB,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPT
```

```
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Para resolver el problema, vuelva a crear el procedimiento después de configurar el parámetro `sql_mode` correcto en el cliente.

> **Note**
>
> Según la documentación de MySQL, MySQL almacena la configuración de `sql_mode` que está en vigor cuando se crea o modifica una rutina. Siempre ejecuta la rutina con esta configuración, independientemente de la configuración de `sql_mode` cuando se inicia la rutina.
>
> Antes de cambiar `sql_mode`, consulte la sección Server SQL Modes en la documentación de MySQL. Evalúe detenidamente cualquier posible impacto que esto pueda tener en la aplicación.

Vuelva a crear el procedimiento sin el parámetro `sql_mode` no compatible.

```
mysql > set sql_mode='PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql > DROP PROCEDURE test.maxdb_stored_routine\G
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER $$
mysql >
mysql > CREATE PROCEDURE test.maxdb_stored_routine()
    -> SQL SECURITY DEFINER
    -> BEGIN
    ->     SELECT * FROM test;
    -> END$$
Query OK, 0 rows affected (0.00 sec)
```

```
mysql >
mysql > DELIMITER ;
mysql > show create procedure test.maxdb_stored_routine\G
*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

La comprobación previa se realiza correctamente.

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "detectedProblems": []
}
```

mysqlDollarSignNameCheck

Nivel de comprobación previa: advertencia

Comprobación de si el uso de signos de dólar únicos está obsoleto en los nombres de objetos

A partir de la versión 8.0.32 de MySQL, el uso del signo de dólar ($) como primer carácter de un identificador sin comillas está obsoleto. Si tiene algún esquema, tabla, vista, columna o rutina que incluya un $ como primer carácter, esta comprobación previa mostrará una advertencia. Si bien esta advertencia no impide que se lleve a cabo la actualización, le recomendamos que tome medidas para resolverlo lo antes posible. A partir de MySQL 8.4, cualquier identificador de este tipo devolverá un error de sintaxis en lugar de una advertencia.

Ejemplo de salida:

```
{
  "id": "mysqlDollarSignNameCheck",
```

```
  "title": "Check for deprecated usage of single dollar signs in object names",
  "status": "OK",
  "description": "Warning: The following objects have names with deprecated usage
 of dollar sign ($) at the begining of the identifier. To correct this warning,
 ensure, that names starting with dollar sign, also end with it, similary to quotes
 ($example$). ",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.$deprecated_syntx",
        "description": " name starts with $ sign."
      }
  ]
},
```

La comprobación previa muestra una advertencia porque la tabla `$deprecated_syntx` del esquema `test` incluye un $ como primer carácter.

reservedKeywordsCheck

Nivel de comprobación previa: advertencia

Uso de objetos de base de datos con nombres que entran en conflicto con las nuevas palabras clave reservadas

Esta comprobación es similar a la de routineSyntaxCheck, ya que comprueba el uso de objetos de base de datos cuyos nombres entran en conflicto con las nuevas palabras clave reservadas. Aunque no impiden las actualizaciones, es necesario evaluar detenidamente las advertencias.

Ejemplo de salida:

Si utilizamos el ejemplo anterior con la tabla denominada `except`, la comprobación previa mostrará una advertencia:

```
{
  "id": "reservedKeywordsCheck",
  "title": "Usage of db objects with names conflicting with new reserved keywords",
  "status": "OK",
  "description": "Warning: The following objects have names that conflict with
 new reserved keywords. Ensure queries sent by your applications use `quotes` when
 referring to them or they will result in errors.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
```

```
        "level": "Warning",
        "dbObject": "test.except",
        "description": "Table name",
        "dbObjectType": "Table"
      }
  ]
}
```

Esta advertencia le permite saber que es posible que haya que revisar algunas consultas de la aplicación. Si las consultas de la aplicación no aplican correctamente caracteres de escape a los literales de cadena, es posible que se produzcan errores después de actualizar a MySQL 8.0. Revise las aplicaciones para confirmar esto y compruébelas con un clon o una instantánea del clúster de base de datos de Aurora MySQL que se ejecute en la versión 3.

Ejemplo de una consulta de aplicación a la que no se le hayan aplicado caracteres de escape que no se realizará tras la actualización:

```
SELECT * FROM escape;
```

Ejemplo de una consulta de aplicación a la que se le hayan aplicado correctamente caracteres de escape que se realizará tras la actualización:

```
SELECT * FROM 'escape';
```

utf8mb3Check

Nivel de comprobación previa: advertencia

Uso del conjunto de caracteres **`utf8mb3`**

En MySQL 8.0, el conjunto de caracteres `utf8mb3` está obsoleto y se eliminará en una futura versión principal de MySQL. Esta comprobación previa se implementa para emitir una advertencia si se detecta algún objeto de la base de datos que utilice este conjunto de caracteres. Si bien esto no impedirá que se continúe con la actualización, le recomendamos encarecidamente que piense en migrar las tablas al conjunto de caracteres `utf8mb4`, que es el predeterminado a partir de MySQL 8.0. Para obtener más información sobre utf8mb3 y utf8mb4, consulte Converting between 3-byte and 4-byte Unicode character sets en la documentación de MySQL.

Ejemplo de salida:

```
{
```

```
  "id": "utf8mb3",
  "title": "Usage of utf8mb3 charset",
  "status": "OK",
  "description": "Warning: The following objects use the deprecated utf8mb3
 character set. It is recommended to convert them to use utf8mb4 instead, for
 improved Unicode support. The utf8mb3 character is subject to removal in the
 future.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/charset-unicode-
utf8mb3.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.t1.col1",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      },
      {
        "level": "Warning",
        "dbObject": "test.t1.col2",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      }
  ]
}
```

Para resolver este problema, vuelva a crear los objetos y las tablas a los que se ha hecho referencia. Para obtener más información y conocer los requisitos previos antes de hacerlo, consulte la sección Converting between 3-byte and 4-byte Unicode character sets en la documentación de MySQL.

zeroDatesCheck

Nivel de comprobación previa: advertencia

Valores cero de fecha, fecha y hora y marca de tiempo

MySQL ahora aplica reglas más estrictas con respecto al uso de valores cero en las columnas de fecha, fecha y hora y marca de tiempo. Le recomendamos que utilice los modos `NO_ZERO_IN_DATE` y `NO_ZERO_DATE SQL` junto con el modo `strict`, ya que se fusionarán con el modo `strict` en una versión futura de MySQL.

Si la configuración de `sql_mode` de alguna de las conexiones de la base de datos, en el momento de ejecutar la comprobación previa, no incluye estos modos, aparecerá una advertencia

en la comprobación previa. Es posible que los usuarios aún puedan insertar los valores de fecha, fecha y hora y marca de tiempo que incluyan valores cero. Sin embargo, recomendamos encarecidamente sustituir los valores cero por valores válidos, ya que su comportamiento podría cambiar en el futuro y puede que no funcionen correctamente. Como se trata de una advertencia, no impedirá las actualizaciones, pero le recomendamos que empiece a planificar las medidas necesarias.

Ejemplo de salida:


```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
  ]
}
```


Comprobaciones previas de Aurora MySQL que informan de advertencias

Las siguientes comprobaciones previas son específicas de Aurora MySQL:

- auroraUpgradeCheckForRollbackSegmentHistoryLength

- auroraUpgradeCheckForUncommittedRowModifications

auroraUpgradeCheckForRollbackSegmentHistoryLength

Nivel de comprobación previa: advertencia

Comprueba si la longitud de la lista del historial de segmentos de reversión del clúster es alta

Tal y como se ha mencionado en auroraUpgradeCheckForIncompleteXATransactions, mientras se ejecuta el proceso de actualización de la versión principal, es esencial que la instancia de base de datos de la versión 2 de Aurora MySQL se cierre por completo. Esto garantiza que todas las transacciones se confirmen o se reviertan y que InnoDB haya purgado todos los registros de deshacer.

Si el clúster de base de datos tiene una longitud de la lista del historial (HLL) de segmentos de reversión larga, puede prolongarse el tiempo que InnoDB tarda en completar la purga de los registros de deshacer, lo que provoca un tiempo de inactividad prolongado durante el proceso de actualización de la versión principal. Si la comprobación previa detecta que la HLL del clúster de base de datos es larga, se generará una advertencia. Si bien esto no impide que se continúe con la actualización, le recomendamos que supervise de cerca la HLL del clúster de base de datos. Al mantenerla en niveles bajos, se reducirá el tiempo de inactividad necesario durante una actualización de la versión principal. Para obtener más información sobre la supervisión de HLL, consulte La longitud de la lista de historial de InnoDB ha aumentado de forma significativa.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForRollbackSegmentHistoryLength",
  "title": "Checks if the rollback segment history length for the cluster is high",
  "status": "OK",
  "description": "Rollback Segment History length is greater than 1M. Upgrade may
 take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_metrics",
        "description": "The InnoDB undo history list length('trx_rseg_history_len')
 is 82989114. Upgrade may take longer due to purging of undo information for old row
 versions."
      }
  ]
```

```
}
```

La comprobación previa devuelve una advertencia porque ha detectado que la HLL de deshacer de InnoDB era larga en el clúster de la base de datos (82989114). Aunque se continúe con la actualización, en función de la cantidad de operaciones de deshacer que se deba purgar, puede prolongar el tiempo de inactividad necesario durante el proceso de actualización.

Le recomendamos que investigue las transacciones abiertas en el clúster de base de datos antes de ejecutar la actualización en producción para asegurarse de que la HLL se mantenga en un tamaño manejable.

auroraUpgradeCheckForUncommittedRowModifications

Nivel de comprobación previa: advertencia

Comprueba si hay muchas modificaciones de fila sin confirmar

Tal y como se ha mencionado en auroraUpgradeCheckForIncompleteXATransactions, mientras se ejecuta el proceso de actualización de la versión principal, es esencial que la instancia de base de datos de la versión 2 de Aurora MySQL se cierre por completo. Esto garantiza que todas las transacciones se confirmen o se reviertan y que InnoDB haya purgado todos los registros de deshacer.

Si el clúster de base de datos tiene transacciones que han modificado un gran número de filas, esto puede prolongar el tiempo que InnoDB tarda en completar la reversión de esta transacción como parte del proceso de cierre completo. Si la comprobación previa detecta transacciones de larga duración, con un gran número de filas modificadas en el clúster de base de datos, se generará una advertencia. Si bien esto no impide que se continúe con la actualización, le recomendamos que supervise de cerca el tamaño de las transacciones activas del clúster de la base de datos. Al mantenerla en niveles bajos, se reducirá el tiempo de inactividad necesario durante una actualización de la versión principal.

Ejemplo de salida:

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "description": "Database contains uncommitted row changes greater than 10M.
 Upgrade may take longer time.",
  "detectedProblems": [
      {
```

```
        "level": "Warning",
        "dbObject": "information_schema.innodb_trx",
        "description": "The database contains 11000000 uncommitted row change(s) in
 1 transaction(s). Upgrade may take longer due to transaction rollback."
      }
  ]
},
```

La comprobación previa indica que el clúster de base de datos incluye una transacción con 11 000 000 de cambios de fila no confirmados que deberán revertirse durante el proceso de cierre completo. La actualización continuará, pero para reducir el tiempo de inactividad durante el proceso de actualización, le recomendamos que lo supervise e investigue antes de ejecutar la actualización en los clústeres de producción.

Para ver las transacciones activas en la instancia de base de datos de escritor, puede utilizar la tabla information_schema.innodb_trx. La siguiente consulta en la instancia de base de datos de escritor muestra las transacciones actuales, el tiempo de ejecución, el estado y las filas modificadas del clúster de la base de datos.

```
# Example of uncommitted transaction
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+-----------------------------
+-------------------------------+----------+-----------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state | rows_modified |
+---------------------+-----------------------------
+-------------------------------+----------+-----------+---------------+
| 2024-08-12 18:32:52 |                          1592 |
 20041 | 52866130 | RUNNING   |      11000000 |
+---------------------+-----------------------------
+-------------------------------+----------+-----------+---------------+
1 row in set (0.01 sec)

# Example of transaction rolling back
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
```

```
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+-----------------------------
+---------------------------------+----------+-------------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state    | rows_modified |
+---------------------+-----------------------------
+---------------------------------+----------+-------------+---------------+
| 2024-08-12 18:32:52 |                          1719 |
 20041 | 52866130 | ROLLING BACK |      10680479 |
+---------------------+-----------------------------
+---------------------------------+----------+-------------+---------------+
1 row in set (0.01 sec)
```

Una vez confirmada o revertida la transacción, la comprobación previa ya no mostrará ninguna advertencia. Consulte la documentación de MySQL y hable con el equipo de aplicaciones antes de revertir cualquier transacción grande, ya que la reversión puede tardar algún tiempo en completarse, según el tamaño de la transacción.

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "detectedProblems": []
},
```

Para obtener más información sobre la optimización de la administración de transacciones de InnoDB y el posible impacto al ejecutar y revertir transacciones de gran tamaño en instancias de bases de datos de MySQL, consulte la sección Optimizing InnoDB Transaction Management en la documentación de MySQL.

Avisos

La siguiente comprobación previa genera un aviso cuando se produce un error en ella, pero se puede continuar con la actualización.

sqlModeFlagCheck

Nivel de comprobación previa: aviso

Uso de indicadores de **`sql_mode`** obsoletos

Además de `MAXDB`, se ha eliminado una serie de otras opciones de `sql_mode`: `DB2`, `MSSQL`, `MYSQL323`, `MYSQL40`, `ORACLE`, `POSTGRESQL`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS` y `NO_TABLE_OPTIONS`. A partir de MySQL 8.0, ninguno de estos valores se puede asignar a la variable del sistema `sql_mode`. Si esta comprobación previa detecta sesiones abiertas con esta configuración de `sql_mode`, asegúrese de que los grupos de parámetros de la instancia de base de datos y del clúster de base de datos, así como las aplicaciones y configuraciones del cliente, estén actualizados para deshabilitarlos. Para obtener más información, consulte la documentación de MySQL.

Ejemplo de salida:

```
{
  "id": "sqlModeFlagCheck",
  "title": "Usage of obsolete sql_mode flags",
  "status": "OK",
  "detectedProblems": []
}
```

Para resolver alguno de estos errores de comprobación previa, consulte maxdbFlagCheck.

Errores, advertencias o avisos

La siguiente comprobación previa puede devolver un error, una advertencia o un aviso en función del resultado de la comprobación previa.

checkTableOutput

Nivel de comprobación previa: error, advertencia o aviso

Problemas notificados por el comando **`check table x for upgrade`**

Antes de iniciar la actualización a la versión 3 de Aurora MySQL, se ejecuta `check table for upgrade` en cada tabla de los esquemas de usuarios del clúster de la base de datos. Esta comprobación previa no es la misma que la de checkTableMysqlSchema.

El comando `check table for upgrade` examina las tablas para detectar posibles problemas que puedan surgir durante una actualización a una versión más reciente de MySQL. Ejecutar este comando antes de intentar una actualización puede ayudar a identificar y resolver cualquier incompatibilidad con antelación, lo que facilita el proceso de actualización propiamente dicho.

Este comando realiza varias comprobaciones en cada tabla, como, por ejemplo:

- Verifica que los metadatos y la estructura de la tabla sean compatibles con la versión de MySQL de destino
- Comprueba si hay alguna característica obsoleta o eliminada que se utilice en la tabla
- Garantiza que la tabla se pueda actualizar correctamente sin que se pierdan datos

A diferencia de otras comprobaciones previas, puede devolver un error, una advertencia o un aviso en función del resultado de `check table`. Si esta comprobación previa devuelve alguna tabla, revísela detenidamente, junto con el código de devolución y el mensaje antes de iniciar la actualización. Para obtener más información, consulte la sección CHECK TABLE statement en la documentación de MySQL.

A continuación, ofrecemos un ejemplo de error y un ejemplo de advertencia.

Ejemplo de error:

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.parent",
        "description": "Table 'test.parent' doesn't exist"
      }
  ]
},
```

La comprobación previa indica que hay un error porque la tabla `test.parent` no existe.

El archivo `mysql-error.log` de la instancia de base de datos de escritor muestra que hay un error de clave externa.

```
2024-08-13T15:32:10.676893Z 62 [Warning] InnoDB: Load table `test`.`parent` failed,
 the table has missing foreign key indexes. Turn off 'foreign_key_checks' and try
 again.
2024-08-13T15:32:10.676905Z 62 [Warning] InnoDB: Cannot open table test/parent from
 the internal data dictionary of InnoDB though the .frm file for the table exists.
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-troubleshooting.html
 for how to resolve the issue.
```

Inicie sesión en la instancia de base de datos de escritor y ejecute `show engine innodb status\G` para obtener más información sobre el error de clave externa.

```
mysql> show engine innodb status\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2024-08-13 15:33:33 0x14ef7b8a1700 INNODB MONITOR OUTPUT
=====================================
.
.
.
------------------------
LATEST FOREIGN KEY ERROR
------------------------
2024-08-13 15:32:10 0x14ef6dbbb700 Error in foreign key constraint of table test/
child:
there is no index in referenced table which would contain
the columns as the first columns, or the data types in the
referenced table do not match the ones in table. Constraint:
,
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
The index in the foreign key in table is p_name_idx
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-foreign-key-
constraints.html for correct foreign key definition.
.
.
```

El mensaje `LATEST FOREIGN KEY ERROR` indica que a la restricción de clave externa `fk_pname` de la tabla `test.child`, que hace referencia a la tabla `test.parent`, le falta un índice o el tipo de datos no coincide. La documentación de MySQL sobre las restricciones de claves externas establece que las columnas a las que se hace referencia en una clave externa

deben tener un índice asociado y las columnas principales o secundarias deben usar el mismo tipo de datos.

Para verificar si esto se debe a la falta de un índice o a que el tipo de datos no coincide, inicie sesión en la base de datos y compruebe las definiciones de la tabla deshabilitando temporalmente la variable de sesión foreign_key_checks. Después de hacerlo, podemos ver que la restricción secundaria en cuestión (`fk_pname`) utiliza `p_name varchar(20) CHARACTER SET latin1 DEFAULT NULL` para hacer referencia a la tabla principal `name varchar(20) NOT NULL`. La tabla principal utiliza `DEFAULT CHARSET=utf8`, pero la columna `p_name` de la tabla secundaria utiliza `latin1`, por lo que se produce un error de incoherencia entre los tipos de datos.

```
mysql> show create table parent\G
ERROR 1146 (42S02): Table 'test.parent' doesn't exist

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> set foreign_key_checks=0;
Query OK, 0 rows affected (0.00 sec)

mysql> show create table parent\G
*************************** 1. row ***************************
       Table: parent
Create Table: CREATE TABLE `parent` (
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
```

```
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Para resolver este problema, podemos cambiar la tabla secundaria para que use el mismo conjunto de caracteres que la tabla principal o cambiar la tabla principal para que use el mismo conjunto de caracteres que la tabla secundaria. En este caso, dado que la tabla secundaria usa explícitamente `latin1` en la definición de columna `p_name`, ejecutamos `ALTER TABLE` para modificar el conjunto de caracteres a `utf8`.

```
mysql> alter table child modify p_name varchar(20) character set utf8 DEFAULT NULL;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> flush tables;
Query OK, 0 rows affected (0.01 sec)
```

Después de hacerlo, se aprueba la comprobación previa y se puede continuar con la actualización.

Ejemplo de advertencia:

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.orders",
        "description": "Trigger test.orders.delete_audit_trigg does not have CREATED
 attribute."
      }
  ]
}
```

La comprobación previa muestra una advertencia para el desencadenador `delete_audit_trigg` de la tabla `test.orders`, ya que no tiene el atributo `CREATED`. Según la sección Checking Version Compatibility de la documentación de MySQL, este mensaje es informativo y se muestra para los desencadenadores creados en las versiones anteriores a MySQL 5.7.2.

Dado que se trata de una advertencia, no impide que se lleve a cabo la actualización. Sin embargo, si desea resolver el problema, puede volver a crear el desencadenador en cuestión, y la comprobación previa se realizará correctamente sin ningún tipo de advertencia.

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": []
},
```

Pasos para realizar una actualización local

También le recomendamos que revise el material de referencia en Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL.

Realice cualquier planificación y prueba previas a la actualización, tal y como se describe en Planificación de una actualización de versión principal para un clúster Aurora MySQL.

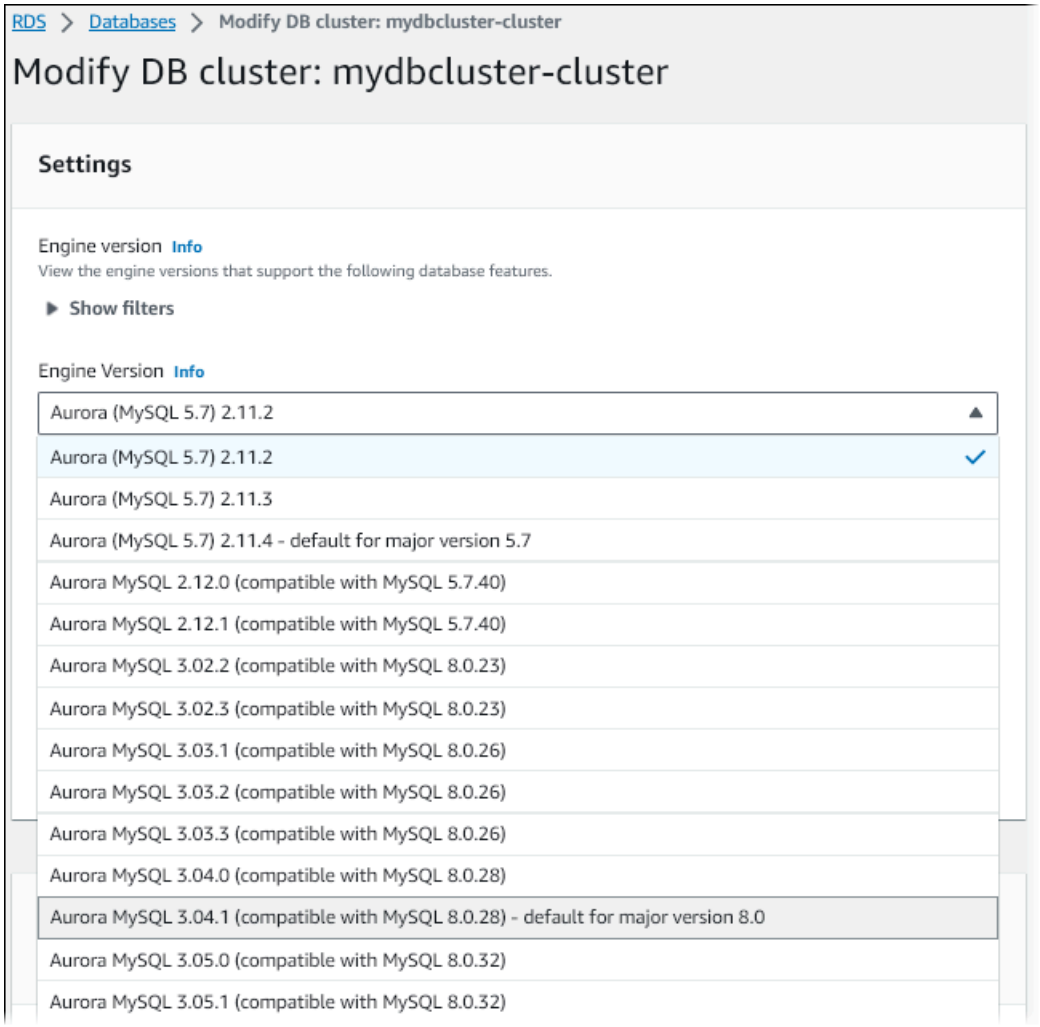
Consola

En el ejemplo siguiente se actualiza el clúster de base de datos `mydbcluster-cluster` a la versión Aurora MySQL 3.04.1.

Para actualizar la versión principal de un clúster de base de datos Aurora MySQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Si utilizó un grupo de parámetros personalizado con el clúster de base de datos original, cree un grupo de parámetros compatible con la nueva versión principal. Realice los ajustes necesarios en los parámetros de configuración del nuevo grupo de parámetros. Para obtener más información, consulte Cómo afectan las actualizaciones en el lugar a los grupos de parámetros de un clúster.
3. En el panel de navegación, seleccione Databases (Bases de datos).
4. En la lista, elija el clúster de base de datos que desea modificar.
5. Elija Modify.
6. Para Version (Versión), elija una nueva versión principal de Aurora.

   También le recomendamos que utilice la versión secundaria de la versión principal. A continuación, elegimos la versión predeterminada actual.



7. Elija Continue.
8. En la página siguiente, especifique cuándo realizar la actualización. Seleccione During the next scheduled maintenance window (Durante la siguiente ventana de mantenimiento programado) o Immediately (Inmediatamente).
9. (Opcional) Examine periódicamente la página Events (Eventos) de la consola RDS durante la actualización. Esto le ayuda a supervisar el progreso de la actualización e identificar cualquier problema. Si la actualización encuentra algún problema, consulte Solución de problemas para la actualización Aurora MySQL en el lugar para conocer los pasos a seguir.
10. Si creó un nuevo grupo de parámetros al inicio de este procedimiento, asocie el grupo de parámetros personalizados con el clúster actualizado. Para obtener más información, consulte Cómo afectan las actualizaciones en el lugar a los grupos de parámetros de un clúster.

> **Note**
>
> Para realizar este paso, deberá reiniciar el clúster de nuevo para aplicar el nuevo grupo de parámetros.

11. (Opcional) Después de completar las pruebas posteriores a la actualización, elimine la instantánea manual que Aurora creó al comienzo de la actualización.

AWS CLI

Para actualizar la versión principal de un clúster de base de datos de Aurora MySQL, utilice el comando modify-db-cluster de la AWS CLI con los siguientes parámetros requeridos:

- `--db-cluster-identifier`
- `--engine-version`
- `--allow-major-version-upgrade`
- `--apply-immediately` o `--no-apply-immediately`

Si el clúster utiliza algún grupo de parámetros personalizados, incluya también una o ambas opciones:

- `--db-cluster-parameter-group-name`, si el clúster utiliza un grupo de parámetros de clúster personalizado
- `--db-instance-parameter-group-name`, si alguna instancia del clúster utiliza un grupo de parámetros de base de datos personalizado

En el ejemplo siguiente se actualiza el clúster de base de datos `sample-cluster` a la versión Aurora MySQL 3.04.1. La actualización se realiza inmediatamente, en lugar de esperar la siguiente ventana de mantenimiento.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --engine-version 8.0.mysql_aurora.3.04.1 \
```

```
          --allow-major-version-upgrade \
          --apply-immediately
```

En Windows:

```
aws rds modify-db-cluster ^
          --db-cluster-identifier sample-cluster ^
          --engine-version 8.0.mysql_aurora.3.04.1 ^
          --allow-major-version-upgrade ^
          --apply-immediately
```

Puede combinar otros comandos de CLI con `modify-db-cluster` para crear un proceso automatizado de extremo a extremo para realizar y verificar actualizaciones. Para obtener más información y ejemplos, consulte Tutorial de actualización de Aurora MySQL en el lugar.

> **Note**
>
> Si el clúster forma parte de una base de datos global Aurora, el procedimiento de actualización en el lugar es ligeramente diferente. Se llama a la operación de comando modify-global-cluster en lugar de `modify-db-cluster`. Para obtener más información, consulte Actualizaciones mayores en el lugar para bases de datos globales.

API de RDS

Para actualizar la versión principal de un clúster de base de datos Aurora MySQL, utilice la operación de la API de RDS ModifyDBCluster con los siguientes parámetros requeridos:

- `DBClusterIdentifier`
- `Engine`
- `EngineVersion`
- `AllowMajorVersionUpgrade`
- `ApplyImmediately` (establecido en `true` o `false`)

> **Note**
>
> Si el clúster forma parte de una base de datos global Aurora, el procedimiento de actualización en el lugar es ligeramente diferente. Se llama a la operación

ModifyGlobalCluster en lugar de `ModifyDBCluster`. Para obtener más información, consulte Actualizaciones mayores en el lugar para bases de datos globales.

Cómo afectan las actualizaciones en el lugar a los grupos de parámetros de un clúster

Los grupos de parámetros de Aurora tienen diferentes conjuntos de opciones de configuración para los clústeres compatibles con MySQL 5.7 u 8.0. Al realizar una actualización en el centro, el clúster actualizado y todas sus instancias deben utilizar los grupos de parámetros de clúster e instancia correspondientes.

Es posible que el clúster y las instancias usen los grupos de parámetros predeterminados compatibles con la versión 5.7. Si es así, el clúster y la instancia actualizados comienzan con los grupos predeterminados de parámetros compatibles con 8.0. Si su clúster e instancias utilizan algún grupo de parámetros personalizado, asegúrese de crear los correspondientes grupos de parámetros compatibles con 8.0. También asegúrese de especificarlos durante el proceso de actualización.

Note

Para la mayoría de las configuraciones de parámetros, puede elegir el grupo de parámetros personalizado en dos puntos. Esto es al crear el clúster o asociar el grupo de parámetros al clúster más adelante.
Sin embargo, si utiliza una configuración no predeterminada para el parámetro `lower_case_table_names`, debe configurar el grupo de parámetros personalizado con esta configuración de antemano. A continuación, especifique el grupo de parámetros durante la restauración de instantáneas para la creación de clúster. Cualquier cambio en el parámetro `lower_case_table_names` no tiene efecto después de crear el clúster.
Le recomendamos que utilice la misma configuración para `lower_case_table_names` cuando actualice de la versión 2 de Aurora MySQL a la versión 3.
Con una base de datos global de Aurora basada en Aurora MySQL, no se puede realizar una actualización local desde la versión 2 a la versión 3 de Aurora MySQL si el parámetro `lower_case_table_names` está activado. Para obtener más información sobre los métodos que puede utilizar, consulte Actualizaciones de la versión principal.

> **Important**
>
> Si especifica algún grupo de parámetros personalizado durante el proceso de actualización, asegúrese de reiniciar manualmente el clúster una vez finalizada la actualización. Al hacerlo, el clúster comienza a usar la configuración de parámetros personalizados.

Cambios en las propiedades del clúster entre versiones de Aurora MySQL

Cuando actualice de la versión 2 a la versión 3 de Aurora MySQL, asegúrese de comprobar cualquier aplicación o script que utilice para configurar o administrar clústeres e instancias de base de datos de Aurora MySQL.

Además, cambie el código que manipula los grupos de parámetros para tener en cuenta el hecho de que los nombres de grupos de parámetros predeterminados son diferentes para los clústeres compatibles con 5.7 y 8.0. Los nombres de los grupos de parámetros predeterminados para los clústeres de las versiones 2 y 3 de Aurora MySQL son `default.aurora-mysql5.7` y `default.aurora-mysql8.0`, respectivamente.

Por ejemplo, es posible que tenga código como el siguiente que se aplique al clúster antes de una actualización.

```
# Check the default parameter values for MySQL 5.7-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql5.7 --
region us-east-1
```

Después de actualizar la versión principal del clúster, modifique ese código de la siguiente manera.

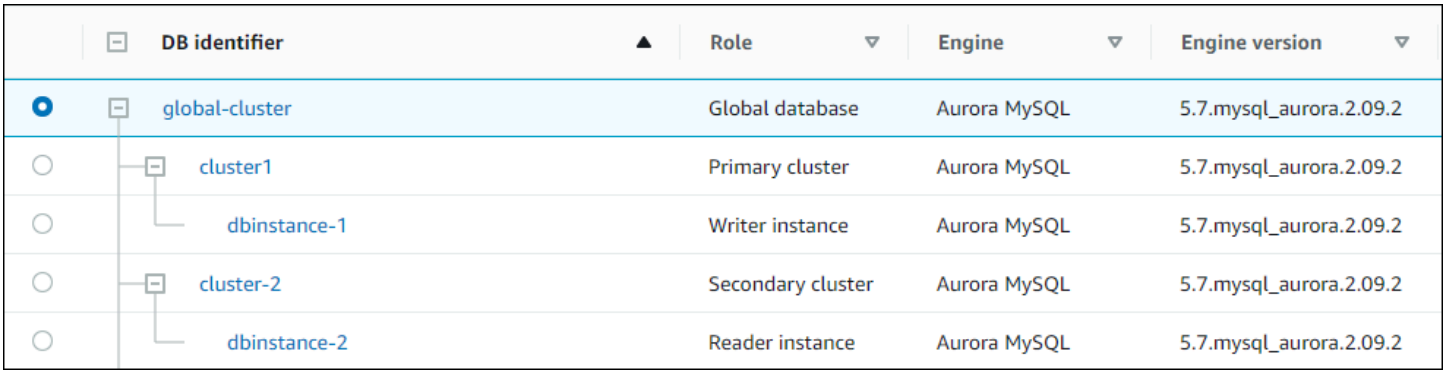
```
# Check the default parameter values for MySQL 8.0-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql8.0 --
region us-east-1
```

Actualizaciones mayores en el lugar para bases de datos globales

Para una base de datos global de Aurora, actualice el clúster de la base de datos global. Aurora actualiza automáticamente todos los clústeres al mismo tiempo y se asegura de que todos ejecuten la misma versión del motor. Este requisito se debe a que cualquier cambio en las tablas del sistema, formatos de archivo de datos, etc., se replican automáticamente en todos los clústeres secundarios.

Siga las instrucciones en Cómo funciona la actualización de la versión principal en el lugar Aurora MySQL. Cuando especifique qué actualizar, asegúrese de elegir el clúster de base de datos global en lugar de uno de los clústeres que contiene.

Si utiliza la Consola de administración de AWS, elija el elemento con el rol Global database (Base de datos global).



Si utiliza la AWS CLI o la API de RDS, inicie el proceso de actualización llamando al comando modify-global-cluster o la operación ModifyGlobalCluster. Se usa uno de estos en lugar de `modify-db-cluster` o `ModifyDBCluster`.

> **Note**
>
> No puede especificar un grupo de parámetros personalizado para el clúster de base de datos global mientras realiza una actualización importante de la versión de esa base de datos global de Aurora. Cree grupos de parámetros personalizados en cada región del clúster global. A continuación, aplíquelos manualmente a los clústeres regionales después de la actualización.

Para actualizar la versión principal de un clúster de base de datos global de Aurora MySQL, utilice el comando modify-global-cluster de la AWS CLI con los siguientes parámetros requeridos:

- `--global-cluster-identifier`
- `--engine aurora-mysql`
- `--engine-version`
- `--allow-major-version-upgrade`

En el ejemplo siguiente se actualiza el clúster de base de datos global a la versión 2.10.2 de Aurora MySQL.

Example

Para Linux, macOS o Unix:

```
aws rds modify-global-cluster \
          --global-cluster-identifier global_cluster_identifier \
          --engine aurora-mysql \
          --engine-version 5.7.mysql_aurora.2.10.2 \
          --allow-major-version-upgrade
```

En Windows:

```
aws rds modify-global-cluster ^
          --global-cluster-identifier global_cluster_identifier ^
          --engine aurora-mysql ^
          --engine-version 5.7.mysql_aurora.2.10.2 ^
          --allow-major-version-upgrade
```

Consideraciones sobre el Backtrack

Si el clúster que actualizó tenía habilitada la característica Backtrack, no podrá realizar un retroceso del clúster actualizado a una hora anterior a la actualización.

Tutorial de actualización de Aurora MySQL en el lugar

En los siguientes ejemplos de Linux se muestra cómo se pueden realizar los pasos generales del procedimiento de actualización en el lugar utilizando AWS CLI.

Este primer ejemplo crea un clúster de base de datos Aurora que ejecuta una versión 2.x de Aurora MySQL. El clúster incluye una instancia de base de datos de escritura y una instancia de base de datos de lector. El comando `wait db-instance-available` se detiene hasta que la instancia de base de datos del escritor esté disponible. Ese es el momento en que el clúster está listo para ser actualizado.

```
aws rds create-db-cluster --db-cluster-identifier mynewdbcluster --engine aurora-mysql
 \
  --db-cluster-version 5.7.mysql_aurora.2.11.2
...
aws rds create-db-instance --db-instance-identifier mynewdbcluster-instance1 \
```

```
  --db-cluster-identifier mynewdbcluster --db-instance-class db.t4g.medium --engine
 aurora-mysql
...
aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

Las versiones de Aurora MySQL 3.x en las que puede actualizar el clúster dependen de la versión 2.x en la que se está ejecutando actualmente el clúster y en la Región de AWS donde se encuentra el clúster. El primer comando, con `--output text`, solo muestra la versión de destino disponible. El segundo comando muestra la salida JSON completa de la respuesta. En esa respuesta, puede ver detalles como el valor `aurora-mysql` que utiliza para el parámetro `engine`. También puede ver el hecho de que la actualización a 3.04.0 es una actualización de la versión principal.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.11.2 \
  --query '*[].[ValidUpgradeTarget]'
...
{
    "Engine": "aurora-mysql",
    "EngineVersion": "8.0.mysql_aurora.3.04.0",
    "Description": "Aurora MySQL 3.04.0 (compatible with MySQL 8.0.28)",
    "AutoUpgrade": false,
    "IsMajorVersionUpgrade": true,
    "SupportedEngineModes": [
        "provisioned"
    ],
    "SupportsParallelQuery": true,
    "SupportsGlobalDatabases": true,
    "SupportsBabelfish": false,
    "SupportsIntegrations": false
},
...
```

En este ejemplo, se muestra que si ingresa un número de versión de destino que no es un destino de actualización válido para el clúster, Aurora no realiza la actualización. Aurora tampoco realiza una actualización de versión principal, a menos que incluya el parámetro `--allow-major-version-upgrade`. De esta manera, no puede realizar accidentalmente una actualización que tenga el potencial de requerir pruebas exhaustivas y cambios en el código de su aplicación.

```
aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 5.7.mysql_aurora.2.09.2 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: Cannot find upgrade target from 5.7.mysql_aurora.2.11.2 with requested
 version 5.7.mysql_aurora.2.09.2.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --region us-east-1 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: The AllowMajorVersionUpgrade flag must be present when upgrading to a new
 major version.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --apply-immediately --allow-major-version-
upgrade
{
  "DBClusterIdentifier": "mynewdbcluster",
  "Status": "available",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.11.2"
}
```

El estado del clúster y las instancias de base de datos asociadas tardan unos minutos en cambiar a `upgrading`. Los números de versión del clúster y de las instancias de base de datos solo cambian cuando finaliza la actualización. De nuevo, puede utilizar el comando `wait db-instance-available` para que la instancia de base de datos de escritor espere hasta que finalice la actualización antes de continuar.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[Status,EngineVersion]' --output text
upgrading 5.7.mysql_aurora.2.11.2

aws rds describe-db-instances --db-instance-identifier mynewdbcluster-instance1 \
  --query '*[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceStatus:DBInstanceStatus} | [0]'
{
    "DBInstanceIdentifier": "mynewdbcluster-instance1",
    "DBInstanceStatus": "upgrading"
}

aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

En este punto, el número de versión del clúster coincide con el especificado para la actualización.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[EngineVersion]' --output text

8.0.mysql_aurora.3.04.0
```

En el ejemplo anterior se realizó una actualización inmediata especificando el parámetro `--apply-immediately`. Para permitir que la actualización se realice en un momento conveniente cuando no se espera que el clúster esté ocupado, puede especificar el parámetro `--no-apply-immediately`. Al hacerlo, la actualización se iniciará durante la siguiente ventana de mantenimiento del clúster. La ventana de mantenimiento define el período durante el cual se pueden iniciar las operaciones de mantenimiento. Es posible que una operación de larga duración no finalice durante el período de mantenimiento. Por lo tanto, no necesita definir una ventana de mantenimiento más grande, incluso si espera que la actualización pueda tardar mucho tiempo.

En el ejemplo siguiente, se actualiza un clúster que inicialmente ejecuta la versión 2.11.2 de Aurora MySQL. En la salida de `describe-db-engine-versions`, los valores `False` y `True` representan la propiedad `IsMajorVersionUpgrade`. Desde la versión 2.11.2, puede actualizar a otras versiones 2.*. Esas actualizaciones no se consideran actualizaciones de versión principales, por lo que no requieren una actualización in situ. La actualización local solo está disponible para las actualizaciones a las versiones 3.* que se muestran en la lista.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.10.2 \
  --query '*[].[ValidUpgradeTarget]|[0][0]|[*].[EngineVersion,IsMajorVersionUpgrade]'
 --output text

5.7.mysql_aurora.2.11.3 False
5.7.mysql_aurora.2.11.4 False
5.7.mysql_aurora.2.11.5 False
5.7.mysql_aurora.2.11.6 False
5.7.mysql_aurora.2.12.0 False
5.7.mysql_aurora.2.12.1 False
5.7.mysql_aurora.2.12.2 False
5.7.mysql_aurora.2.12.3 False
8.0.mysql_aurora.3.04.0 True
```

```
8.0.mysql_aurora.3.04.1 True
8.0.mysql_aurora.3.04.2 True
8.0.mysql_aurora.3.04.3 True
8.0.mysql_aurora.3.05.2 True
8.0.mysql_aurora.3.06.0 True
8.0.mysql_aurora.3.06.1 True
8.0.mysql_aurora.3.07.1 True

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --no-apply-immediately --allow-major-
version-upgrade
...
```

Cuando se crea un clúster sin una ventana de mantenimiento especificada, Aurora selecciona un día aleatorio de la semana. En este caso, el comando `modify-db-cluster` se envía un lunes. Por lo tanto, cambiamos la ventana de mantenimiento para que sea el martes por la mañana. Todas las horas se representan en la zona horaria UTC. El periodo `tue:10:00-tue:10:30` se corresponde a las 2:00-2:30 h, hora del Pacífico. El cambio en la ventana de mantenimiento surte efecto inmediatamente.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "sat:08:20-sat:08:50"
    ]
]

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster --preferred-
maintenance-window tue:10:00-tue:10:30"
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "tue:10:00-tue:10:30"
    ]
]
```

En el siguiente ejemplo se muestra cómo obtener un informe de los eventos generados por la actualización. El argumento `--duration` representa el número de minutos para recuperar la

información del evento. Este argumento es necesario porque, de forma predeterminada, `describe-events` solo devuelve eventos de la última hora.

```
aws rds describe-events --source-type db-cluster --source-identifier mynewdbcluster --
duration 20160
{
    "Events": [
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "DB cluster created",
            "EventCategories": [
                "creation"
            ],
            "Date": "2022-11-17T01:24:11.093000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing online pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:08.450000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing offline pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:59.519000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-
mynewdbcluster-5-7-mysql-aurora-2-10-2-to-8-0-mysql-aurora-3-02-0-2022-11-18-22-55].",
            "EventCategories": [
```

```
                "maintenance"
            ],
            "Date": "2022-11-18T23:00:22.318000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Cloning volume.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:01:45.428000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:02:25.141000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:23.036000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Upgrading database objects.",
            "EventCategories": [
                "maintenance"
            ],
```

```
            "Date": "2022-11-18T23:06:48.208000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster major version has been upgraded",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:10:28.999000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        }
    ]
}
```

Determinación del motivo de los errores en una actualización de la versión principal de Aurora MySQL

En el tutorial, la actualización de Aurora MySQL versión 2 a la versión 3 se realizó correctamente. Pero si la actualización hubiera fallado, querría saber por qué.

Puede empezar por utilizar el comando `describe-events` de la AWS CLI para ver los eventos del clúster de base de datos. En este ejemplo, se muestran los eventos de `mydbcluster` de las últimas 10 horas.

```
aws rds describe-events \
    --source-type db-cluster \
    --source-identifier mydbcluster \
    --duration 600
```

En este caso, se produjo un error en la comprobación previa de la actualización.

```
{
    "Events": [
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
```

```
            "Date": "2024-04-11T13:23:22.846000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        },
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster is in a state that cannot be upgraded: Upgrade
 prechecks failed. For more details, see the
             upgrade-prechecks.log file. For more information on troubleshooting the
 cause of the upgrade failure, see
             https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
AuroraMySQL.Upgrading.Troubleshooting.html",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:24.373000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        }
    ]
}
```

Para diagnosticar la causa exacta del problema, examine los registros de la base de datos de la instancia de base de datos de escritor. Cuando se produce un fallo en una actualización a Aurora MySQL versión 3, la instancia de escritor contiene un archivo de registro con el nombre `upgrade-prechecks.log`. En este ejemplo se muestra cómo detectar la presencia de ese registro y, luego, descargarlo en un archivo local para su análisis.

```
aws rds describe-db-log-files --db-instance-identifier mydbcluster-instance \
    --query '*[].[LogFileName]' --output text

error/mysql-error-running.log
error/mysql-error-running.log.2024-04-11.20
error/mysql-error-running.log.2024-04-11.21
error/mysql-error.log
external/mysql-external.log
upgrade-prechecks.log

aws rds download-db-log-file-portion --db-instance-identifier mydbcluster-instance \
    --log-file-name upgrade-prechecks.log \
    --starting-token 0 \
    --output text >upgrade_prechecks.log
```

El archivo `upgrade-prechecks.log` está en formato JSON. Lo descargamos con la opción `--output text` para evitar codificar la salida JSON dentro de otro contenedor JSON. Para las actualizaciones a la versión 3 de Aurora MySQL, este registro siempre incluye ciertos mensajes informativos y de advertencia. Solo incluye mensajes de error si no se puede actualizar. Si se actualiza correctamente, el archivo de registro ya no se produce.

Para resumir todos los errores y mostrar los campos de objeto y descripción asociados, puede ejecutar el comando `grep -A 2 '"level": "Error"'` en el contenido del archivo `upgrade-prechecks.log`. Al hacerlo, se muestra cada línea de error y las dos líneas posteriores. Estas contienen el nombre del objeto de base de datos correspondiente e instrucciones sobre cómo corregir el problema.

```
$ cat upgrade-prechecks.log | grep -A 2 '"level": "Error"'

"level": "Error",
"dbObject": "problematic_upgrade.dangling_fulltext_index",
"description": "Table `problematic_upgrade.dangling_fulltext_index` contains dangling
 FULLTEXT index. Kindly recreate the table before upgrade."
```

En este ejemplo, puede ejecutar el siguiente comando de SQL en la tabla infractora para intentar solucionar el problema, o bien puede volver a crear la tabla sin el índice pendiente.

```
OPTIMIZE TABLE problematic_upgrade.dangling_fulltext_index;
```

A continuación, vuelva a intentar la actualización.

Solución de problemas para la actualización Aurora MySQL en el lugar

Utilice las siguientes sugerencias para solucionar problemas con las actualizaciones in situ de Aurora MySQL. Estas sugerencias no se aplican a los clústeres de base de datos de Aurora Serverless.

Motivo por el que la actualización en el lugar se cancela o se ralentiza	Efecto	Solución para permitir que la actualización en el lugar se complete dentro de la ventana de mantenimiento
La réplica entre regiones de Aurora asociadas aún no cuenta con revisión	Aurora cancela la actualización.	Actualice la réplica entre regiones de Aurora e inténtelo de nuevo.

Motivo por el que la actualización en el lugar se cancela o se ralentiza	Efecto	Solución para permitir que la actualización en el lugar se complete dentro de la ventana de mantenimiento
El clúster tiene transacciones XA en el estado preparado	Aurora cancela la actualización.	Confirme o revierta todas las transacciones XA preparadas.
El clúster está procesando cualquier declaración de lenguaje de definición de datos (DDL)	Aurora cancela la actualización.	Considere la posibilidad de esperar y realizar la actualización una vez finalizadas todas las instrucciones DDL.
El clúster tiene cambios no confirmados para muchas filas	Esto puede llevar mucho tiempo.	El proceso de actualización revierte los cambios no confirmados. El indicador para esta condición es el valor de `TRX_ROWS_MODIFIED` en la `INFORMATION_SCHEMA.INNODB_TRX` tabla. Considere la posibilidad de realizar la actualización solo después de que todas las transacciones grandes se hayan confirmado o deshecho.

Motivo por el que la actualización en el lugar se cancela o se ralentiza	Efecto	Solución para permitir que la actualización en el lugar se complete dentro de la ventana de mantenimiento
El clúster tiene un gran número de registros de deshacer	Esto puede llevar mucho tiempo.	Incluso si las transacciones no confirmadas no afectan a un gran número de filas, pueden implicar un gran volumen de datos. Por ejemplo, puede que esté insertando BLOBs grandes. Aurora no detecta ni genera automáticamente un evento para este tipo de actividad de transacción. El indicador de esta condición es la longitud de la lista de historial (HLL). El proceso de actualización revierte los cambios no confirmados. Puede comprobar el HLL en la salida del comando SQL `SHOW ENGINE INNODB STATUS` o directamente mediante la siguiente consulta SQL: `SELECT count FROM information_schema .innodb_metrics WHERE name = 'trx_rseg_history_len';` También puede monitorizar la métrica `RollbackSegmentHistoryListLength` en Amazon CloudWatch. Considere la posibilidad de realizar la actualización solo después de que la HLL sea menor.

Motivo por el que la actualización en el lugar se cancela o se ralentiza	Efecto	Solución para permitir que la actualización en el lugar se complete dentro de la ventana de mantenimiento
El clúster está en el proceso de confirmar una transacción de registro binario grande	Esto puede llevar mucho tiempo.	El proceso de actualización espera hasta que se apliquen los cambios en el registro binario. Más transacciones o instrucciones DDL podrían comenzar durante este período, lo que ralentiza aún más el proceso de actualización. Programe el proceso de actualización cuando el clúster no esté ocupado con la generación de cambios de reproducción de registros binarios. Aurora no detecta ni genera automáticamente un evento para esta condición.

Motivo por el que la actualización en el lugar se cancela o se ralentiza	Efecto	Solución para permitir que la actualización en el lugar se complete dentro de la ventana de mantenimiento
Inconsistencias en el esquema causadas por la eliminación o la corrupción de un archivo	Aurora cancela la actualización.	Cambie el motor de almacenamiento predeterminado para las tablas temporales de MyISAM a InnoDB. Siga estos pasos: 1. Modifique el parámetro de base de datos `default_tmp_storage_engine` por `InnoDB`. 2. Reinicie el clúster de base de datos. 3. Tras reiniciar, confirme que el parámetro de base de datos `default_tmp_storage_engine` se haya establecido en `InnoDB`. Utilice el siguiente comando: `show global variables like 'default_tmp_storage_engine';` 4. Asegúrese de no crear ninguna tabla temporal que utilice el motor de almacenamiento MyISAM. Le recomendamos que ponga en pausa la carga de trabajo de la base de datos y no cree ninguna conexión nueva a la base de datos, ya que la actualizará pronto. 5. Vuelva a intentar la actualización in situ.

Motivo por el que la actualización en el lugar se cancela o se ralentiza	Efecto	Solución para permitir que la actualización en el lugar se complete dentro de la ventana de mantenimiento

Se ha eliminado el usuario maestro	Aurora cancela la actualización.	**Important**
		No elimine el usuario maestro.
		
		Sin embargo, si por alguna razón elimina el usuario maestro, restáurelo mediante los siguientes comandos SQL:
		
		`CREATE USER 'master_username '@'% IDENTIFIED BY ' master_user_password ' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;`
		
		`GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, LOAD FROM S3, SELECT INTO S3, INVOKE LAMBDA, INVOKE SAGEMAKER, INVOKE COMPREHEND ON *.* TO 'master_username '@'% WITH GRANT OPTION;`

Para obtener más información sobre la solución de problemas que provocan el error de las comprobaciones previas a la actualización, consulte los siguientes blogs:

- Lista de verificación de actualización de Amazon Aurora MySQL versión 2 (compatible con MySQL 5.7) a la versión 3 (compatible con MySQL 8.0), parte 1

- Lista de verificación de actualización de Amazon Aurora MySQL versión 2 (compatible con MySQL 5.7) a la versión 3 (compatible con MySQL 8.0), parte 2

Puede utilizar los siguientes pasos para realizar sus propias comprobaciones de algunas de las condiciones de la tabla anterior. De esta forma, puede programar la actualización en un momento en que sepa que la base de datos se encuentra en un estado en el que la actualización puede completarse correctamente y rápidamente.

- Puede comprobar si hay transacciones XA abiertas ejecutando la instrucción `XA RECOVER`. A continuación, puede confirmar o revertir las transacciones XA antes de iniciar la actualización.
- Puede comprobar si hay instrucciones DDL ejecutando una instrucción `SHOW PROCESSLIST` y buscando las instrucciones `CREATE`, `DROP`, `ALTER`, `RENAME` y `TRUNCATE` en la salida. Permita que todas las instrucciones DDL finalicen antes de iniciar la actualización.
- Puede comprobar el número total de filas no confirmadas consultando la tabla `INFORMATION_SCHEMA.INNODB_TRX`. La tabla contiene una fila para cada transacción. La columna `TRX_ROWS_MODIFIED` contiene el número de filas modificadas o insertadas por la transacción.
- Puede verificar la longitud de la lista de historial de InnoDB ejecutando la instrucción `SHOW ENGINE INNODB STATUS SQL` y buscando el `History list length` en la salida. También puede comprobar el valor directamente ejecutando la siguiente consulta:

  ```
  SELECT count FROM information_schema.innodb_metrics WHERE name =
   'trx_rseg_history_len';
  ```

  La longitud de la lista de historial corresponde a la cantidad de información de deshacer almacenada por la base de datos para implementar el control de concurrencia multiversión (MVCC).

Limpieza posterior a la actualización para Aurora MySQL versión 3

Una vez que haya terminado de actualizar cualquier clúster de Aurora MySQL versión 2 a Aurora MySQL versión 3, puede realizar estas otras acciones de limpieza:

- Cree nuevas versiones compatibles con MySQL 8.0 de cualquier grupo de parámetros personalizados. Aplique los valores de parámetros personalizados necesarios a los nuevos grupos de parámetros.

- Actualice las alarmas de CloudWatch, los scripts de configuración, etc. para utilizar los nuevos nombres para cualquier métrica cuyos nombres se hayan visto afectados por cambios de idioma inclusivos. Para ver una lista de estas métricas, consulte Cambios de idioma inclusivos para Aurora MySQL versión 3.
- Actualice cualquier plantilla CloudFormation para utilizar los nuevos nombres para cualquier parámetro de configuración cuyos nombres se hayan visto afectados por cambios de idioma inclusivos. Para obtener una lista completa de estos parámetros, consulte Cambios de idioma inclusivos para Aurora MySQL versión 3.

Índices espaciales

Después de actualizar a Aurora MySQL versión 3, verifique si necesita eliminar o volver a crear objetos e índices relacionados con los índices espaciales. Antes de MySQL 8.0, Aurora podía optimizar las consultas espaciales utilizando índices que no contenían un identificador de recursos espaciales (SRID). Aurora MySQL versión 3 solo utiliza índices espaciales que contienen SRID. Durante una actualización, Aurora elimina automáticamente cualquier índice espacial sin SRID e imprime mensajes de advertencia en el registro de la base de datos. Si observa estos mensajes de advertencia, cree nuevos índices espaciales con SRID después de la actualización. Para obtener más información sobre los cambios en las funciones espaciales y los tipos de datos de MySQL 8.0, consulte Cambios en MySQL 8.0 en el Manual de referencia de MySQL.

Actualizaciones y correcciones de motor de base de datos de Amazon Aurora MySQL

Encontrará la siguiente información en las Notas de la versión de la edición compatible con Amazon Aurora MySQL:

- Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 3
- Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 2
- Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 1 (obsoleta)
- Errores de MySQL corregidos en las actualizaciones del motor de base de datos de Aurora MySQL
- Vulnerabilidades de seguridad corregidas en Amazon Aurora MySQL

Uso de Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL es un motor de bases de datos relacionales, completamente administrado, compatible con PostgreSQL y conforme a ACID, que combina la velocidad, la fiabilidad y la manejabilidad de Amazon Aurora con la sencillez y la rentabilidad de las bases de datos de código abierto. Aurora PostgreSQL es un reemplazo instántaneo para PostgreSQL que simplifica y hace más rentable configurar, usar y escalar las implementaciones de PostgreSQL nuevas y existentes, lo que le permitirá centrarse en su negocio y sus aplicaciones. Para obtener más información sobre Aurora en general, consulte ¿Qué es Amazon Aurora?.

Además de los beneficios de Aurora, Aurora PostgreSQL ofrece una ruta de migración conveniente de Amazon RDS a Aurora, con herramientas de migración con tan solo pulsar un botón que convierten sus aplicaciones de RDS for PostgreSQL en Aurora PostgreSQL. Las tareas de base de datos rutinarias como el aprovisionamiento, la aplicación de parches, las copias de seguridad, la recuperación, la detección de errores y la reparación también son fáciles de gestionar con Aurora PostgreSQL.

Aurora PostgreSQL puede trabajar con muchos estándares del sector. Por ejemplo, puede utilizar las bases de datos de Aurora PostgreSQL para crear aplicaciones conformes a la HIPAA y para almacenar información relacionada con la sanidad, incluida información sanitaria protegida (PHI) bajo un contrato de asociación empresarial (BAA) con AWS.

Aurora PostgreSQL es apto para FedRAMP HIGH. Para obtener más detalles sobre AWS y los esfuerzos de conformidad, consulte Servicios de AWS en el alcance por programa de conformidad.

> **Important**
>
> Si detecta algún problema con el clúster de base de datos de Aurora PostgreSQL, es posible que el agente de asistencia de AWS necesite más información sobre el estado de las bases de datos. El objetivo es asegurarnos de que la asistencia de AWS recibe la mayor cantidad posible de información requerida en el menor tiempo posible.
> Puede utilizar PG Collector para recopilar información valiosa sobre la base de datos en un archivo HTML consolidado. Para obtener más información sobre PG Collector, cómo ejecutarlo y cómo descargar el informe HTML, consulte PG Collector.
> Una vez finalizado correctamente, y a menos que se indique lo contrario, el script devuelve el resultado en un formato HTML legible. El script está diseñado para excluir cualquier

dato o detalle de seguridad del HTML que pueda comprometer el negocio. Además, no realiza modificaciones en la base de datos ni en el entorno. Sin embargo, si encuentra alguna información en el HTML que no le resulte cómodo compartir, no dude en eliminar la información problemática antes de cargar el HTML. Cuando el HTML sea aceptable, cárguelo utilizando la sección de archivos adjuntos en los detalles del caso de su caso de asistencia.

Temas

- Trabajo con el entorno de vista previa de base de datos
- Seguridad con Amazon Aurora PostgreSQL
- Actualización de aplicaciones para la conexión a los clústeres de base de datos de PostgreSQL de Aurora con los nuevos certificados SSL/TLS
- Uso de la autenticación Kerberos con Aurora PostgreSQL
- Migración de datos a Amazon Aurora con compatibilidad con PostgreSQL
- Optimización del rendimiento de las consultas en Aurora PostgreSQL
- Trabajo con tablas no registradas en Aurora PostgreSQL
- Uso de autovacuum de PostgreSQL en Amazon Aurora PostgreSQL
- Administración de la contención de TOAST OID en Amazon Aurora PostgreSQL
- Uso de Babelfish para Aurora PostgreSQL
- Rendimiento y escalado para Amazon Aurora PostgreSQL
- Ajuste con eventos de espera de Aurora PostgreSQL
- Ajuste de Aurora PostgreSQL con información proactiva de Amazon DevOps Guru
- Prácticas recomendadas con Amazon Aurora PostgreSQL
- Replicación con Amazon Aurora PostgreSQL
- Reenvío de escritura local en Aurora PostgreSQL
- Uso de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock
- Integración de Amazon Aurora PostgreSQL con otros servicios de AWS
- Monitorización de planes de ejecución de consultas y máximo de memoria para Aurora PostgreSQL
- Administración de planes de ejecución de consultas para Aurora PostgreSQL
- Uso de extensiones y contenedores de datos externos

- Uso de Extensiones de lenguaje de confianza para PostgreSQL
- Referencia de Amazon Aurora PostgreSQL
- Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL

Trabajo con el entorno de vista previa de base de datos

La comunidad de PostgreSQL lanza una nueva versión principal de PostgreSQL cada año. Del mismo modo, Amazon Aurora ofrece algunas versiones principales de PostgreSQL como versiones preliminares. Esto le permite crear clústeres de base de datos con la versión de vista previa y probar sus características en el entorno de vista previa de la base de datos.

Los clústeres de base de datos de Aurora en el entorno de vista previa de bases de datos son funcionalmente similares a otros clústeres de base de datos de Aurora PostgreSQL. Sin embargo, no puede usar una versión de vista previa para la producción.

Tenga en cuenta las siguientes limitaciones importantes:

- Todas las instancias y clústeres de base de datos se eliminan 60 días después de crearlas, junto con las copias de seguridad e instantáneas.
- Solo puede crear una instancia de base de datos en una nube privada virtual (VPC) en función del servicio de Amazon VPC.
- No puede copiar una instantánea de una instancia de base de datos en un entorno de producción.

Las siguientes opciones son compatibles con la vista previa:

- Puede crear instancias de base de datos únicamente con los tipos de instancia r5, r6g, r6i, r7g, r7i, r8g, x2g, t3 y t4g. Para obtener más información sobre las clases de instancias, consulte Clases de instancia de base de datos de Amazon Aurora.
- Puede utilizar implementaciones single-AZ y multi-AZ.
- Puede utilizar las funciones estándar de volcado y carga de PostgreSQL para exportar bases de datos desde o importar bases de datos hacia el entorno de la vista previa de base de datos.

Tipos de clases de instancia de base de datos compatibles

Amazon Aurora PostgreSQL admite las siguientes clases de instancia base de datos en la región de vista previa:

Clases optimizadas para memoria

- db.r5: clases de instancia optimizada para memoria
- db.r6g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton2 de AWS
- db.r6i: clases de instancia optimizada para memoria
- db.r7g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton3 de AWS
- db.r7i: clases de instancia optimizada para memoria
- db.r8g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton4 de AWS
- db.x2g: clases de instancia optimizada para memoria con tecnología de procesadores Graviton2 de AWS

> **Note**
>
> Para obtener más información sobre la lista de clases de instancia, consulte Tipos de clase de instancia de base de datos.

Clases ampliables

- db.t3.medium
- db.t3.large
- db.t4g.medium
- db.t4g.large

Características no admitidas en el entorno de vista previa

Las siguientes características no están disponibles en el entorno de vista previa:

- Aurora Serverless v1 y v2
- Actualizaciones de la versión principal (MVU)
- No se lanzarán nuevas versiones secundarias en la región de vista previa

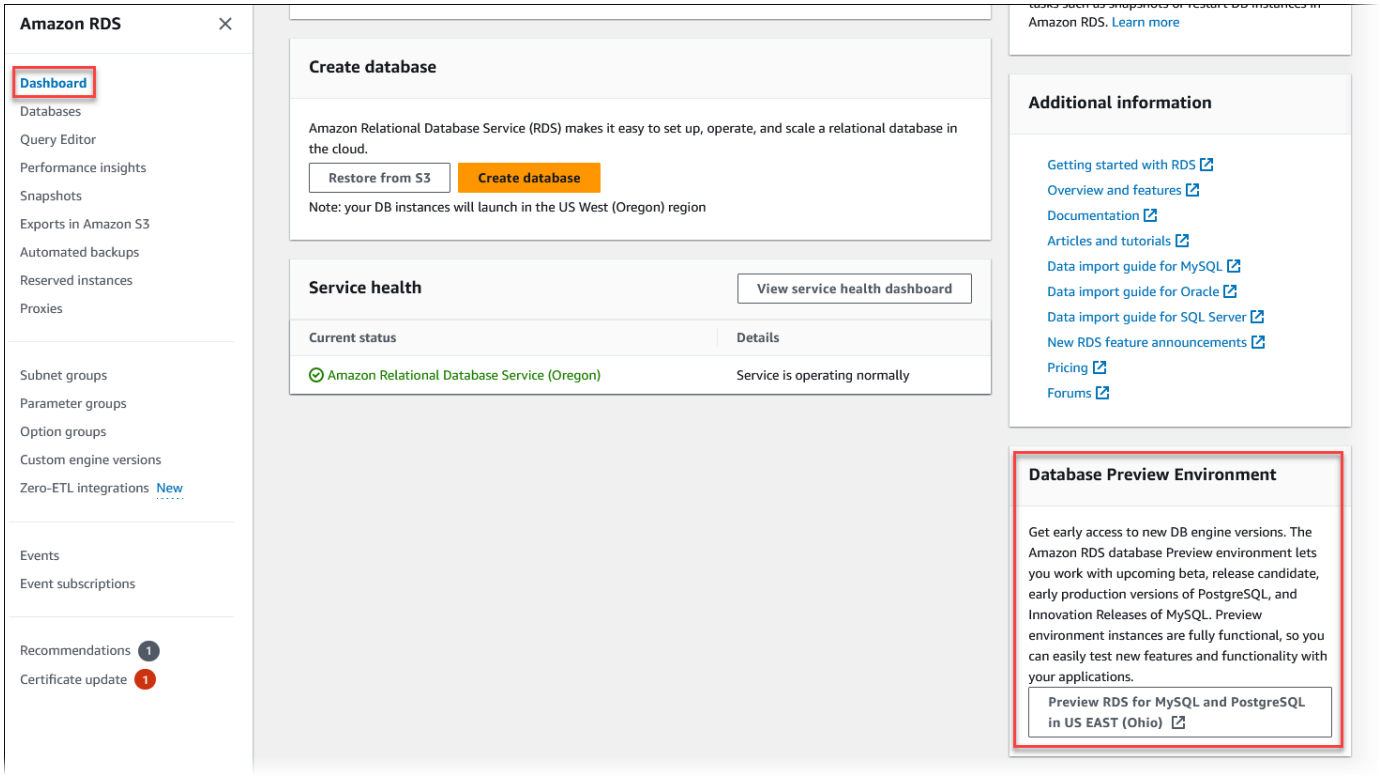
- Replicación entrante de RDS for PostgreSQL a Aurora PostgreSQL
- Implementaciones azules/verdes de Amazon RDS
- Copia de instantáneas entre regiones
- Base de datos global de Aurora
- Secuencias de actividades de la base de datos (DAS), RDS Proxy y AWS DMS
- Réplicas de lectura de escalado automático
- AWS Bedrock
- Exportación a RDS
- Información de rendimiento
- Puntos de conexión personalizados
- Copia de instantánea
- Sin ETL
- Babelfish
- Todas las extensiones

Creación de un nuevo clúster de base de datos en el entorno de vista previa

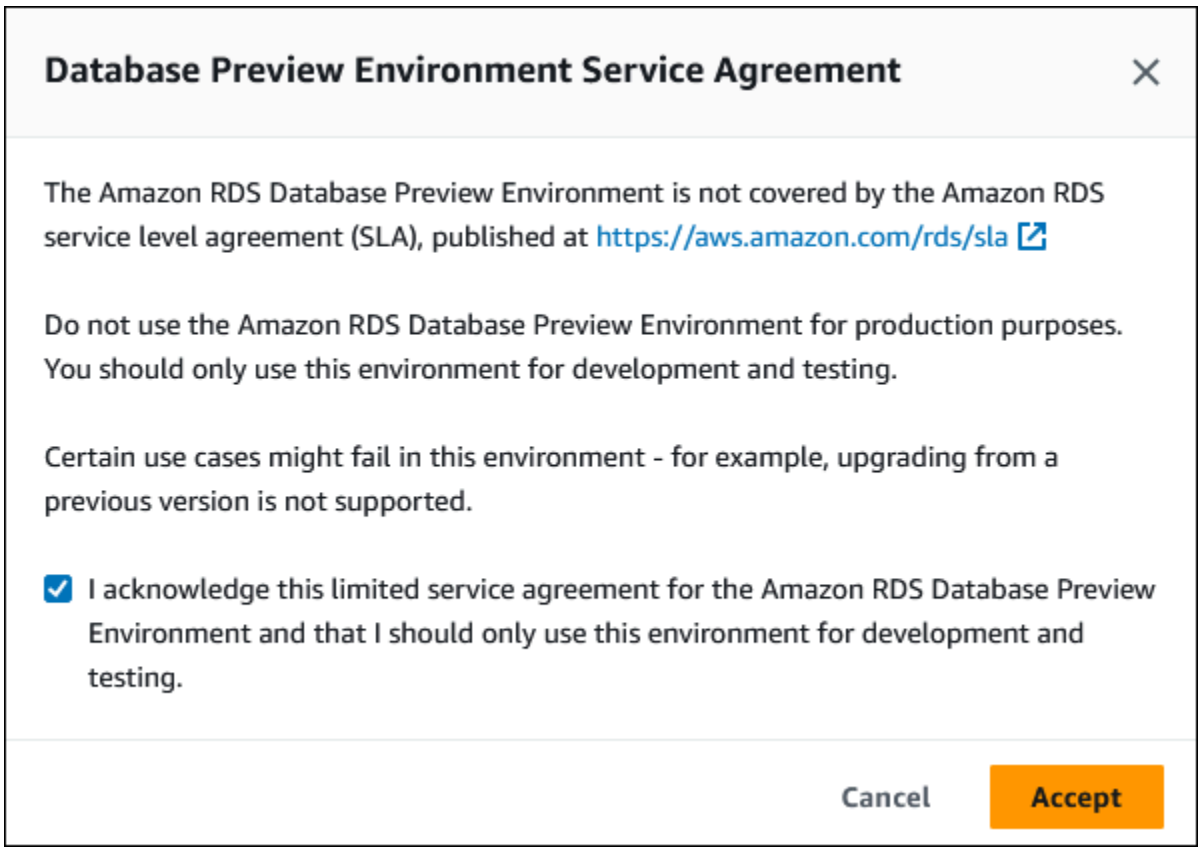
Utilice el procedimiento siguiente para crear un clúster de base de datos en el entorno de vista previa.

Para crear un clúster de base de datos en el entorno de vista previa

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Panel en el panel de navegación.
3. En la página Dashboard (Panel), busque la sección Database Preview Environment (Entorno de vista previa de base de datos), tal como se muestra en la siguiente imagen.



También puede navegar directamente a Database Preview Environment (Entorno de vista previa de base de datos). Antes de continuar, debe reconocer y aceptar las limitaciones.



4. Para crear el clúster de base de datos de Aurora PostgreSQL, siga el mismo proceso que para crear cualquier clúster de base de datos de Aurora. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

Para crear una instancia en el entorno de vista previa de bases de datos mediante la API de Aurora o la AWS CLI, utilice el siguiente punto de conexión.

```
rds-preview.us-east-2.amazonaws.com
```

Versión 17 de PostgreSQL en el entorno de vista previa de bases de datos

⚠ Esta es la documentación preliminar de la versión 17 de Aurora PostgreSQL. Está sujeta a cambios.

La versión 17.0 de PostgreSQL ya está disponible en el entorno de vista previa de bases de datos de Amazon RDS. PostgreSQL versión 17 contiene varias mejoras que se describen en la siguiente documentación de PostgreSQL:

- Lanzamiento de PostgreSQL 17

Para obtener información acerca del entorno de vista previa de base de datos, consulte Trabajo con el entorno de vista previa de base de datos. Para acceder al entorno de vista previa desde la consola, seleccione https://console.aws.amazon.com/rds-preview/.

Seguridad con Amazon Aurora PostgreSQL

Para obtener información general de la seguridad de Aurora, consulte Seguridad en Amazon Aurora. Puede administrar la seguridad de Amazon Aurora PostgreSQL en varios niveles diferentes:

- Utilice AWS Identity and Access Management (IAM) para controlar quién puede realizar acciones de administración de Amazon RDS en clústeres de base de datos e instancias de base de datos de Aurora PostgreSQL. IAM gestiona la autenticación de la identidad del usuario antes de que el usuario pueda acceder al servicio. También se encarga de la autorización, es decir, si el usuario puede hacer lo que está intentando hacer. La autenticación de bases de datos de IAM es un método de autenticación adicional que puede elegir al crear el clúster de base de datos Aurora PostgreSQL. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.

  Si utiliza IAM con el clúster de bases de datos de Aurora PostgreSQL, inicie sesión en la Consola de administración de AWS con sus credenciales de IAM primero, antes de abrir la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
- Asegúrese de crear clústeres de base de datos de Aurora en una nube privada virtual (VPC) basada en el servicio Amazon VPC. Utilice un grupo de seguridad de VPC para controlar qué dispositivos e instancias de Amazon EC2 pueden abrir conexiones con el punto de conexión y el puerto de la instancia de base de datos para los clústeres de base de datos de Aurora en una VPC. Puede realizar estas conexiones de puntos de conexión y puertos mediante el uso de capa de sockets seguros (SSL). Además, las reglas del firewall de su compañía pueden controlar si los dispositivos que se ejecutan en ella pueden abrir conexiones a una instancia de base de datos. Para obtener más información acerca de las VPC, consulte VPC de Amazon y Amazon Aurora.

La tenencia de VPC admitida depende de la clase de instancia de base de datos que utilicen los clústeres de base de datos de Aurora PostgreSQL. En el caso de la tenencia de VPC `default`, el clúster de base de datos se ejecuta en hardware compartido. En el caso de la tenencia de una VPC `dedicated`, el clúster de base de datos se ejecuta en una instancia de hardware dedicada. Las clases de instancia de base de datos de rendimiento ampliable solo admiten el arrendamiento de VPC predeterminado. Las clases de instancia de base de datos de rendimiento ampliable incluyen las clases de instancia de base de datos db.t3 y db.t4g. Todas las demás clases de instancia de base de datos de Aurora PostgreSQL admiten la tenencia de VPC predeterminada y dedicada.

Para obtener más información sobre las clases de instancias, consulte Clases de instancia de base de datos de Amazon Aurora. Para obtener más información acerca de la tenencia de VPC `default` y `dedicated`, consulte Instancias dedicadas en la Guía del usuario de Amazon Elastic Compute Cloud.

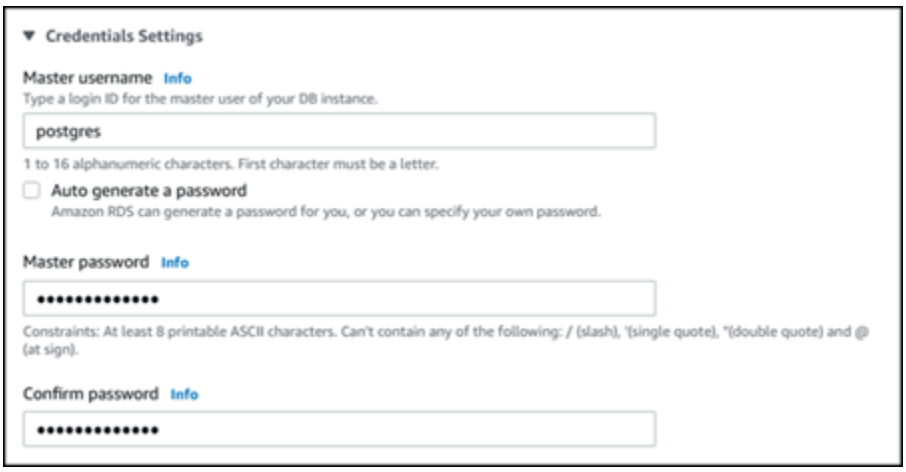
- Para conceder permisos a las bases de datos PostgreSQL que se ejecutan en el clúster de bases de datos Amazon Aurora, puede seguir el mismo procedimiento general que con las instancias independientes de PostgreSQL. Los comandos como `CREATE ROLE`, `ALTER ROLE`, `GRANT` y `REVOKE` funcionan de la misma forma que en las bases de datos en las instalaciones, al igual que la modificación directa de las tablas de los esquemas de las bases de datos.

  PostgreSQL administra los privilegios utilizando roles. El rol `rds_superuser` es el rol más privilegiado de un clúster de base de datos de Aurora PostgreSQL. Este rol se crea automáticamente y se otorga al usuario que crea el clúster de base de datos (la cuenta de usuario principal, `postgres` de forma predeterminada). Para obtener más información, consulte Descripción de los roles y permisos de PostgreSQL.

Todas las versiones de Aurora PostgreSQL, incluidas las versiones 10, 11, 12, 13, 14 y las versiones posteriores, admiten el mecanismo de autenticación mediante desafío-respuesta discontinuo (SCRAM) para las contraseñas como alternativa al resumen de mensajes (MD5). Le recomendamos que utilice SCRAM porque es más seguro que MD5. Para obtener más información, incluida la forma de migrar las contraseñas de los usuarios de base de datos de MD5 a SCRAM, consulte Uso de SCRAM para el cifrado de contraseñas de PostgreSQL.

Descripción de los roles y permisos de PostgreSQL

Al crear un clúster de base de datos Aurora PostgreSQL utilizando la Consola de administración de AWS, se crea una cuenta de administrador al mismo tiempo. De forma predeterminada su nombre es `postgres`, tal y como se muestra en la siguiente captura de pantalla:



Puede elegir otro nombre en lugar de aceptar el valor predeterminado (`postgres`). Si lo hace, el nombre que elija debe empezar por una letra y tener entre 1 y 16 caracteres alfanuméricos. Por simplicidad, nos referimos a esta cuenta de usuario principal por su valor predeterminado (`postgres`) en toda esta guía.

Si utiliza `create-db-cluster` de la AWS CLI en lugar de la Consola de administración de AWS, crea el nombre de usuario pasándolo con el parámetro `master-username`. Para obtener más información, consulte Paso 2: crear un clúster de base de datos de Aurora PostgreSQL.

Ya sea que utilice la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS y si usa el nombre predeterminado `postgres` o elige otro nombre, esta primera cuenta de usuario de la base de datos es miembro del grupo `rds_superuser` y tiene `rds_superuser` privilegios.

Temas

- Descripción del rol rds_superuser
- Control del acceso de los usuarios a la base de datos de PostgreSQL
- Delegación y control de la administración de contraseñas de usuario
- Uso de SCRAM para el cifrado de contraseñas de PostgreSQL

Descripción del rol rds_superuser

En PostgreSQL, un rol puede definir un usuario, un grupo o un conjunto de permisos específicos concedidos a un grupo o usuario para varios objetos de la base de datos. Comandos de PostgreSQL para `CREATE USER` y `CREATE GROUP` se han sustituidos por los más generales, `CREATE ROLE` con propiedades específicas para distinguir a los usuarios de bases de datos. Un usuario de base de datos se puede concebir como un rol con el privilegio LOGIN.

> **Note**
>
> Los comandos `CREATE USER` y `CREATE GROUP` se pueden seguir utilizando. Para obtener más información, consulte Roles de base de datos en la documentación de PostgreSQL.

El usuario `postgres` es el usuario de base de datos más privilegiado del clúster de base de datos Aurora PostgreSQL. Tiene las características definidas mediante la siguiente instrucción `CREATE ROLE`.

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

Las propiedades `NOSUPERUSER`, `NOREPLICATION`, `INHERIT` y `VALID UNTIL 'infinity'` son las opciones predeterminadas de CREATE ROLE, a menos que se especifique lo contrario.

De forma predeterminada, `postgres` tiene privilegios otorgados al rol `rds_superuser` y permisos para crear roles y bases de datos. El rol `rds_superuser` permite al usuario `postgres` hacer lo siguiente:

- Añadir extensiones que estén disponibles para el uso con Aurora PostgreSQL. Para obtener más información, consulte Uso de extensiones y contenedores de datos externos.
- Cree roles para los usuarios y conceder privilegios a los usuarios. Para obtener más información, consulte CREATE ROLE y GRANT en la documentación de PostgreSQL.
- Creación de bases de datos Para obtener más información, consulte CREATE DATABASE en la documentación de PostgreSQL.
- Conceder privilegios de `rds_superuser` a los roles de usuario que no tengan estos privilegios y revocarlos según sea necesario. Le recomendamos que conceda este rol solo a los usuarios que realizan tareas de superusuario. En otras palabras, puede conceder este rol a los administradores de bases de datos (DBA) o a los administradores del sistema.

- Conceda (y revoque) el rol `rds_replication` a usuarios de bases de datos que no tengan el rol `rds_superuser`.
- Conceder (y revocar) el rol `rds_password` a usuarios de bases de datos que no tengan el rol `rds_superuser`.
- Obtener información de estado sobre todas las conexiones de base de datos mediante la vista `pg_stat_activity`. Cuando sea necesario, `rds_superuser` puede detener cualquier conexión mediante `pg_terminate_backend` o `pg_cancel_backend`.

En la instrucción `CREATE ROLE postgres...`, se puede ver que el rol de usuario `postgres` no permite específicamente permisos `superuser` de PostgreSQL. Aurora PostgreSQL es un servicio administrado, por lo que no puede acceder al sistema operativo host y no puede conectarse con la cuenta `superuser` de PostgreSQL. Muchas de las tareas que requieren acceso `superuser` en un PostgreSQL independiente se administra automáticamente mediante Aurora.

Para obtener más información sobre la concesión de privilegios, consulte GRANT en la documentación de PostgreSQL.

El rol `rds_superuser` es uno de varios roles predefinidos en un clúster de base de datos de Aurora PostgreSQL.

> **Note**
>
> En PostgreSQL 13 y versiones anteriores, los roles predefinidos se denominan roles predeterminados.

En la siguiente lista encontrará algunos de los otros roles predefinidos que se crean automáticamente para un nuevo clúster de base de datos de Aurora PostgreSQL. Los roles predefinidos y sus privilegios no se pueden cambiar. No se pueden eliminar, cambiar de nombre ni modificar los privilegios de estos roles predefinidos. Intentar realizar una de estas operaciones producirá un error.

- rds_password: rol que puede cambiar las contraseñas y configurar restricciones de contraseña para los usuarios de bases de datos. Al rol `rds_superuser` se le otorga este rol de forma predeterminada y puede otorgarlo a los usuarios de la base de datos. Para obtener más información, consulte Control del acceso de los usuarios a la base de datos de PostgreSQL.

- En las versiones de RDS para PostgreSQL anteriores a la 14, el rol `rds_password` puede cambiar las contraseñas y establecer restricciones de contraseña para los usuarios de la base de datos y los usuarios con el rol `rds_superuser`. A partir de la versión 14 de RDS para PostgreSQL, el rol `rds_password` puede cambiar las contraseñas y establecer restricciones de contraseña solo para los usuarios de la base de datos. Solo los usuarios con el rol `rds_superuser` pueden realizar estas acciones en otros usuarios con el rol `rds_superuser`.
- rdsadmin: rol creado para administrar muchas de las tareas de administración que el administrador con privilegios de `superuser` realizaría en una base de datos PostgreSQL independiente. Este rol lo utiliza internamente Aurora PostgreSQLpara muchas tareas de administración.

Visualización los de roles y sus privilegios

Puede ver los roles predefinidos y sus privilegios en la instancia de base de datos de RDS para PostgreSQL mediante distintos comandos en función de la versión de PostgreSQL.

Para PostgreSQL 15 y versiones anteriores

Conéctese a la instancia de base de datos de RDS para PostgreSQL y use el comando \du en psql:

```
postgres=> \du
                                                              List of roles
    Role name    |                         Attributes                          |
                  Member of
-----------------+-----------------------------------------------------------------
+-----------------------------------------------------
 postgres        | Create role, Create DB                                      +|
 {rds_superuser}
                 | Password valid until infinity                                |
 rds_ad          | Cannot login                                                 | {}
 rds_iam         | Cannot login                                                 | {}
 rds_password    | Cannot login                                                 | {}
 rds_replication | Cannot login                                                 | {}
 rds_superuser   | Cannot login                                                 |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
                 | Password valid until infinity                                |
```

Para PostgreSQL 16 y versiones posteriores

A partir de PostgreSQL 16, utilice el comando `\drg+` para ver información detallada sobre la pertenencia a roles:

```
postgres=> \drg+
                              List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

Para comprobar la pertenencia a roles independientemente de la versión de PostgreSQL, puede utilizar la siguiente consulta SQL:

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

En la salida, puede ver que `rds_superuser` no es un rol de usuario de base de datos (no puede iniciar sesión), pero tiene los privilegios de muchos otros roles. También puede ver que el usuario de la base de datos `postgres` es miembro del rol `rds_superuser`. Como se ha mencionado anteriormente, `postgres` es el valor predeterminado de Create database (Crear base de datos) de la consola de Amazon RDS. Si ha elegido otro nombre, ese nombre se muestra en la lista de roles.

> **Note**
>
> Las versiones 15.2 y 14.7 de Aurora PostgreSQL, introdujeron un comportamiento restrictivo del rol `rds_superuser`. Es necesario que al usuario de Aurora PostgreSQL se le conceda el privilegio `CONNECT` en la base de datos correspondiente para conectarse, incluso si se ha concedido al usuario el rol `rds_superuser`. Antes de las versiones 14.7 y 15.2 de Aurora PostgreSQL, un usuario podía conectarse a cualquier base de datos y tabla del sistema si

se le concedía el rol `rds_superuser`. Este comportamiento restrictivo se alinea con los compromisos de AWS y Amazon Aurora de mejora continua de la seguridad.
Actualice la lógica correspondiente de sus aplicaciones si la mejora anterior tiene alguna repercusión.

Control del acceso de los usuarios a la base de datos de PostgreSQL

Las nuevas bases de datos de PostgreSQL siempre se crean con un conjunto predeterminado de privilegios en el esquema `public` de la base de datos que permite a todos los usuarios y roles de la base de datos crear objetos. Estos privilegios permiten a los usuarios de la base de datos conectarse a la base de datos, or ejemplo, y crear tablas temporales mientras están conectados.

Para controlar mejor el acceso de los usuarios a las instancias de bases de datos que cree en el nodo principal del clúster de base de datos de Aurora PostgreSQL , le recomendamos que revoque estos privilegios de `public` predeterminados. Después de ello, conceda a continuación los privilegios específicos a los usuarios de base de datos de forma más detallada, como se muestra en el siguiente procedimiento.

Para configurar roles y privilegios para una nueva instancia de base de datos

Supongamos que está configurando una base de datos en un clúster de base de datos Aurora PostgreSQL de reciente creación para que lo utilicen varios investigadores, todos los cuales necesitan acceso de lectura y escritura a la base de datos.

1. Use `psql` (o pgAdmin) para conectarse a la instancia de base de datos principal de su clúster de base de datos de Aurora PostgreSQL

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Escriba la contraseña cuando se le solicite. El cliente `psql` se conecta y muestra la base de datos de conexión administrativa predeterminada, `postgres=>`, como el símbolo del sistema.
2. Para evitar que los usuarios de la base de datos creen objetos en el esquema `public`, realice una de las siguientes opciones:

   ```
   postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
   REVOKE
   ```

3. A continuación, cree una nueva instancia de base de datos:

```
postgres=> CREATE DATABASE lab_db;
CREATE DATABASE
```

4. Revoque todos los privilegios del esquema PUBLIC de esta nueva base de datos.

```
postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
REVOKE
```

5. Cree un rol para los usuarios de bases de datos.

```
postgres=> CREATE ROLE lab_tech;
CREATE ROLE
```

6. Otorgue a los usuarios de bases de datos que tengan este rol la posibilidad de conectarse a la base de datos.

```
postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
GRANT
```

7. Conceda a todos los usuarios que tengan el rol `lab_tech` todos los privilegios de esta base de datos.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. Cree usuarios de bases de datos de la siguiente manera:

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. Conceda a estos dos usuarios los privilegios asociados al rol lab_tech:

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
postgres=> GRANT lab_tech TO lab_user2;
GRANT ROLE
```

En este punto,`lab_user1` y `lab_user2` se pueden conectar a la base de datos de `lab_db`. En este ejemplo no se siguen las prácticas recomendadas para el uso empresarial, que pueden incluir la creación de varias instancias de base de datos, distintos esquemas y la concesión de permisos limitados. Para obtener más información y escenarios adicionales, consulte Administración de usuarios y roles de PostgreSQL.

Para obtener más información sobre los privilegios en las bases de datos de PostgreSQL, consulte el comando GRANT en la documentación de PostgreSQL.

Delegación y control de la administración de contraseñas de usuario

Como DBA, es posible que desee delegar la administración de contraseñas de usuario. O bien, puede que desee evitar que los usuarios de la base de datos cambien sus contraseñas o reconfiguren las restricciones de contraseña, como la duración de la contraseña. Para asegurarse de que solo los usuarios de la base de datos que elija puedan cambiar la configuración de contraseñas, puede activar la función de administración de contraseñas restringidas. Cuando activa esta función, solo pueden administrar contraseñas aquellos usuarios de base de datos a los que se les haya concedido el rol `rds_password`.

> **Note**
>
> Para utilizar la administración de contraseñas restringida, su el clúster de base de datos de PostgreSQL Aurora debe estar ejecutando Amazon Aurora para 10.6 o una versión posterior.

De forma predeterminada, esta función es `off`, tal y como se muestra en el ejemplo siguiente:

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

Para activar esta función, utilice un grupo de parámetros personalizado y cambie la configuración de `rds.restrict_password_commands` a 1. Asegúrese de reiniciar su Instancia de base de datos principal de Aurora PostgreSQL para que la configuración surta efecto.

Con esta función activa, se necesitan privilegios de `rds_password` para los siguientes comandos SQL:

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

Cambiar el nombre de un rol (`ALTER ROLE myrole RENAME TO newname`) también está restringido si la contraseña utiliza el algoritmo hash MD5.

Con esta función activa, intentar cualquiera de estos comandos SQL sin los permisos de rol `rds_password`, genera el siguiente error:

```
ERROR: must be a member of rds_password to alter passwords
```

Recomendamos que otorgar el `rds_password` solamente a unos cuantos roles que utilice únicamente para la administración de contraseñas. Si concede privilegios de `rds_password` a usuarios de bases de datos que no tengan privilegios de `rds_superuser`, también debe otorgarles el atributo `CREATEROLE`.

Asegúrese de que comprueba los requisitos de las contraseñas del lado del cliente, como el vencimiento y la complejidad necesaria. Si utiliza su propia utilidad del lado del cliente para cambios relacionados con la contraseña, la utilidad debe ser miembro de `rds_password` tener privilegios de `CREATE ROLE`.

Uso de SCRAM para el cifrado de contraseñas de PostgreSQL

El mecanismo de autenticación mediante desafío-respuesta discontinuo (SCRAM) es una alternativa al algoritmo de resumen de mensajes (MD5) predeterminado de PostgreSQL para cifrar contraseñas. El mecanismo de autenticación SCRAM se considera más seguro que MD5. Para obtener más información sobre estos dos enfoques diferentes para proteger las contraseñas, consulte Password Authentication (Autenticación de contraseñas) en la documentación de PostgreSQL.

Le recomendamos que utilice SCRAM en lugar de MD5 como esquema de cifrado de contraseñas para su clúster de base de datos de Aurora PostgreSQL. SCRAM se admite en la versión 10 de Aurora PostgreSQL y en todas las versiones superiores, principales y secundarias. Es un mecanismo criptográfico de desafío-respuesta que utiliza el algoritmo scram-sha-256 algorithm para la autenticación y el cifrado de contraseñas.

Es posible que deba actualizar las bibliotecas de las aplicaciones cliente para que sean compatibles con SCRAM. Por ejemplo, las versiones de JDBC anteriores a la 42.2.0 no admiten SCRAM. Para obtener más información, consulte PostgreSQL JDBC Driver (Controlador JDBC de PostgreSQL) en la documentación del controlador JDBC de PostgreSQL. Para ver una lista de otros controladores de PostgreSQL y la compatibilidad con SCRAM, consulte List of drivers (Lista de controladores) en la documentación de PostgreSQL.

Aurora PostgreSQL versión 14 y posteriores admite scram-sha-256 para el cifrado de contraseñas de forma predeterminada para los nuevos clústeres de base de datos. Para estas versiones, el grupo de parámetros del clúster de bases de datos predeterminado (`default.aurora-postgresql14`) tiene el valor de `password_encryption` establecido en scram-sha-256. No se admite en SCRAM para Aurora Serverless v1.

Configuración del clúster de base de datos de RDS para PostgreSQL para que requiera SCRAM

Para Aurora PostgreSQL 14.3 y versiones posteriores, puede requerir que el clúster de Aurora PostgreSQL DB acepte únicamente contraseñas que utilicen el algoritmo scram-sha-256.

> **Important**
>
> En el caso de los proxies RDS existentes con bases de datos de PostgreSQL, si modifica la autenticación de la base de datos para utilizar únicamente `SCRAM`, el proxy dejará de estar disponible durante un máximo de 60 segundos. Para evitar este problema, lleve a cabo alguna de las siguientes operaciones:
>
> - Asegúrese de que la base de datos permita la autenticación `SCRAM` y `MD5`.
> - Para utilizar únicamente la autenticación `SCRAM`, cree un nuevo proxy, migre el tráfico de la aplicación al nuevo proxy y, a continuación, elimine el proxy previamente asociado a la base de datos.

Antes de realizar cambios en el sistema, asegúrese de entender el proceso completo, como se indica a continuación:

- Obtenga información sobre todos los roles y el cifrado de las contraseñas de todos los usuarios de la base de datos.
- Compruebe de nuevo la configuración de los parámetros del clúster de base de datos de RDS para PostgreSQL correspondiente a los parámetros que controlan el cifrado de las contraseñas.

- Si el clúster de base de datos RDS para Aurora PostgreSQL utiliza un grupo de parámetros predeterminado, deberá crear un grupo de parámetros de clúster de base de datos personalizado y aplicarlo al clúster de base de datos de Aurora PostgreSQL para poder modificar los parámetros cuando sea necesario. Si el clúster de base de datos de Aurora PostgreSQL utiliza un grupo de parámetros personalizado, puede modificar los parámetros necesarios más adelante en el proceso, según sea necesario.
- Cambie el parámetro `password_encryption` por `scram-sha-256`.
- Notifique a todos los usuarios de la base de datos que deben actualizar las contraseñas. Haga lo mismo con su cuenta de `postgres`. Las nuevas contraseñas se cifran y almacenan mediante el algoritmo scram-sha-256.
- Verifique que todas las contraseñas están cifradas con el tipo de cifrado.
- Si todas las contraseñas utilizan scram-sha-256, puede cambiar el parámetro `rds.accepted_password_auth_method` de `md5+scram` a `scram-sha-256`.

> **Warning**
>
> Después de cambiar `rds.accepted_password_auth_method` a scram-sha-256 únicamente, no podrá conectarse ningún usuario (rol) con una contraseña cifrada con `md5`.

Preparativos para requerir SCRAM en clúster de base de datos de Aurora PostgreSQL

Antes de realizar cambios en el clúster de base de datos de Aurora PostgreSQL , compruebe todas las cuentas de usuario de base de datos existentes. Compruebe también el tipo de cifrado utilizado para las contraseñas. Puede hacer estas tareas con la extensión `rds_tools`. Para ver qué versiones de PostgreSQL son compatibles con `rds_tools`, consulte Versiones de extensión para Amazon RDS para PostgreSQL.

Para obtener una lista de usuarios de base de datos (roles) y métodos de cifrado de contraseñas

1. Use `psql` para conectarse a la instancia principal del clúster de base de datos de Aurora PostgreSQL , tal como se muestra a continuación.

   ```
   psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Instale la extensión de `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. Obtenga una lista de los roles y el cifrado.

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

Se muestra una salida similar a la siguiente.

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

Creación de un grupo de parámetros del clúster de base de datos personalizado

> **Note**
>
> Si el clúster de base de datos de RDS para Aurora PostgreSQL ya utiliza un grupo de parámetros personalizado, no necesita crear uno nuevo.

Para obtener información general sobre los grupos de parámetros para Aurora, consulte Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

El tipo de cifrado de contraseñas que se usa para las contraseñas se establece a un parámetro, `password_encryption`. El cifrado que permite el clúster de base de datos de Aurora PostgreSQL se establece a otro parámetro, `rds.accepted_password_auth_method`. Para cambiar cualquiera de los valores predeterminados, es necesario crear un grupo de parámetros de clúster de base de datos personalizado y aplicarlo al clúster .

También puede utilizar la Consola de administración de AWS o la API de RDS para crear un grupo de parámetros de clúster de base de datos personalizado. Para obtener más información, consulte Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Ahora puede asociar el grupo de parámetros personalizado con su instancia de base de datos.

Para crear un grupo de parámetros del clúster de base de datos personalizado

1. Utilice el comando `create-db-cluster-parameter-group` de la CLI para crear el grupo de parámetros personalizado para el clúster. En el siguiente ejemplo se utiliza `aurora-postgresql13` como origen de este grupo de parámetros personalizado.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-lab-scram-passwords' \
     --db-parameter-group-family aurora-postgresql13  --description 'Custom DB cluster parameter group for SCRAM'
   ```

   Para Windows:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-lab-scram-passwords" ^
     --db-parameter-group-family aurora-postgresql13  --description "Custom DB cluster parameter group for SCRAM"
   ```

   Ahora puede asociar el grupo de parámetros personalizado con el clúster.

2. Utilice el comando `modify-db-cluster` de la CLI para aplicar este grupo de parámetros personalizado al clúster de base de datos de Aurora PostgreSQL.

   Para Linux, macOS o Unix:

   ```
   aws rds modify-db-cluster --db-cluster-identifier 'your-instance-name' \
           --db-cluster-parameter-group-name "docs-lab-scram-passwords
   ```

   Para Windows:

   ```
   aws rds modify-db-cluster --db-cluster-identifier "your-instance-name" ^
           --db-cluster-parameter-group-name "docs-lab-scram-passwords
   ```

Para volver a sincronizar el clúster de base de datos de Aurora PostgreSQL con el grupo de parámetros del clúster de base de datos personalizado, reinicie la instancia principal y todas las demás instancias del clúster.

Configuración del cifrado de contraseñas para utilizar SCRAM

El mecanismo de cifrado de contraseñas que utiliza un clúster de base de datos de Aurora PostgreSQL se establece al grupo de parámetros de clúster de base de datos en el parámetro `password_encryption`. Los valores permitidos son: no establecido, `md5` o `scram-sha-256`. El valor predeterminado depende de la versión de Aurora PostgreSQL: del modo que se indica a continuación:

- Aurora PostgreSQL 14: el valor predeterminado es `scram-sha-256`
- Aurora PostgreSQL 13: el valor predeterminado es `md5`

Con un grupo de parámetros de clúster de base de datos personalizado adjuntado al clúster de base de datos de Aurora PostgreSQL, puede modificar los valores del parámetro de cifrado de contraseñas.

Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Para cambiar la configuración de cifrado de contraseñas a scram-sha-256

- Cambie el valor del cifrado de contraseñas a scram-sha-256, como se muestra a continuación. El cambio se puede aplicar inmediatamente porque el parámetro es dinámico, por lo que no se requiere un reinicio para que el cambio surta efecto.

  Para Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name \
```

```
  'docs-lab-scram-passwords' --parameters
 'ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate'
```

Para Windows:

```
aws rds modify-db-parameter-group --db-parameter-group-name ^
  "docs-lab-scram-passwords" --parameters
 "ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate"
```

Migración de las contraseñas de los roles de usuario a SCRAM

Puede migrar las contraseñas de los roles de usuario a SCRAM, tal y como se describe a continuación.

Para migrar las contraseñas de usuario (rol) de base de datos de MD5 a SCRAM

1. Inicie sesión como usuario administrador (nombre de usuario predeterminado, `postgres`) como se muestra a continuación.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Compruebe la configuración del parámetro `password_encryption` en la instancia de base de datos de RDS para PostgreSQL con el siguiente comando.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 md5
 (1 row)
```

3. Cambie el valor de este parámetro a scram-sha-256. Se trata de un parámetro dinámico, por lo que no es necesario reiniciar la instancia después de realizar este cambio. Compruebe de nuevo el valor para asegurarse de que ahora está establecido en `scram-sha-256`, como se indica a continuación.

```
postgres=> SHOW password_encryption;
 password_encryption
```

```
---------------------
 scram-sha-256
 (1 row)
```

4. Notifique a todos los usuarios de base de datos que deben cambiar la contraseña. Asegúrese de cambiar también su propia contraseña para la cuenta `postgres` (el usuario de base de datos con privilegios `rds_superuser`).

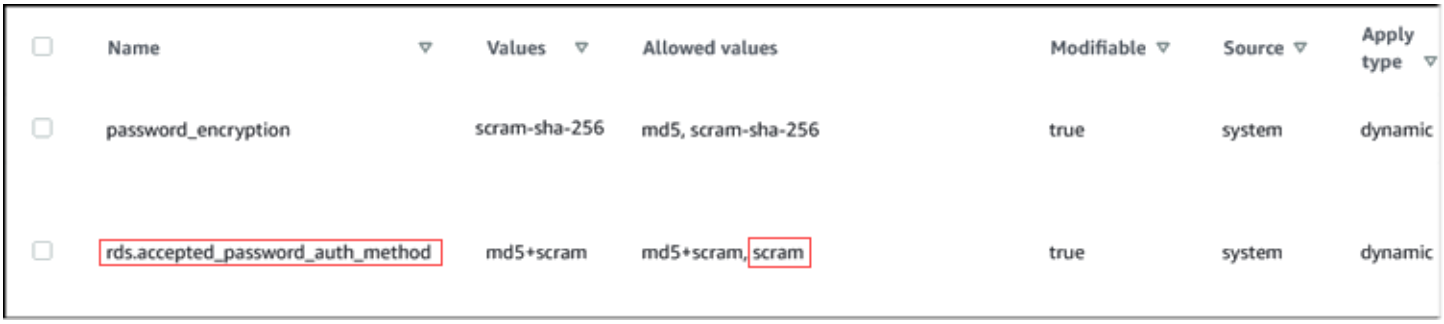
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

5. Repita el proceso para todas las bases de datos del clúster de base de datos de Aurora PostgreSQL.

Cambio del parámetro para requerir SCRAM

Este es el último paso del proceso. Después de realizar el cambio en el siguiente procedimiento, ninguna cuenta de usuario (rol) que aún utilice el cifrado `md5` para las contraseñas podrá iniciar sesión en el clúster de base de datos PostgreSQL de Aurora.

Con `rds.accepted_password_auth_method` se especifica el método de cifrado que el clúster de base de datos de Aurora PostgreSQL acepta una contraseña de usuario durante el proceso de inicio de sesión. El valor predeterminado es `md5+scram`, lo que significa que se acepta cualquiera de los dos métodos. En la siguiente imagen, puede encontrar la configuración predeterminada de este parámetro.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Los valores permitidos para este parámetro son:`md5+scram` o `scram` solo. El cambio de este valor de parámetro a `scram` lo convierte en un requisito.

Para cambiar el valor de parámetro para requerir la autenticación SCRAM para las contraseñas

1. Verifique que todas las contraseñas de los usuarios de base de datos del clúster de base de datos de Aurora PostgreSQL utilizan `scram-sha-256` para el cifrado de contraseña. Para ello, consulte `rds_tools` para el rol (usuario) y el tipo de cifrado, de la siguiente manera.

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. Repita la consulta en todas las instancias de base de datos del clúster de base de datos de Aurora PostgreSQL.

   Si todas las contraseñas usan scram-sha-256, puede continuar.

3. Cambie el valor de la autenticación de contraseña aceptada a scram-sha-256, como se indica a continuación.

   Para Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-
lab-scram-passwords' \
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

   Para Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-
lab-scram-passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Protección de los datos de Aurora PostgreSQL con SSL/TLS

Amazon RDS admite el cifrado mediante Capa de conexión segura (SSL) y Transport Layer Security (TLS) para los clústeres de base de datos de Aurora PostgreSQL. Con SSL/TLS, puede cifrar conexiones entre sus aplicaciones y sus clústeres de base de datos de Aurora PostgreSQL. También puede obligar a todas las conexiones con su clúster de base de datos de Aurora PostgreSQL a usar SSL/TLS. Amazon Aurora PostgreSQL admite Transport Layer Security (TLS), versiones 1.1 y 1.2. Se recomienda utilizar TLS 1.2 para conexiones cifradas. Se ha agregado compatibilidad con TLSv1.3 para las siguientes versiones de Aurora PostgreSQL:

- Versión 15.3 y versiones posteriores
- Versión 14.8 y versiones posteriores a la 14
- Versión 13.11 y versiones posteriores a la 13
- Versión 12.15 y versiones posteriores a la 12
- Versión 11.20 y versiones posteriores a la 11

Para obtener la información general acerca de la compatibilidad con SSL/TLS y las bases de datos de PostgreSQL, consulte Compatibilidad con SSL en la documentación de PostgreSQL. Para obtener información sobre el uso de una conexión SSL/TLS a través de JDBC, consulte Configuración del cliente en la documentación de PostgreSQL.

Temas

- Requerir una conexión SSL/TLS a un clúster de base de datos de Aurora PostgreSQL
- Determinar el estado de la conexión SSL/TLS
- Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora PostgreSQL

La compatibilidad con SSL/TLS está disponible en todas las regiones de AWS para Aurora PostgreSQL. Amazon RDS crea un certificado SSL/TLS destinado al clúster de base de datos de Aurora PostgreSQL cuando se crea el clúster de base de datos. Si se habilita la verificación con certificado SSL/TLS, el certificado incluye el punto de enlace del clúster de base de datos como nombre común (CN) que el certificado de SSL/TLS debe proteger frente a los ataques de suplantación.

Para conectar con un clúster de base de datos de Aurora PostgreSQL a través de SSL/TLS

1. Descargue el certificado.

   Para obtener más información acerca de cómo descargar certificados, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.
2. Importe el certificado en su sistema operativo.
3. Conéctese a su clúster de base de datos de Aurora PostgreSQL a través de SSL/TLS.

   Cuando se conecte mediante SSL/TLS, su cliente podrá elegir verificar la cadena de certificados o no. Si sus parámetros de conexión especifican `sslmode=verify-ca` o `sslmode=verify-full`, su cliente precisa que los certificados de CA de RDS estén en su almacén de confianza o se haga referencia a ellos en la URL de conexión. Este requisito es para verificar la cadena de certificados que firma su certificado de base de datos.

   Cuando un cliente, como psql o JDBC, está configurado con soporte de SSL/TLS, primero el cliente intenta conectarse a la base de datos con SSL/TLS de forma predeterminada. Si el cliente no puede conectarse mediante SSL/TLS, vuelve a la conexión sin SSL/TLS. De forma predeterminada, la opción `sslmode` para los clientes basados en JDBC y libpq está establecida en `prefer`.

   Use el parámetro `sslrootcert` para hacer referencia al certificado, por ejemplo: `sslrootcert=rds-ssl-ca-cert.pem`.

A continuación, aparece un ejemplo de cómo usar psql para conectarse a un clúster de base de datos de Aurora PostgreSQL.

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
    "dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Requerir una conexión SSL/TLS a un clúster de base de datos de Aurora PostgreSQL

Para exigir conexiones SSL/TLS al clúster de base de datos de Aurora PostgreSQL, use el parámetro `rds.force_ssl`.

- Para requerir conexiones SSL/TLS, establezca el valor del parámetro `rds.force_ssl` en 1 (activado).

- Para desactivar las conexiones SSL/TLS requeridas, establezca el valor del parámetro `rds.force_ssl` en 0 (desactivado).

El valor predeterminado de este parámetro depende de la versión de Aurora PostgreSQL:

- Para las versiones 17 y posteriores de Aurora PostgreSQL: el valor predeterminado es 1 (activado).
- Para las versiones 16 y más antiguas de Aurora PostgreSQL: el valor predeterminado es 0 (desactivado).

> **Note**
>
> Cuando realiza una actualización de versión principal de Aurora PostgreSQL versión 16 o anterior a la versión 17 o posterior, el valor predeterminado del parámetro cambia de 0 (desactivado) a 1 (activado). Este cambio puede provocar errores de conectividad en las aplicaciones que no estén configuradas para SSL. Puede volver al comportamiento predeterminado anterior estableciendo este parámetro en 0 (desactivado).

Para obtener más información acerca de la gestión de parámetros, consulte Grupos de parámetros para Amazon Aurora.

Al actualizar el parámetro `rds.force_ssl`, se define también el parámetro `ssl` de PostgreSQL como 1 (activado) y se modifica el archivo `pg_hba.conf` del clúster de base de datos para permitir la nueva configuración de SSL/TLS.

Cuando el parámetro `rds.force_ssl` se haya definido en 1 para un clúster de base de datos, al conectarse verá una salida similar a la siguiente, que indica que ahora se requiere SSL/TLS:

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Determinar el estado de la conexión SSL/TLS

El estado cifrado de su conexión se muestra en el banner de inicio de sesión al establecer conexión con el clúster de base de datos.

```
Password for user master:
psql (9.3.12)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

También puede cargar la extensión `sslinfo` y llamar después a la función `ssl_is_used()` para determinar si se está utilizando SSL/TLS. La función devuelve `t` si la conexión usa SSL/TLS; de lo contrario, devuelve `f`.

```
postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
---------
t
(1 row)
```

Puede utilizar el comando `select ssl_cipher()` para determinar el cifrado SSL/TLS:

```
postgres=> select ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

Si habilita `set rds.force_ssl` y reinicia el clúster de la base de datos, las conexiones que no usen SSL se rechazarán con el siguiente mensaje:

```
$ export PGSSLMODE=disable
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database
 "postgres", SSL off
$
```

Para obtener información acerca de la opción `sslmode`, consulte Funciones de control de conexión de la base de datos en la documentación de PostgreSQL.

Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora PostgreSQL

Mediante el uso de conjuntos de cifrado configurables, puede tener más control sobre la seguridad de las conexiones de su base de datos. Puede especificar una lista de conjuntos de cifrado que desea permitir para proteger las conexiones SSL/TLS del cliente a su base de datos. Con conjuntos de cifrado configurables, puede controlar el cifrado de conexión que acepta el servidor de base de datos. Esto ayuda a evitar el uso de cifrados inseguros o obsoletos.

Los conjuntos de cifrado configurables son compatibles con las versiones 11.8 y posteriores de Aurora PostgreSQL.

Con el fin de especificar la lista de cifrados permitidos para cifrar conexiones, modifique el parámetro del clúster `ssl_ciphers`. Establezca el parámetro `ssl_ciphers` en una cadena de valores de cifrado separados por comas en un grupo de parámetros de clúster mediante la Consola de administración de AWS, la AWS CLI o la API de RDS. Para configurar los parámetros de clúster, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

En la tabla siguiente, se muestran los cifrados compatibles para las versiones de motor válidas de Aurora PostgreSQL.

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
9.6, 10.20 y versiones anteriores, 11.15 versiones anteriores, 12.10 versiones anteriores, 13.6 versiones anteriores	DHE-RSA-AES128-SHA	Sí No	No Sí	No No

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
	DHE-RSA-AES128-SHA256	No	Sí	No
	DHE-RSA-AES128-GCM-SHA256	No	Sí	No
	DHE-RSA-AES256-SHA	No	Sí	No
	DHE-RSA-AES256-SHA256	Sí	Sí	No
	DHE-RSA-AES256-GCM-SHA384	No	Sí	No
	ECDHE-ECDSA-AES256-SHA	Sí	Sí	No
	ECDHE-ECDSA-AES256-GCM-SHA384	No	Sí	No
	ECDHE-RSA-AES256-SHA384	No	Sí	No
	ECDHE-RSA-AES128-SHA	Sí	Sí	No
	ECDHE-RSA-AES128-SHA256	No	Sí	No
	ECDHE-RSA-AES128-GCM-SHA256	No	Sí	No
	ECDHE-RSA-AES256-SHA	No	Sí	No

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
	ECDHE-RSA-AES256-GCM-SHA384			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
10.21, 11.16, 12.11, 13.7, 14.3 y 14.4	ECDHE-RSA-AES128-SHA	Sí	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Sí	Sí	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Sí	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA			
	TLS_ECDHE_RSA_WITH			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
	_AES_256_ GCM_SHA384 TLS_RSA_W ITH_AES_2 56_GCM_SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_W ITH_AES_1 28_GCM_SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_WITH _CHACHA20 _POLY1305_SHA256			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
10.22, 11.17, 12.12, 13.8, 14.5 y 15.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	No	Sí	No
		Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Sí	Sí	No
		No	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	No	Sí	No
		Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sí	Sí	No
		Sí	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
11.20, 12.15, 13.11, 14.8, 15.3, 16.1 y versiones posteriores	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	No	Sí	No
		Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Sí	Sí	No
		No	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	No	Sí	No
		Sí	Sí	No
		No	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sí	Sí	No
		Sí	Sí	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	No	Sí	No
		No	No	Sí
		No	No	Sí
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256			

Versiones del motor de Aurora PostgreSQL	Cifrados compatibles	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_AES_256_GCM_SHA384			

También puede utilizar el comando de la CLI describe-motor-default-cluster-parameters para determinar qué conjuntos de cifrado se admiten actualmente para una familia de grupos de parámetros específica. El siguiente ejemplo muestra cómo obtener los valores permitidos para el parámetro `ssl_cipher` de clúster para Aurora PostgreSQL 11.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
postgresql11

    ...some output truncated...
 {
  "ParameterName": "ssl_ciphers",
  "Description": "Sets the list of allowed TLS ciphers to be used on secure
 connections.",
  "Source": "engine-default",
  "ApplyType": "dynamic",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,
  ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-
SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-
SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384,

 TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,T

 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AE
  "IsModifiable": true,
  "MinimumEngineVersion": "11.8",
  "SupportedEngineModes": [
   "provisioned"
  ]
 },
    ...some output truncated...
```

El parámetro `ssl_ciphers` se establece de forma predeterminada en todos los conjuntos de cifrado permitidos. Para obtener más información sobre los cifrados, consulte la variable ssl_ciphers en la documentación de PostgreSQL.

Uso del enmascaramiento dinámico con Aurora PostgreSQL

El enmascaramiento de datos dinámico es una característica de seguridad que protege la información confidencial de las bases de datos de Aurora PostgreSQL al controlar la forma en que los usuarios ven los datos en el momento de la consulta. Aurora lo implementa a través de la extensión `pg_columnmask`. `pg_columnmask` proporciona una protección de datos por columna que complementa los mecanismos nativos de control de acceso granular y de seguridad por fila de PostgreSQL.

Con `pg_columnmask`, puede crear políticas de enmascaramiento que determinan la visibilidad de los datos en función de los roles de los usuarios. Cuando los usuarios consultan tablas con políticas de enmascaramiento, Aurora PostgreSQL aplica la función de enmascaramiento adecuada en el momento de la consulta en función del rol del usuario y del peso de la política. Los datos subyacentes permanecen sin cambios en el almacenamiento.

`pg_columnmask` admite las siguientes capacidades:

- Funciones de enmascaramiento integradas y personalizadas: utilice funciones prediseñadas para patrones comunes, como el enmascaramiento del correo electrónico y el texto, o cree sus propias funciones personalizadas para proteger la información confidencial (PII) mediante políticas de enmascaramiento basadas en SQL.
- Múltiples estrategias de enmascaramiento: oculte completamente la información, sustituya valores parciales por caracteres comodín o defina enfoques de enmascaramiento personalizados.
- Priorización de políticas: defina varias políticas para una sola columna. Use ponderaciones para determinar qué política de enmascaramiento debe usarse cuando se apliquen varias políticas a una columna. Aurora PostgreSQL aplica políticas según el peso y la pertenencia a los roles de usuario.

`pg_columnmask` está disponible en la versión 16.10 y superior de Aurora PostgreSQL y en la versión 17.6 y superior. Está disponible sin costo adicional.

Introducción al enmascaramiento dinámico

Para enmascarar los datos de forma dinámica, debe instalar la extensión `pg_columnmask` en la base de datos y crear políticas de enmascaramiento para las tablas. El proceso de configuración implica la verificación de los requisitos previos, la instalación de la extensión, la configuración de los roles, la creación de políticas y las pruebas de validación.

Instalación y configuración de la extensión

Conéctese al clúster de Aurora PostgreSQL mediante el editor de consultas de la consola de RDS o un cliente de PostgreSQL, como psql, con las credenciales de rds_superuser (usuario maestro).

Ejecute el comando de creación de extensiones para habilitar la funcionalidad `pg_columnmask`:

```
CREATE EXTENSION pg_columnmask;
```

Este comando instala la extensión `pg_columnmask`, crea las tablas de catálogo necesarias y registra las funciones de enmascaramiento integradas. La instalación de la extensión es específica de la base de datos, lo que significa que debe instalarla de forma independiente en cada base de datos en la que se requiera la funcionalidad.

> **Note**
>
> Las conexiones realizadas antes de instalar esta extensión seguirán mostrando los datos desenmascarados. Ciérrela y vuelva a conectarla para solucionar este problema.

Verifique la instalación de la extensión comprobando las funciones de enmascaramiento disponibles:

```
SELECT proname FROM pg_proc
    WHERE pronamespace = 'pgcolumnmask'::regnamespace AND proname LIKE 'mask_%';
    proname
--------Output --------
 mask_email
 mask_text
 mask_timestamp
(3 rows)
```

Procedimientos para administrar políticas de enmascaramiento de datos

Puede administrar las políticas de enmascaramiento mediante los procedimientos proporcionados por la extensión `pg_columnmask`. Para crear, modificar o eliminar políticas de enmascaramiento, debe contar con uno de los siguientes privilegios:

- Propietario de la tabla en la que va a crear la política de `pg_columnmask`.
- Miembro de `rds_superuser`.
- Miembro del rol de mánager de la política de `pg_columnmask` establecido por el parámetro `pgcolumnmask.policy_admin_rolname`.

El siguiente comando crea una tabla que se utiliza en las siguientes secciones:

```
CREATE TABLE public.customers (
    id SERIAL PRIMARY KEY,
    name TEXT,
    phone TEXT,
    address TEXT,
    email TEXT
);
```

CREATE_MASKING_POLICY

El siguiente procedimiento crea una nueva política de enmascaramiento para una tabla de usuarios:

Sintaxis

```
create_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Argumentos

Parámetro	Tipo de datos	Descripción
`policy_name`	NAME	Nombre de la política de enmascaramiento. Debe ser único por tabla.
`table_name`	REGCLASS	El nombre cualificado o no cualificado o nulo de la tabla para aplicar la política de enmascaramiento.
`masking_expressions`	JSONB	Objeto JSON que contiene los pares de nombre de columna y función de enmascara miento. Cada clave es el nombre de una columna y su valor es la expresión de enmascaramiento que se va a aplicar a esa columna.
`roles`	NAME[]	Los roles a los que se aplica esta política de enmascara miento. Predeterminado es PUBLIC.
`weight`	INT	Peso de la política de enmascaramiento. Cuando se apliquen varias políticas a la consulta de un usuario determinado, la política con el peso más alto (número entero más alto) se aplicará a cada columna enmascarada. El valor predeterminado es 0. No hay dos políticas de enmascaramiento de la tabla

Parámetro	Tipo de datos	Descripción
		que puedan tener el mismo peso.

Tipo de devolución

Ninguno

Example de crear una política de enmascaramiento que oculte la columna de correo electrónico del rol **`test_user`**:

```
CALL pgcolumnmask.create_masking_policy(
    'customer_mask',
    'public.customers',
    JSON_OBJECT('{
        "email", "pgcolumnmask.mask_email(email)"
    }')::JSONB,
    ARRAY['test_user'],
    100
);
```

ALTER_MASKING_POLICY

Este procedimiento modifica una política de enmascaramiento existente. `ALTER_MASKING_POLICY` puede modificar las expresiones de enmascaramiento de la política, el conjunto de roles a los que se aplica la política y el peso de la política de enmascaramiento. Cuando se omite uno de esos parámetros, la parte correspondiente de la política permanece inalterada.

Sintaxis

```
alter_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Argumentos

Parámetro	Tipo de datos	Descripción
`policy_name`	NAME	El nombre existente de la política de enmascaramiento.
`table_name`	REGCLASS	El nombre cualificado o no cualificado o nulo de la tabla que contiene la política de enmascaramiento.
`masking_expressions`	JSONB	Nuevo objeto JSON que contiene los pares de nombre de columna y función de enmascaramiento, de lo contrario, NULL.
`roles`	NAME[]	La lista de roles nuevos a los que se aplica esta política de enmascaramiento o NULL en caso contrario.
`weight`	INT	Nuevo peso para la política de enmascaramiento o NULL en caso contrario.

Tipo de devolución

Ninguno

Example de agregar el rol de analista a una política de enmascaramiento existente sin cambiar otros atributos de la política.

```
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'public.customers',
    NULL,
    ARRAY['test_user', 'analyst'],
    NULL
);
```

```
-- Alter the weight of the policy without altering other details
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'customers',
    NULL,
    NULL,
    4
);
```

DROP_MASKING_POLICY

Este procedimiento elimina una política de enmascaramiento existente.

Sintaxis

```
drop_masking_policy(
        policy_name,
        table_name)
```

Argumentos

Parámetro	Tipo de datos	Descripción
`policy_name`	NAME	El nombre existente de la política de enmascaramiento.
`table_name`	REGCLASS	El nombre cualificado o no cualificado o nulo de la tabla que contiene la política de enmascaramiento.

Tipo de devolución

Ninguno

Example de eliminar la política de enmascaramiento customer_mask

```
-- Drop a masking policy
```

```
    CALL pgcolumnmask.drop_masking_policy(
        'customer_mask',
        'public.customers',
    );
```

Identificadores de escape en el procedimiento de DDL de la política de enmascaramiento

Al crear políticas de enmascaramiento de datos con los identificadores entre comillas, es necesario utilizarlas adecuadamente para garantizar que las referencias a los objetos y la aplicación de la política sean correctas. Para utilizar los identificadores entre comillas en los procedimientos de administración de la política de enmascaramiento `pg_columnmask`:

- Nombre de la política: debe estar entre comillas dobles.
- Nombre de la tabla: el nombre del esquema y el nombre de la tabla deben escribirse entre comillas dobles de forma individual cuando sea necesario.
- Expresiones de enmascaramiento: los nombres de las columnas y funciones de las expresiones de enmascaramiento deben ir entre comillas dobles y las propias comillas deben ir precedidas de una barra invertida.
- Roles: la matriz de nombres de roles se cita automáticamente. El nombre del rol debe coincidir exactamente con el nombre que se muestra en `pg_roles` al distinguir mayúsculas y minúsculas.

Example de la sintaxis de escape y comillas

En este ejemplo, se muestra la sintaxis de escape y comillas adecuada al crear políticas de enmascaramiento para tablas, columnas, funciones y roles que utilizan nombres en mayúsculas y minúsculas o que requieren identificadores entre comillas en Aurora PostgreSQL.

```
-- Create a table and columns with mixed case name
CREATE TABLE public."Employees" (
    "Name" TEXT,
    "Email" TEXT,
    ssn VARCHAR(20)
);

-- Create a role with mixed case name
CREATE ROLE "Masked_user";
```

```
-- Create a function with mixed case name
CREATE OR REPLACE FUNCTION public."MaskEmail"(text)
    RETURNS character varying
    LANGUAGE plpgsql
    IMMUTABLE PARALLEL SAFE
    AS $$ BEGIN
        RETURN 'XXXXXXXX'::text;
    END $$;

-- Now use these objects with mixed case names in
-- masking policy management procedures
CALL pgcolumnmask.create_masking_policy(
    '"Policy1"',  -- policy name should be surrounded with double quotes for quoting
    'public."Employees"', -- table and schema name should be individually
                          -- surrounded with double quotes for quoting
    JSON_OBJECT('{
        "\"Email\"", "\"MaskEmail\"(\"Email\")"
    }')::JSONB, -- masking expression should have double quotes around function names
                -- and columns names etc when needed. Also the double quotes itself
                -- should be escaped using \ (backslash) since this is a JSON string
    ARRAY['Masked_user'], -- Rolename do not need quoting
                          -- (this behaviour may change in future release)
    100
);

SELECT * FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'Employees';
-[ RECORD 1 ]-----+-------------------------------------
schemaname        | public
tablename         | Employees
policyname        | Policy1
roles             | {Masked_user}
masked_columns    | {Email}
masking_functions | {"(\"MaskEmail\"(\"Email\"))::text"}
weight            | 100
```

Vistas administrativas

Puede revisar toda la política de `pg_columnmask` mediante la vista administrativa de `pgcolumnmask.pg_columnmask_policies` de acceso público. La siguiente información está disponible mediante esta vista. La vista solo devuelve las políticas de enmascaramiento que son propiedad del usuario actual.

Nombre de la columna	Tipo de datos:	Descripción
schemaname	NAME	Esquema de la relación a la que está asociada la política
tablename	NAME	Nombre de la relación a la que está asociada la política
policyname	NAME	Nombre de la política de enmascaramiento, todas las políticas de enmascaramiento tienen nombres únicos
roles	TEXT[]	Rol al que se aplica la política.
masked_columns	TEXT[]	Columnas de enmascara miento
masking_functions	TEXT[]	Funciones de enmascara miento
weight	INT	Peso de la política adjunta

Funciones de enmascaramiento de datos predefinidas

La extensión `pg_columnmask` proporciona funciones de utilidad integradas escritas en lenguaje C (para una ejecución más rápida) que se pueden utilizar como expresión de enmascaramiento para las políticas de `pg_columnmask`.

mask_text

Una función para enmascarar datos de texto con opciones de visibilidad configurables.

Argumentos

Parámetro	Tipo de datos	Descripción
`input`	TEXT	La cadena de texto original que se va a enmascarar

Parámetro	Tipo de datos	Descripción
`mask_char`	CHAR (1)	Carácter utilizado para enmascarar (predeterminado: “X”)
`visible_prefix`	INT	Número de caractere s al principio del texto ingresado que permanecerán desenmascarados (predeter minado: 0)
`visible_suffix`	INT	Número de caracteres al final del texto ingresado que permanecerán desenmasc arados (predeterminado: 0)
`use_hash_mask`	BOOLEANO	Si es TRUE, utiliza un enmascaramiento basado en hash en lugar de mask_char (predeterminado: FALSE)

Example de usar diferentes opciones de enmascaramiento

Enmascaramiento de toda la cadena de entrada con el carácter “X” predeterminado

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World');
  mask_text
-------------
 XXXXXXXXXXX
```

Uso del argumento `mask_char` para enmascarar la entrada de texto con un carácter diferente

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*');
  mask_text
-------------
 ***********
```

Uso de los parámetros `visible_prefix` y `visible_suffix` para controlar el número de caracteres que permanecen desenmascarados al principio y al final del texto

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 5, 1);
  mask_text
-------------
 Hello*****d
```

Cuando `use_hash_mask` es verdadero, la cadena de entrada está enmascarada con caracteres aleatorios, el argumento `mask_char` se ignora pero se sigue respetando `visible_prefix` y `visible_suffix`

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 2, 2, true);
  mask_text
-------------
 Hex36dOHild
```

mask_timestamp

Parámetro	Tipo de datos	Descripción
`ts_to_mask`	TIMESTAMP	La marca temporal original que se va a enmascarar
`mask_part`	TEXT	Especifica qué parte de la marca temporal se debe enmascarar (predeterminado: “todas”) Valores válidos: “año”, “mes”, “día”, “hora”, “minuto”, “segundo”, “todo”
`mask_value`	TIMESTAMP	El valor de la marca de tiempo que se va a usar para enmascarar (predeterminado: “1900-01-01 00:00:00”)

Example de cómo usar **`mask_timestamps`**

Estos ejemplos muestran el enmascaramiento completo de la marca de tiempo con un valor predeterminado, el enmascaramiento parcial de componentes específicos de la marca de tiempo (solo en el año) y el enmascaramiento con un valor de reemplazo personalizado.

Enmascaramiento completo del valor de entrada con la marca de tiempo predeterminada

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00');
   mask_timestamp
---------------------
 1900-01-01 00:00:00
```

Enmascaramiento solo de una parte de la marca de tiempo, por ejemplo, solo el año

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'year');
   mask_timestamp
---------------------
 1900-06-15 14:30:00
```

Cambio del valor enmascarado de la marca de tiempo mediante el argumento `mask_value`

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'all', '2012-12-12
 12:12:12');
   mask_timestamp
---------------------
 2012-12-12 12:12:12
```

mask_timestamp

Una función para enmascarar las direcciones de correo electrónico y, al mismo tiempo, preservar la estructura del correo electrónico.

Parámetro	Tipo de datos	Descripción
`input`	TEXT	La dirección de correo electrónico original que se va a enmascarar

Parámetro	Tipo de datos	Descripción
`mask_char`	CHAR (1)	Carácter utilizado para enmascarar (predeterminado: “X”)
`mask_local`	BOOLEANO	Si es TRUE, enmascara la parte local del correo electróni co (antes de @) (predeter minado: TRUE)
`mask_domain`	BOOLEANO	Si es TRUE, enmascara la parte de dominio del correo electrónico (después de @) (predeterminado: TRUE)

Example de cómo usar **`mask_email`**

Estos ejemplos muestran el enmascaramiento completo del correo electrónico, los caracteres de máscara personalizados y el enmascaramiento selectivo de la parte local o de la parte del dominio de la dirección de correo electrónico.

Enmascaramiento completo

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com');
    mask_email
------------------
 XXXX@XXXXXXX.com
```

Uso de `mask_char` para cambiar el carácter utilizado para enmascarar

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*');
    mask_email
------------------
 ****@*******.com
```

Uso de `mask_local` y `mask_domain` para controlar el enmascaramiento en el entorno local y en el dominio

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', true, false);
    mask_email
------------------
 ****@example.com

postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', false, true);
    mask_email
------------------
 user@*******.com
```

Implementación de pg_columnmask en un flujo de trabajo integral

En esta sección se muestra una implementación completa del `pg_columnmask` mediante el uso de un ejemplo de tabla de empleados con información confidencial. Obtendrá información sobre cómo crear funciones de enmascaramiento personalizadas, definir múltiples políticas de enmascaramiento con diferentes niveles de peso para roles diferentes (pasante, de soporte, analista) y a observar cómo los usuarios que pertenecen a uno o varios roles ven diferentes niveles de datos enmascarados. Los ejemplos también incluyen el comportamiento de enmascaramiento en las instrucciones DML con cláusulas RETURNING, los activadores que utilizan tablas frente a vistas y las operaciones de administración de políticas, como el cambio de nombre, la modificación de los pesos y la limpieza.

1. Cree una tabla de ejemplo con información confidencial:

```
CREATE SCHEMA hr;

CREATE TABLE hr.employees (
    id INT PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT,
    ssn TEXT,
    salary NUMERIC(10,2)
 );

INSERT INTO hr.employees VALUES
    (1, 'John Doe', 'john.doe@example.com', '123-45-6789', 50000.00),
    (2, 'Jane Smith', 'jane.smith@example.com', '987-65-4321', 60000.00);
```

2. Cree funciones de enmascaramiento personalizadas:

```
CREATE OR REPLACE FUNCTION public.mask_ssn(ssn TEXT)
```

```
    RETURNS TEXT AS $$
    BEGIN
        RETURN 'XXX-XX-' || RIGHT(ssn, 4);
    END;
    $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION public.mask_salary(salary NUMERIC, multiplier NUMERIC
 DEFAULT 0.0)
    RETURNS NUMERIC AS $$
    BEGIN
        RETURN salary * multiplier;
    END;
    $$ LANGUAGE plpgsql;
```

3. Cree varias políticas con diferentes niveles de enmascaramiento en función de los roles de los usuarios:

```
-- Create different roles
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE intern_role;

GRANT USAGE ON SCHEMA hr TO analyst_role, support_role, intern_role;
GRANT SELECT ON hr.employees TO analyst_role, support_role, intern_role;
-----------------------------------------------------------------------

-- Low-Weight Policy (Intern)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    JSON_BUILD_OBJECT('name', 'pgcolumnmask.mask_text(name, ''*'')',
                      'email', 'pgcolumnmask.mask_email(email)',
                      'ssn', 'pgcolumnmask.mask_text(ssn, ''*'')',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['intern_role'],
    10  -- Lowest weight
);

-----------------------------------------------------------------------
-- Medium-Weight Policy (Support)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_moderate',
    'hr.employees',
```

```
    JSON_BUILD_OBJECT('email', 'pgcolumnmask.mask_email(email, ''#'')',
                      'ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['support_role'],
    50   -- Medium weight
);

-------------------------------------------------------------------------
-- High-Weight Policy (Analyst)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_light',
    'hr.employees',
    JSON_BUILD_OBJECT('ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary, 0.9)')::JSONB,
    ARRAY['analyst_role'],
    100   -- Highest weight
);
```

4. Los siguientes ejemplos muestran cómo los diferentes usuarios ven los datos en función de la pertenencia a un rol y los pesos de las políticas.

```
-- Create users
CREATE USER sarah_intern;
GRANT intern_role TO sarah_intern;

CREATE USER lisa_support;
GRANT support_role TO lisa_support;

CREATE USER mike_analyst;
GRANT analyst_role TO mike_analyst;

CREATE USER ethan_support_intern;
GRANT support_role, intern_role TO ethan_support_intern;

CREATE USER john_analyst_intern;
GRANT analyst_role, intern_role TO john_analyst_intern;
```

Como pasante (enmascaramiento más estricto):

```
SET ROLE sarah_intern;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
```

```
----+------------+-----------------------+-------------+--------
  1 | ********   | XXXXXXXX@XXXXXXX.com  | ********** |   0.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | ********** |   0.00
```

Como usuario de apoyo (enmascaramiento moderado):

```
SET ROLE lisa_support;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | John Doe   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | Jane Smith | ##########@#######.com | XXX-XX-4321 |   0.00
```

Como analista (el enmascaramiento más ligero):

```
SET ROLE mike_analyst;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
  2 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
```

Como usuario ethan_support_intern, que es a la vez pasante y usuario de soporte:

```
SET ROLE ethan_support_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_moderate
-- id : unmasked because no masking policy appliable on ethan_support_intern
--            masks these columns
-- name : masked because of employee_mask_strict policy
-- email, ssn, salary : both employee_mask_strict and employee_mask_moderate mask
 these columns
--                      but employee_mask_moderate will be use because of higher
 weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | ########@#######.com   | XXX-XX-6789 |   0.00
```

```
  2 | ********** | ##########@#######.com | XXX-XX-4321 |   0.00
```

Como john_analyst_intern, que es a la vez pasante y analista:

```
SET ROLE john_analyst_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_light
-- id : unmasked because no masking policy appliable on john_analyst_intern
--            masks these columns
-- name, email : masked because of employee_mask_strict
-- ssn, salary : both employee_mask_strict and employee_mask_light mask these
 columns
--               but employee_mask_light will be use because of higher weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | XXX-XX-6789 | 45000.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | XXX-XX-4321 | 54000.00
```

Comprensión del comportamiento de enmascaramiento en las operaciones de DML

`pg_columnmask` se aplica de forma coherente en todas las operaciones de DML, como las instrucciones INSERT, UPDATE, DELETE y MERGE. Al ejecutar estas operaciones, Aurora PostgreSQL enmascara los datos de acuerdo con un principio fundamental: todos los datos leídos del almacenamiento se enmascaran de acuerdo con las políticas aplicables del usuario actual.

El enmascaramiento afecta a algunos de los siguientes componentes de la consulta, como:

- cláusulas WHERE
- condiciones de JOIN
- subconsultas
- cláusulas RETURNING

Todos estos componentes funcionan con valores enmascarados, no con los datos originales. Aunque los datos se escriben en el almacenamiento desenmascarado, los usuarios solo ven su vista enmascarada cuando los vuelven a leer.

Aurora PostgreSQL aplica todas las restricciones de la base de datos (NOT NULL, UNIQUE, CHECK, FOREIGN KEY) a los valores almacenados reales, no a los valores enmascarados. En ocasiones, esto puede crear aparentes incoherencias si las funciones de enmascaramiento no se diseñan cuidadosamente.

El enmascaramiento trabaja junto con los permisos por columnas:

- Los usuarios sin privilegios SELECT no pueden leer las columnas
- Los usuarios con privilegios SELECT ven los valores enmascarados de acuerdo con las políticas aplicables

Comprensión del comportamiento de enmascaramiento en las funciones de activación

Cuando se aplican políticas de `pg_columnmask` a las tablas, es importante entender cómo interactúa el enmascaramiento con las funciones de activación. Los activadores son funciones de la base de datos que se ejecutan automáticamente en respuesta a determinados eventos de una tabla, como las operaciones INSERT, UPDATE o DELETE.

De forma predeterminada, DDM aplica diferentes reglas de enmascaramiento según el tipo de activador:

Activadores de tablas

Las tablas de transición están desenmascaradas: las funciones activadoras de las tablas tienen acceso a los datos desenmascarados de sus tablas de transición en las versiones de filas antiguas y en las nuevas

Los propietarios de las tablas crean los activadores y son dueños de los datos, por lo que tienen acceso total para administrar sus tablas de manera eficaz

Consulta de activadores (EN LUGAR DE los activadores)

Las tablas de transición están enmascaradas: las funciones de activación de las vistas ven los datos enmascarados según los permisos del usuario actual

Los propietarios de las vistas pueden diferir de los propietarios de la tabla base y deben respetar las políticas de enmascaramiento de las tablas subyacentes

Dos parámetros de configuración por servidor controlan el comportamiento de los activadores con tablas enmascaradas. Estos solo se pueden configurar mediante `rds_superuser`:

- Restringir los activadores en las tablas enmascaradas: impide la ejecución de los activadores cuando un usuario enmascarado realiza operaciones de DML en tablas con las políticas de enmascaramiento aplicables.
- Restringir los activadores en las vistas con tablas enmascaradas: impide la ejecución de los activadores en las vistas cuando la definición de la vista incluye tablas con políticas de enmascaramiento aplicables al usuario actual.

Example de las diferencias entre la aplicación de la función a la tabla y la vista

El siguiente ejemplo crea una función de activación que imprime los valores de fila antiguos y nuevos y, a continuación, demuestra cómo la misma función se comporta de forma diferente cuando se adjunta a una tabla y no a una vista.

```
-- Create trigger function
CREATE OR REPLACE FUNCTION print_changes()
    RETURNS TRIGGER AS
    $$
        BEGIN
        RAISE NOTICE 'Old row: name=%, email=%, ssn=%, salary=%',
            OLD.name, OLD.email, OLD.ssn, OLD.salary;

        RAISE NOTICE 'New row: name=%, email=%, ssn=%, salary=%',
            NEW.name, NEW.email, NEW.ssn, NEW.salary;

        RETURN NEW;
        END;
    $$ LANGUAGE plpgsql;

-- Create trigger
CREATE TRIGGER print_changes_trigger
    BEFORE UPDATE ON hr.employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Grant update to analyst role
GRANT UPDATE ON hr.employees TO analyst_role;

-- Unmasked data must be seen inside trigger even for masked user for the OLD and NEW
-- row passed to trigger function
BEGIN;
SET ROLE mike_analyst;
```

```
UPDATE hr.employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)

ROLLBACK;


-- Triggers on views (which are supposed to see masked data for new/old row)
CREATE VIEW hr.view_over_employees AS SELECT * FROM hr.employees;
GRANT UPDATE, SELECT ON hr.view_over_employees TO analyst_role;

-- Create trigger for this view
CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON hr.view_over_employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Masked new and old rows should be passed to trigger if trigger is on view
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.view_over_employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
```

```
(2 rows)
ROLLBACK;
```

Se recomienda revisar el comportamiento de los activadores antes de implementarlos en las tablas enmascaradas. Los activadores de tablas tienen acceso a los datos desenmascarados de las tablas de transición, mientras que los activadores de visualización ven los datos enmascarados.

Example de cambio de nombre de la política de enmascaramiento

En el siguiente ejemplo se muestra cómo cambiar el nombre de las políticas existentes mediante el procedimiento `rename_masking_policy`.

```
-- Rename the strict policy
CALL pgcolumnmask.rename_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    'intern_protection_policy'
);

-- Verify the rename
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;

        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     50
 intern_protection_policy | {intern_role}  |     10
```

Example de modificar el peso de las políticas

En el siguiente ejemplo se muestra cómo modificar las ponderaciones de las políticas para cambiar su peso.

```
-- Change weight of moderate policy
CALL pgcolumnmask.alter_masking_policy(
    'employee_mask_moderate'::NAME,
    'hr.employees'::REGCLASS,
    NULL,    -- Keep existing masking expressions
    NULL,    -- Keep existing roles
```

```
    75       -- New weight
);

-- Verify the changes
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;
        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     75
 intern_protection_policy | {intern_role}  |     10
```

Example de limpieza

En el siguiente ejemplo se muestra cómo eliminar todas las políticas, tablas y usuarios.

```
-- Drop policies
CALL pgcolumnmask.drop_masking_policy(
    'intern_protection_policy',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_moderate',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_light',
    'hr.employees'
);

-- Drop table and functions
DROP VIEW IF EXISTS hr.view_over_employees;
DROP TABLE IF EXISTS hr.employees;
DROP SCHEMA IF EXISTS hr;
DROP FUNCTION IF EXISTS public.mask_ssn(text);
DROP FUNCTION IF EXISTS public.mask_salary(numeric, numeric);

-- Drop users
DROP USER sarah_intern, lisa_support, mike_analyst,
    ethan_support_intern, john_analyst_intern;
```

```
DROP ROLE intern_role, support_role, analyst_role;
```

Configuración del rol de administración de políticas de enmascaramiento

La extensión de enmascaramiento de columnas de PostgreSQL, `pg_columnmask`, le permite delegar la administración de las políticas de enmascaramiento a un rol específico, en lugar de requerir privilegios de propietario de tablas o `rds_superuser`. Esto proporciona un control más detallado sobre quién puede crear, modificar y eliminar las políticas de enmascaramiento.

Para configurar el rol que tendrá privilegios de administración de políticas de enmascaramiento, siga estos pasos:

1. Cree el rol de administrador de políticas: como un `rds_superuser`, cree un nuevo rol responsable de administrar las políticas de enmascaramiento:

   ```
   CREATE ROLE mask_admin NOLOGIN;
   ```

2. Configure el parámetro PostgreSQL: en el grupo de parámetros de clúster de base de datos personalizado, defina el parámetro de configuración del motor de `pgcolumnmask.policy_admin_rolname` con el nombre del rol que creó:

   ```
   pgcolumnmask.policy_admin_rolname = mask_admin
   ```

   Estos parámetros de configuración del motor se pueden establecer en un grupo de parámetros de un clúster de base de datos y no requiere el reinicio de la instancia. Para obtener más información sobre la actualización del parámetro, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

3. Otorgue el rol a los usuarios Como `rds_superuser`, conceda el rol de `mask_admin` a los usuarios que deberían administrar las políticas de enmascaramiento:

   ```
   CREATE USER alice LOGIN;
   CREATE USER bob LOGIN;
   GRANT mask_admin TO alice, bob;
   ```

   Además, asegúrese de que los usuarios tengan el privilegio de USO en los esquemas en los que administrarán las políticas de enmascaramiento:

   ```
   GRANT USAGE ON SCHEMA hr TO alice, bob;
   ```

Ahora, cuando los usuarios `alice` y `bob` se conecten a la base de datos, podrán utilizar las funciones de extensión estándar `pg_columnmask` para crear, modificar y eliminar políticas de enmascaramiento en todas las tablas de todos los esquemas en los que tengan privilegios `USAGE` en el esquema.

Prácticas recomendadas para la implementación segura de pg_columnmask

En la siguiente sección, se proporcionan las prácticas recomendadas de seguridad para la implementación de `pg_columnmask` en el entorno de Aurora PostgreSQL. Siga estas recomendaciones para:

- Establecimiento de una arquitectura de control de acceso basada en roles segura
- Desarrollo de funciones de enmascaramiento que eviten las vulnerabilidades de seguridad
- Comprensión y control del comportamiento de los activadores con datos enmascarados

Arquitectura de seguridad basada en roles

Defina una jerarquía de roles para implementar controles de acceso en la base de datos. Aurora PostgreSQL `pg_columnmask` amplía estos controles al proporcionar una capa adicional para el enmascaramiento de datos detallado dentro de esos roles.

Cree roles dedicados que se alineen con funciones organizativas en lugar de conceder permisos a usuarios individuales. Este enfoque proporciona una mejor auditoría y simplifica la administración de permisos a medida que la estructura organizativa evoluciona.

Example de creación de una jerarquía de roles organizativos

En el siguiente ejemplo, se crea una jerarquía de roles organizativos con roles específicos para distintas funciones y, a continuación, se asignan usuarios individuales a los roles correspondientes. En este ejemplo, primero se crean los roles organizativos (analyst_role, support_role) y, a continuación, se concede a los usuarios individuales la pertenencia a estos roles. Esta estructura le permite administrar los permisos por rol y no para cada usuario individual.

```
-- Create organizational role hierarchy
CREATE ROLE data_admin_role;
CREATE ROLE security_admin_role;
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE developer_role;
```

```
-- Specify security_admin_role as masking policy manager in the DB cluster parameter
-- group pgcolumnmask.policy_admin_rolname = 'security_admin_role'

-- Create specific users and assign to appropriate roles
CREATE USER security_manager;
CREATE USER data_analyst1, data_analyst2;
CREATE USER support_agent1, support_agent2;

GRANT security_admin_role TO security_manager;
GRANT analyst_role TO data_analyst1, data_analyst2;
GRANT support_role TO support_agent1, support_agent2;
```

Implemente el principio de privilegio mínimo mediante la concesión de solo los permisos mínimos necesarios para cada rol. Evite conceder permisos amplios que puedan aprovecharse si las credenciales se ven comprometidas.

```
-- Grant specific table permissions rather than schema-wide access
GRANT SELECT ON sensitive_data.customers TO analyst_role;
GRANT SELECT ON sensitive_data.transactions TO analyst_role;
-- Do not grant: GRANT ALL ON SCHEMA sensitive_data TO analyst_role;
```

Los administradores de políticas requieren privilegios de USAGE en los esquemas en los que administran las políticas de enmascaramiento. Conceda estos privilegios de forma selectiva, siguiendo el principio de privilegio mínimo. Realice revisiones periódicas de los permisos de acceso al esquema para asegurarse de que solo el personal autorizado mantenga las capacidades de administración de políticas.

La configuración de los parámetros del rol de administrador de políticas está restringida solo a los administradores de bases de datos. Este parámetro no se puede modificar en la base de datos o sesión, lo que impide que los usuarios sin privilegios invaliden las asignaciones de administradores de políticas. Esta restricción garantiza que el control de la política de enmascaramiento permanezca centralizado y seguro.

Asigne el rol de administrador de políticas a individuos específicos en lugar de a grupos.
Este enfoque específico garantiza el acceso selectivo a la administración de políticas de enmascaramiento, ya que los administradores de políticas tienen la capacidad de enmascarar todas las tablas de la base de datos.

Desarrollo seguro de la función de enmascaramiento

Desarrolle funciones de enmascaramiento utilizando una semántica de enlace temprana para garantizar un seguimiento adecuado de las dependencias y evitar vulnerabilidades de enlace tardío, como la modificación de la ruta de búsqueda durante el tiempo de ejecución. Se recomienda utilizar la sintaxis `BEGIN ATOMIC` de las funciones de SQL para permitir la validación en tiempo de compilación (es decir, la vinculación temprana) y la administración de las dependencias.

```
-- Example - Secure masking function with early binding
CREATE OR REPLACE FUNCTION secure_mask_ssn(input_ssn TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN input_ssn IS NULL THEN NULL
            WHEN length(input_ssn) < 4 THEN repeat('X', length(input_ssn))
            ELSE repeat('X', length(input_ssn) - 4) || right(input_ssn, 4)
        END;
    END;
```

Como alternativa, cree funciones que sean inmunes a los cambios en la ruta de búsqueda mediante un esquema explícito que califique todas las referencias a los objetos para garantizar un comportamiento coherente en las distintas sesiones de usuario.

```
-- Function immune to search path changes
CREATE OR REPLACE FUNCTION data_masking.secure_phone_mask(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    AS $$
    SELECT CASE
        WHEN phone_number IS NULL THEN NULL
        WHEN public.length(public.regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10
 THEN 'XXX-XXX-XXXX'
        ELSE public.regexp_replace(
            phone_number,
            '([0-9]{3})[0-9]{3}([0-9]{4})',
            public.concat('\1-XXX-\2')
        )
    END;
    $$;
```

Implemente la validación de entradas en las funciones de enmascaramiento para gestionar los casos de periferia y evitar comportamientos inesperados. Incluya siempre el manejo NULL y valide los formatos de entrada para garantizar un comportamiento de enmascaramiento coherente.

```
-- Robust masking function with comprehensive input validation
CREATE OR REPLACE FUNCTION secure_mask_phone(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN phone_number IS NULL THEN NULL
            WHEN length(trim(phone_number)) = 0 THEN phone_number
            WHEN length(regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10 THEN
 'XXX-XXX-XXXX'
            ELSE regexp_replace(phone_number, '([0-9]{3})[0-9]{3}([0-9]{4})', '\1-XXX-
\2')
        END;
    END;
```

Comportamiento de activadores de DML con pg_columnmask

Para los activadores de tablas, las tablas de transición se desenmascararán por completo. Para los activadores de visualización (IOT), las tablas de transición se enmascararán de acuerdo con los permisos de visualización del usuario actual.

Activadores de tablas con pg_columnmask

A los activadores se les pasa una tabla de transición que contiene la versión antigua y la nueva de las filas modificadas por la consulta de DML de activación. Según el momento en que se active el activador, Aurora PostgreSQL rellena las filas antiguas y nuevas. Por ejemplo, un activador `BEFORE INSERT` solo tiene versiones nuevas de las filas y versiones antiguas vacías porque no hay ninguna versión anterior a la que hacer referencia.

`pg_columnmask` no enmascara las tablas de transición dentro de los activadores de las tablas. Los activadores pueden usar columnas enmascaradas dentro de su cuerpo y ve los datos desenmascarados. El creador del activador debe asegurarse de cómo se ejecuta el activador para un usuario. El siguiente ejemplo funciona correctamente en este caso.

```
-- Example for table trigger uses masked column in its definition
-- Create a table and insert some rows
```

```
CREATE TABLE public.credit_card_table (
    name TEXT,
    credit_card_no VARCHAR(16),
    is_fraud BOOL
);

INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

-- Create a role which will see masked data and grant it privileges
CREATE ROLE intern_user;
GRANT SELECT, DELETE ON public.credit_card_table TO intern_user;

-- Trigger which will silenty skip delete of non fraudelent credit cards
CREATE OR REPLACE FUNCTION prevent_non_fraud_delete()
    RETURNS TRIGGER AS
    $$
    BEGIN
        IF OLD.is_fraud = false THEN
            RETURN NULL;
        END IF;
        RETURN OLD;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_non_fraud_delete
    BEFORE DELETE ON credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION prevent_non_fraud_delete();

CREATE OR REPLACE FUNCTION public.return_false()
    RETURNS BOOLEAN
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
      SELECT false;
    END;

-- A masking policy that masks both credit card number and is_fraud column.
-- If we apply masking inside trigger then prevent_non_fraud_delete trigger will
-- allow deleting more rows to masked user (even non fraud ones).
```

```
CALL pgcolumnmask.create_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    JSON_BUILD_OBJECT('credit_card_no', 'pgcolumnmask.mask_text(credit_card_no)',
                      'is_fraud', 'public.return_false()')::JSONB,
    ARRAY['intern_user']::NAME[],
    10::INT
);

-- Test trigger behaviour using intern_user
BEGIN;
SET ROLE intern_user;
-- credit card number & is_fraud is completely masked from intern_user
SELECT * FROM public.credit_card_table;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | XXXXXXXXXXXXXXXX | f
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(3 rows)

-- The delete trigger lets the intern user delete rows for Jane and Brad even though
-- intern_user sees their is_fraud = false, but the table trigger works with
 original
-- unmasked value
DELETE FROM public.credit_card_table RETURNING *;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(2 rows)

COMMIT;
```

El creador del activador filtra datos desenmascarados al usuario si no tiene cuidado con las instrucciones que utiliza en el cuerpo del activador. Por ejemplo, el uso de `RAISE NOTICE '%', masked_column;` imprime la columna al usuario actual.

```
-- Example showing table trigger leaking column value to current user
CREATE OR REPLACE FUNCTION leaky_trigger_func()
    RETURNS TRIGGER AS
    $$
```

```
    BEGIN
        RAISE NOTICE 'Old credit card number was: %', OLD.credit_card_no;
        RAISE NOTICE 'New credit card number is %', NEW.credit_card_no;
        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER leaky_trigger
    AFTER UPDATE ON public.credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION leaky_trigger_func();

-- Grant update on column is_fraud to auditor role
-- auditor will NOT HAVE PERMISSION TO READ DATA
CREATE ROLE auditor;
GRANT UPDATE (is_fraud) ON public.credit_card_table TO auditor;

-- Also add auditor role to existing masking policy on credit card table
CALL pgcolumnmask.alter_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    NULL::JSONB,
    ARRAY['intern_user', 'auditor']::NAME[],
    NULL::INT
);

-- Log in as auditor
-- [auditor]
-- Update will fail if trying to read data from the table
UPDATE public.credit_card_table
    SET is_fraud = true
    WHERE credit_card_no = '4532015112830366';
ERROR:  permission denied for table cc_table

-- [auditor]
-- But leaky update trigger will still print the entire row even though
-- current user does not have permission to select from public.credit_card_table
UPDATE public.credit_card_table SET is_fraud = true;
NOTICE:  Old credit_card_no was: 4532015112830366
NOTICE:  New credit_card_no is 4532015112830366
```

Se activa en las vistas con pg_columnmask (en lugar de activadores)

Los activadores solo se pueden crear en las vistas de PostgreSQL. Se utilizan para ejecutar instrucciones DML en vistas que no se pueden actualizar. Las tablas de tránsito siempre están enmascaradas en el interior en lugar de activarlas (IOT), ya que la vista y las tablas base utilizadas dentro de la consulta de vista pueden tener propietarios diferentes. En ese caso, las tablas base pueden tener algunas políticas de enmascaramiento aplicables al propietario de la vista y el propietario de la vista siempre debe ver los datos enmascarados de las tablas base dentro de sus activadores. Esto es diferente de los activadores de las tablas porque, en ese caso, el creador del activador y los datos contenidos en las tablas son propiedad del mismo usuario, lo que no es el caso aquí.

```
-- Create a view over credit card table
CREATE OR REPLACE VIEW public.credit_card_view
    AS
    SELECT * FROM public.credit_card_table;

-- Truncate credit card table and insert fresh data
TRUNCATE TABLE public.credit_card_table;
INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

CREATE OR REPLACE FUNCTION public.print_changes()
    RETURNS TRIGGER AS
    $$
    BEGIN
        RAISE NOTICE 'Old row: name=%, credit card number=%, is fraud=%',
            OLD.name, OLD.credit_card_no, OLD.is_fraud;

        RAISE NOTICE 'New row: name=%, credit card number=%, is fraud=%',
            NEW.name, NEW.credit_card_no, NEW.is_fraud;

    RETURN NEW;
   END;
   $$ LANGUAGE plpgsql;

CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON public.credit_card_view
    FOR EACH ROW
```

```
    EXECUTE FUNCTION public.print_changes();

GRANT SELECT, UPDATE ON public.credit_card_view TO auditor;

-- [auditor]
-- Login as auditor role
BEGIN;

-- Any data coming out from the table will be masked in instead of triggers
-- according to masking policies applicable to current user
UPDATE public.credit_card_view
    SET name = CONCAT(name, '_new_name')
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
        name         |  credit_card_no  | is_fraud
---------------------+------------------+----------
 John Doe_new_name   | XXXXXXXXXXXXXXXX | f
 Jane Smith_new_name | XXXXXXXXXXXXXXXX | f
 Brad Smith_new_name | XXXXXXXXXXXXXXXX | f

 -- Any new data going into the table using INSERT or UPDATE command will be
 unmasked
 UPDATE public.credit_card_view
    SET credit_card_no = '9876987698769876'
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith, credit card number=9876987698769876, is fraud=f
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | 9876987698769876 | f
 Jane Smith | 9876987698769876 | f
 Brad Smith | 9876987698769876 | f
```

```
COMMIT;
```

Los GUC por base de datos o usuario para controlar el comportamiento de los activadores

Dos parámetros de configuración controlan el comportamiento de ejecución de los activadores para los usuarios con políticas de enmascaramiento aplicables. Utilice estos parámetros para evitar que los activadores se ejecuten en tablas o vistas enmascaradas cuando se requieran restricciones de seguridad adicionales. Ambos parámetros están desactivados de forma predeterminada, lo que permite que los activadores se ejecuten con normalidad.

Primer GUC: desencadene la restricción de activación en tablas enmascaradas

Especificaciones:

- Nombre: `pgcolumnmask.restrict_dml_triggers_for_masked_users`
- Tipo:: `boolean`
- Predeterminado: `false` (se permite la ejecución de los activadores)

Impide la ejecución del activador en tablas enmascaradas para los usuarios enmascarados cuando se establece en TRUE. `pg_columnmask` corrige el error.

Segundo GUC: desencadene la restricción de activación en vistas con tablas enmascaradas

Especificaciones:

- Nombre: `pgcolumnmask.restrict_iot_triggers_for_masked_users`
- Tipo:: `boolean`
- Predeterminado: `false` (se permite la ejecución de los activadores)

Impide la ejecución del activador en vistas que incluyen tablas enmascaradas en su definición para los usuarios enmascarados cuando se establece en TRUE.

Estos parámetros funcionan de forma independiente y se pueden configurar como los parámetros de configuración de una base de datos estándar.

Escenarios de movimiento de datos pg_columnmask de Aurora PostgreSQL

El comportamiento `pg_columnmask` varía según las distintas operaciones de movimiento de datos en función de si la operación se produce en la capa de almacenamiento, lógica o aplicación. Las

operaciones por almacenamiento (como la clonación) se comportan de manera diferente a las operaciones lógicas (como pg_dump) y las operaciones por aplicación (como las consultas de FDW). En esta sección se describe el comportamiento de enmascaramiento en escenarios comunes, como la replicación, las copias de seguridad, las exportaciones y las migraciones, y se explican las implicaciones de seguridad de cada uno de ellos.

Temas

- Base de datos global de Aurora y réplicas de lectura
- Restauración de copias e instantáneas de bases de datos
- Replicación lógica
- Implementaciones azul/verde
- Transmisiones zero-ETL y CDC
- AWS Database Migration Service
- Exportaciones de datos
- Vistas y vistas materializadas
- Volcado y restauración de datos
- Contenedor de datos externos

Base de datos global de Aurora y réplicas de lectura

Las políticas de pg_columnmask de Aurora se almacenan en tablas del sistema de bases de datos dentro del volumen del clúster. Todas las réplicas acceden a las mismas políticas y devuelven resultados enmascarados de forma coherente. Para las implementaciones de Aurora Global Database, las políticas de pg_columnmask se replican en Regiones de AWS secundarias junto con otras tablas del sistema de bases de datos, lo que garantiza una protección de datos coherente en todas las regiones. Durante los escenarios de conmutación por error, todas las políticas de pg_columnmask permanecen intactas y funcionales.

Restauración de copias e instantáneas de bases de datos

Las operaciones de restauración de instantáneas y clonación rápida de Aurora conservan todas las políticas de pg_columnmask, roles y configuraciones como parte de las tablas del sistema de bases de datos. La base de datos clonada o restaurada hereda todas las políticas existentes del clúster de origen. Tras la clonación o la restauración, cada clúster de base de datos mantiene políticas de pg_columnmask independientes.

Replicación lógica

Durante la sincronización inicial, la replicación lógica utiliza operaciones estándar de SQL COPY y las políticas de `pg_columnmask` se aplican en función de los permisos del usuario de la replicación. Durante el CDC continuo (captura de datos de cambio), no se aplican políticas de enmascaramiento y los datos desenmascarados se replican a través de los registros de WAL. Los usuarios con privilegios de `pg_create_subscription` pueden filtrar datos desenmascarados configurando la replicación en un sistema que controlen.

Implementaciones azul/verde

Durante la restauración de las instantáneas, las políticas de `pg_columnmask` se incluyen automáticamente. El entorno verde comienza con una copia idéntica de todas las políticas del entorno azul. Durante la replicación, de azul a verde, los datos no se enmascaran. Los cambios posteriores en la política de enmascaramiento (comandos DDL) en el clúster azul no se replican en el clúster verde e invalidan las implementaciones azul/verde de RDS.

Transmisiones zero-ETL y CDC

La replicación de datos no se ve afectada por las políticas de `pg_columnmask`. Zero-ETL admite la replicación de DDL, pero no replica `pg_columnmask` ni las políticas de RLS. No se aplican políticas de enmascaramiento a los datos replicados en Zero-ETL.

AWS Database Migration Service

La sincronización de datos inicial se enmascara o desenmascara en función del usuario seleccionado para la tarea de DMS. Los datos de CDC siempre están desenmascarados. Aunque las políticas de RLS internas relacionadas con `pg_columnmask` se pueden migrar, no funcionarán en destinos non-pg_columnmask-enabled.

Exportaciones de datos

`pg_columnmask` trata las exportaciones como cualquier otra operación de consulta: el enmascaramiento se aplica en función de los permisos del usuario ejecutor. Esto se aplica a los comandos de SQL como COPY, SELECT INTO, CREATE TABLE AS y a la funcionalidad de exportación de S3 de Aurora PostgreSQL.

> **Note**
>
> Cuando los usuarios enmascarados exportan datos, los archivos resultantes contienen valores enmascarados que, al restaurarse, pueden infringir las restricciones de la base de datos.

Vistas y vistas materializadas

Tenga en cuenta las siguientes consideraciones al utilizar las vistas:

- Vistas normales: utilice siempre la semántica de `INVOKER`. Las políticas de enmascaramiento del usuario actual se aplican al consultar la vista, independientemente de quién la haya creado.
- Vistas materializadas: cuando se actualizan, se aplican las políticas de enmascaramiento del propietario de la vista materializada, no las políticas del usuario que realiza la actualización. Si el propietario tiene políticas de enmascaramiento, la vista materializada siempre contiene datos enmascarados.

Volcado y restauración de datos

`pg_dump` funciona como un usuario normal de la base de datos y aplica políticas de enmascaramiento en función de los permisos del usuario que se conecta. Si un usuario enmascarado realiza un volcado, el archivo de copia de seguridad contiene datos enmascarados. Las políticas de `pg_columnmask` se incluyen en el volcado como parte del esquema de la base de datos. La restauración correcta requiere que todos los roles a los que se hace referencia estén en la base de datos de destino y que el destino tenga instalada la extensión `pg_columnmask`.

> **Note**
>
> A partir de PostgreSQL 18, `pg_dump` admite la opción `–no-policies` que excluye la seguridad por fila (RLS) y políticas de enmascaramiento de `pg_columnmask` de los volcados de bases de datos. Para obtener más información, consulte pg_dump.

Contenedor de datos externos

Cuando se utilizan contenedores de datos externos, las políticas de enmascaramiento en las tablas remotas se aplican en función de los permisos del usuario asignado en el servidor de origen,

no de los permisos del usuario local que consulta, y aunque puede acceder a los datos remotos enmascarados mediante FDW, no puede crear políticas de DDM o RLS directamente en las tablas externas de la base de datos local.

Actualización de aplicaciones para la conexión a los clústeres de base de datos de PostgreSQL de Aurora con los nuevos certificados SSL/TLS

El 13 de enero de 2023, Amazon RDS publicó nuevos certificados de entidades de certificación (CA) para la conexión a sus clústeres de base de datos de Aurora mediante la capa de sockets seguros o la seguridad de la capa de transporte (SSL/TLS). Después, puede encontrar la información sobre la actualización de sus aplicaciones para utilizar los nuevos certificados.

Este tema puede ayudarle a determinar las aplicaciones de cualquier cliente utilizan SSL/TLS para conectarse a sus clústeres de base de datos. Si lo hacen, puede comprobar de manera adicional si esas aplicaciones precisan una verificación de certificados para conectarse.

> **Note**
>
> Algunas aplicaciones están configuradas para conectarse a los clústeres de base de datos de PostgreSQL de Aurora solo si pueden verificar con éxito el certificado del servidor. Para esas aplicaciones, debe actualizar los almacenes de confianza de la aplicación de su cliente para incluir los nuevos certificados de CA.

Después de que actualice sus certificados de CA en los almacenes de confianza de la aplicación de su cliente, puede rotar los certificados en sus clústeres de base de datos. Recomendamos encarecidamente probar estos procedimientos en un entorno de desarrollo o ensayo antes de implementarlos en sus entornos de producción.

Para obtener más información acerca de la rotación de certificados, consulte Rotar certificados SSL/TLS. Para obtener más información acerca de cómo descargar certificados, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos. Para obtener información sobre el uso de SSL/TLS con los clústeres de base de datos de PostgreSQL, consulte Protección de los datos de Aurora PostgreSQL con SSL/TLS.

Temas

- Determinación de si las aplicaciones se conectan a sus clústeres de base de datos de PostgreSQL de Aurora mediante SSL
- Determinación de si un cliente necesita una verificación de certificados para conectarse
- Actualización del almacén de confianza de su aplicación
- Uso de conexiones SSL/TLS para diferentes tipos de aplicaciones

Determinación de si las aplicaciones se conectan a sus clústeres de base de datos de PostgreSQL de Aurora mediante SSL

Compruebe la configuración del clúster de base de datos para obtener el valor del parámetro de `rds.force_ssl`. De forma predeterminada, el parámetro `rds.force_ssl` está definido como `0` (desactivado). Si el parámetro `rds.force_ssl` está configurado como `1` (activado), se precisa que los clientes utilicen SSL/TLS para las conexiones. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.

Si `rds.force_ssl` no está configurado como `1` (activado), consulte la vista `pg_stat_ssl` para comprobar las conexiones que utilizan SSL. Por ejemplo, la siguiente consulta devuelve solo las conexiones SSL y la información acerca de los clientes que utilizan SSL.

```
select datname, usename, ssl, client_addr from pg_stat_ssl inner join pg_stat_activity
 on pg_stat_ssl.pid = pg_stat_activity.pid where ssl is true and usename<>'rdsadmin';
```

Solo las filas que utilizan conexiones SSL/TLS se muestran con información sobre la conexión. A continuación, se muestra un ejemplo del resultado.

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

La anterior consulta solo muestra las conexiones actuales en el momento de la consulta. La ausencia de resultados no indica que no haya ninguna aplicación utilizando conexiones SSL. Se pueden establecer otras conexiones SSL en un momento diferente.

Determinación de si un cliente necesita una verificación de certificados para conectarse

Cuando un cliente, como psql o JDBC, está configurado con soporte de SSL, primero el cliente intenta conectarse a la base de datos con SSL de manera predeterminada. SI el cliente no puede conectarse con SSL, vuelve a la conexión sin SSL. El modo `sslmode` predeterminado utilizado para clientes con libpq (como psql) y JDBC está establecido en `prefer`. El certificado del servidor se verifica solo cuando se proporciona `sslrootcert` con `sslmode` configurado como `verify-ca` o `verify-full`. Se lanza un error si el certificado no es válido.

Utilice `PGSSLROOTCERT` para verificar el certificado con la variable de entorno `PGSSLMODE`, con `PGSSLMODE` establecido como `verify-ca` o `verify-full`.

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U primaryuser -d postgres
```

Utilice el argumento `sslrootcert` para verificar el certificado con `sslmode` en el formato de la cadena de conexión, con `sslmode` establecido como `verify-ca` o `verify-full`.

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=primaryuser dbname=postgres"
```

Por ejemplo, en el caso anterior, si utiliza un certificado raíz no válido, observa un error similar a lo siguiente en su cliente.

```
psql: SSL error: certificate verify failed
```

Actualización del almacén de confianza de su aplicación

Para obtener información sobre la actualización del almacén de confianza para las aplicaciones de PostgreSQL, consulte Conexiones TCP/IP seguras con SSL en la documentación de PostgreSQL.

> **Note**
>
> Cuando actualice el almacén de confianza, puede retener certificados antiguos además de añadir los nuevos certificados.

Actualización del almacén de confianza de su aplicación para JDBC

Puede actualizar el almacén de confianza para las aplicaciones que utilizan JDBC para las conexiones SSL/TLS.

Para obtener información sobre la descarga del certificado raíz, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

Para obtener secuencias de comandos de ejemplo que importan certificados, consulte Script de muestra para la importación de certificados en su almacén de confianza.

Uso de conexiones SSL/TLS para diferentes tipos de aplicaciones

Lo siguiente proporciona información acerca del uso de conexiones SSL/TLS para diferentes tipo de aplicaciones:

- psql

  El cliente se ha invocado desde la línea de comandos especificando las opciones como una cadena de conexión o como variables del entorno. Para las conexiones SSL/TLS, las opciones relevantes son `sslmode` (variable de entorno `PGSSLMODE`), `sslrootcert` (variable de entorno `PGSSLROOTCERT`).

  Para obtener la lista completa de opciones, consulte Palabras de clave del parámetro en la documentación de PostgreSQL. Para obtener la lista completa de variables de entorno, consulte Variables de entorno en la documentación de PostgreSQL.

- pgAdmin

  Este cliente basado en el navegador es una interfaz más intuitiva para conectarse a la base de datos de PostgreSQL.

  Para obtener información sobre la configuración de las conexiones, consulte la documentación de pgAdmin.

- JDBC

  JDBC habilita las conexiones de base de datos con aplicaciones de Java.

  Para obtener información general sobre la conexión a la base de datos de PostgreSQL con JDBC, consulte Conexión a la base de datos en la documentación de PostgreSQL. Para

obtener información sobre la conexión con SSL/TLS, consulte Configuración del cliente en la documentación de PostgreSQL.

- Python

  Una biblioteca de Python popular para la conexión a las bases de datos de PostgreSQL es `psycopg2`.

  Para obtener información acerca del uso de `psycopg2`, consulte la documentación de psycopg2. Para obtener un breve tutorial sobre cómo conectarse a una base de datos de PostgreSQL, consulte Tutorial de psycopg2. Puede encontrar información acerca de las opciones que acepta del comando de conexión en El contenido del módulo psycopg2.

> **Important**
>
> Después de que haya determinado que sus conexiones de base de datos utilizan SSL/TLS y haya actualizado el almacén de confianza de su aplicación, puede actualizar su base de datos para que utilice los certificados de rds-ca-rsa2048-g1. Para obtener instrucciones, consulte el paso 3 en Actualización del certificado de entidad de certificación modificando la instancia de base de datos.

Uso de la autenticación Kerberos con Aurora PostgreSQL

Puede usar Kerberos para autenticar a los usuarios cuando se conecten a su clúster de base de datos en la que se ejecuta PostgreSQL. Para ello, configure el clúster de base de datos para utilizar AWS Directory Service for Microsoft Active Directory para la autenticación Kerberos. AWS Directory Service for Microsoft Active Directory también se denomina AWS Managed Microsoft AD. Es una función disponible con Directory Service. Para obtener más información, consulte ¿Qué esDirectory Service? en la Guía de administración de AWS Directory Service.

Para empezar, cree un directorio de AWS Managed Microsoft AD para almacenar las credenciales de usuario. A continuación, proporcione a su clúster de base de datos de PostgreSQL el dominio de Active Directory y otra información. Cuando los usuarios se autentican con la de clúster de base de datos de PostgreSQL, las solicitudes de autenticación se reenvían al directorio AWS Managed Microsoft AD.

Mantener todas las credenciales en el mismo directorio puede ahorrarle tiempo y esfuerzo. Tiene un lugar centralizado para almacenar y administrar credenciales para varios clústeres de bases de datos. El uso de un directorio también puede mejorar su perfil de seguridad general.

Además, puede acceder a las credenciales desde su propio Microsoft Active Directory en las instalaciones. Para ello, cree una relación de dominio de confianza para que el directorio de AWS Managed Microsoft AD confíe en su Microsoft Active Directory en las instalaciones. De esta manera, los usuarios pueden acceder a las instancias de los clústeres con la misma experiencia de inicio de sesión único (SSO) de Windows que cuando acceden a cargas de trabajo en las instalaciones.

Una base de datos puede usar la autenticación de Kerberos, de AWS Identity and Access Management (IAM), o ambas. Sin embargo, dado que la autenticación de Kerberos y de IAM proporcionan diferentes métodos de autenticación, un usuario específico puede iniciar sesión en una base de datos mediante solo uno u otro método de autenticación, pero no ambos. Para obtener más información acerca de la autenticación IAM, consulte Autenticación de bases de datos de IAM .

> **Note**
>
> RDS para PostgreSQL no admite la autenticación Kerberos para grupos de Active Directory.

Temas

- Disponibilidad en regiones y versiones
- Información general de la autenticación Kerberos para clústeres de base de datos de PostgreSQL
- Configuración de autenticación Kerberos para clústeres de base de datos de PostgreSQL
- Administración de un clúster de base de datos de Aurora PostgreSQL en un dominio de Active Directory
- Conexión a PostgreSQL con autenticación Kerberos
- Uso de grupos de seguridad de AD para el control de acceso de Aurora PostgreSQL

Disponibilidad en regiones y versiones

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos y entre Regiones de AWS. Para obtener más información sobre la disponibilidad en versiones y regiones de Aurora PostgreSQL con autenticación Kerberos, consulte Autenticación Kerberos con con Aurora PostgreSQL.

Información general de la autenticación Kerberos para clústeres de base de datos de PostgreSQL

Para configurar la autenticación Kerberos para un clúster de base de datos de PostgreSQL, complete los siguientes pasos generales, que se describen con más detalle más adelante:

1. Utilice AWS Managed Microsoft AD para crear un directorio de AWS Managed Microsoft AD. Puede utilizar la Consola de administración de AWS, la AWS CLI o la API de Directory Service para crear el directorio. Asegúrese de abrir los puertos de salida relevantes en el grupo de seguridad del directorio para que el directorio pueda comunicarse con el clúster.
2. Cree un rol que proporcione a Amazon Aurora acceso para realizar llamadas a su directorio de AWS Managed Microsoft AD. Para ello, cree un rol de AWS Identity and Access Management (IAM) que utilice la política administrada de IAM `AmazonRDSDirectoryServiceAccess`.

   Para que el rol de IAM permita el acceso, el punto de enlace AWS Security Token Service (AWS STS) debe activarse en la región de AWS correcta para su cuenta de AWS. Los puntos de conexión de AWS STS están activos de forma predeterminada en todas las Regiones de AWS y puede usarlos sin ninguna acción posterior. Para obtener más información, consulte Activación y desactivación de AWS STS en una región de AWS en la Guía del usuario de IAM.
3. Cree y configure usuarios en el directorio de AWS Managed Microsoft AD usando las herramientas de Microsoft Active Directory. Para obtener más información sobre la creación de usuarios en su Active Directory, consulte Administrar usuarios y grupos en AWS Managed Microsoft AD en la Guía de administración de Directory Service.
4. Si piensa localizar el directorio y la instancia de base de datos en diferentes cuentas de AWS o nubes virtuales privadas (VPC), configure la interconexión de VPC. Para obtener más información, consulte ¿Qué es una interconexión de VPC? en la Amazon VPC Peering Guide.
5. Cree o modifique un clúster de base de datos de PostgreSQL desde la consola, la CLI o la API de RDS utilizando uno de los siguientes métodos:
   - Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él
   - Modificación de un clúster de base de datos de Amazon Aurora
   - Restauración de una instantánea de clúster de base de datos
   - Restauración de un clúster de base de dato a un momento indicado

   Puede localizar el clúster en la misma Amazon Virtual Private Cloud (VPC) que el directorio o en una VPC o cuenta de AWS diferente. Cuando cree o modifique el clúster de base de datos de PostgreSQL, haga lo siguiente:

- Proporcione el identificador de dominio (identificador `d-*`) que se generó cuando creó el directorio.
- Proporcione el nombre del rol de IAM que ha creado.
- Asegúrese de que el grupo de seguridad de la instancia de base de datos pueda recibir tráfico de entrada del grupo de seguridad del directorio.

6. Use las credenciales de usuario maestro de RDS para conectarse al clúster de base de datos de PostgreSQL. Cree el usuario en PostgreSQL para que sea identificado externamente. Los usuarios identificados externamente pueden iniciar sesión en el clúster de base de datos de PostgreSQL utilizando la autenticación Kerberos.

Configuración de autenticación Kerberos para clústeres de base de datos de PostgreSQL

Utilice AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) para configurar la autenticación Kerberos para un clúster de base de datos de PostgreSQL. Para configurar la autenticación Kerberos, siga los pasos que se indican a continuación:

Temas

- Paso 1: crear un directorio con AWS Managed Microsoft AD
- Paso 2: (opcional) crear una relación de confianza entre su Active Directory en las instalaciones y Directory Service
- Paso 3: crear un rol de IAM para que Amazon Aurora acceda a Directory Service
- Paso 4: crear y configurar usuarios
- Paso 5: habilitar el tráfico entre VPC entre el directorio y la instancia de base de datos
- Paso 6: crear o modificar un clúster de base de datos de PostgreSQL
- Paso 7: crear usuarios de PostgreSQL para las entidades principales de Kerberos
- Paso 8: configurar un cliente de PostgreSQL

Paso 1: crear un directorio con AWS Managed Microsoft AD

Directory Service crea un directorio de Active Directory completamente administrado en la nube de AWS. Al crear un directorio de AWS Managed Microsoft AD, Directory Service crea dos controladores de dominio y servidores DNS para usted. Los servidores de directorios se crean en

diferentes subredes de una VPC. Esta redundancia ayuda a garantizar que su directorio permanezca accesible incluso si ocurre un error.

Cuando se crea un directorio de AWS Managed Microsoft AD, AWS Directory Service realiza las siguientes tareas en su nombre:

- Configura un Active Directory dentro de la VPC.
- Crea una cuenta de administrador del directorio con el nombre de usuario `Admin` y la contraseña especificada. Esta cuenta le permite administrar el directorio.

  > **Important**
  >
  > Asegúrese de guardar esta contraseña. Directory Service no almacena esta contraseña y no se puede recuperar ni restablecer.

- Crea un grupo de seguridad para los controladores del directorio. El grupo de seguridad debe permitir la comunicación con el clúster de base de datos de PostgreSQL.

Al lanzar AWS Directory Service for Microsoft Active Directory, AWS crea una unidad organizativa (OU) que contiene todos los objetos del directorio. Esta unidad organizativa, que tiene el nombre de NetBIOS que introdujo al crear el directorio, se encuentra en la raíz del dominio. La raíz del dominio es propiedad de AWS, que también se encarga de su administración.

La cuenta `Admin` que se creó con el directorio de AWS Managed Microsoft AD dispone de permisos para realizar las actividades administrativas más habituales para la unidad organizativa:

- Crear, actualizar o eliminar usuarios
- Añadir recursos a su dominio, como servidores de archivos o de impresión y, a continuación, asignar permisos para esos recursos a usuarios dentro de la unidad organizativa
- Crear unidades organizativas y contenedores adicionales
- Delegar autoridad
- Restaurar objetos eliminados de la papelera de reciclaje de Active Directory
- Ejecute módulos de Active Directory y Domain Name Service (DNS) para Windows Powershell en el servicio web de Active Directory

La cuenta `Admin` también tiene derechos para realizar las siguientes actividades en todo el dominio:

- Administrar configuraciones DNS (agregar, quitar o actualizar registros, zonas y programas de envío).
- Ver logs de eventos DNS
- Ver logs de eventos de seguridad

Para crear un directorio con AWS Managed Microsoft AD

1. En el panel de navegación de la consola de Directory Service, elija Directories (Directorios) y, a continuación, elija Set up Directory (Configurar directorio).
2. Elija AWS Managed Microsoft AD. AWS Managed Microsoft AD es la única opción que se admite actualmente para usar con Amazon Aurora.
3. Elija Next (Siguiente).
4. En la página Enter directory information (Especifique la información del directorio), facilite la siguiente información:

   Edición

   Elija la edición que se adapte a sus necesidades.

   Nombre de DNS del directorio

   El nombre completo del directorio, como por ejemplo **`corp.example.com`**.

   Nombre NetBIOS del directorio

   Un nombre abreviado del directorio opcional, como COR `CORP`.

   Descripción del directorio

   Descripción opcional del directorio.

   Contraseña de administrador

   Contraseña del administrador del directorio. Al crear el directorio, se crea también una cuenta de administrador con el nombre de usuario `Admin` y esta contraseña.

   La contraseña del administrador del directorio no puede contener la palabra "admin".
   La contraseña distingue entre mayúsculas y minúsculas y debe tener un mínimo de 864 caracteres y un máximo de 64. También debe contener al menos un carácter de tres de las siguientes categorías:

   - Letras minúsculas (a–z)

- Letras mayúsculas (A–Z)
- Números (0–9)
- Caracteres no alfanuméricos (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password

Vuelva a escribir la contraseña de administrador.

> **Important**
>
> Asegúrese de guardar esta contraseña. Directory Service no almacena esta contraseña y no se puede recuperar ni restablecer.

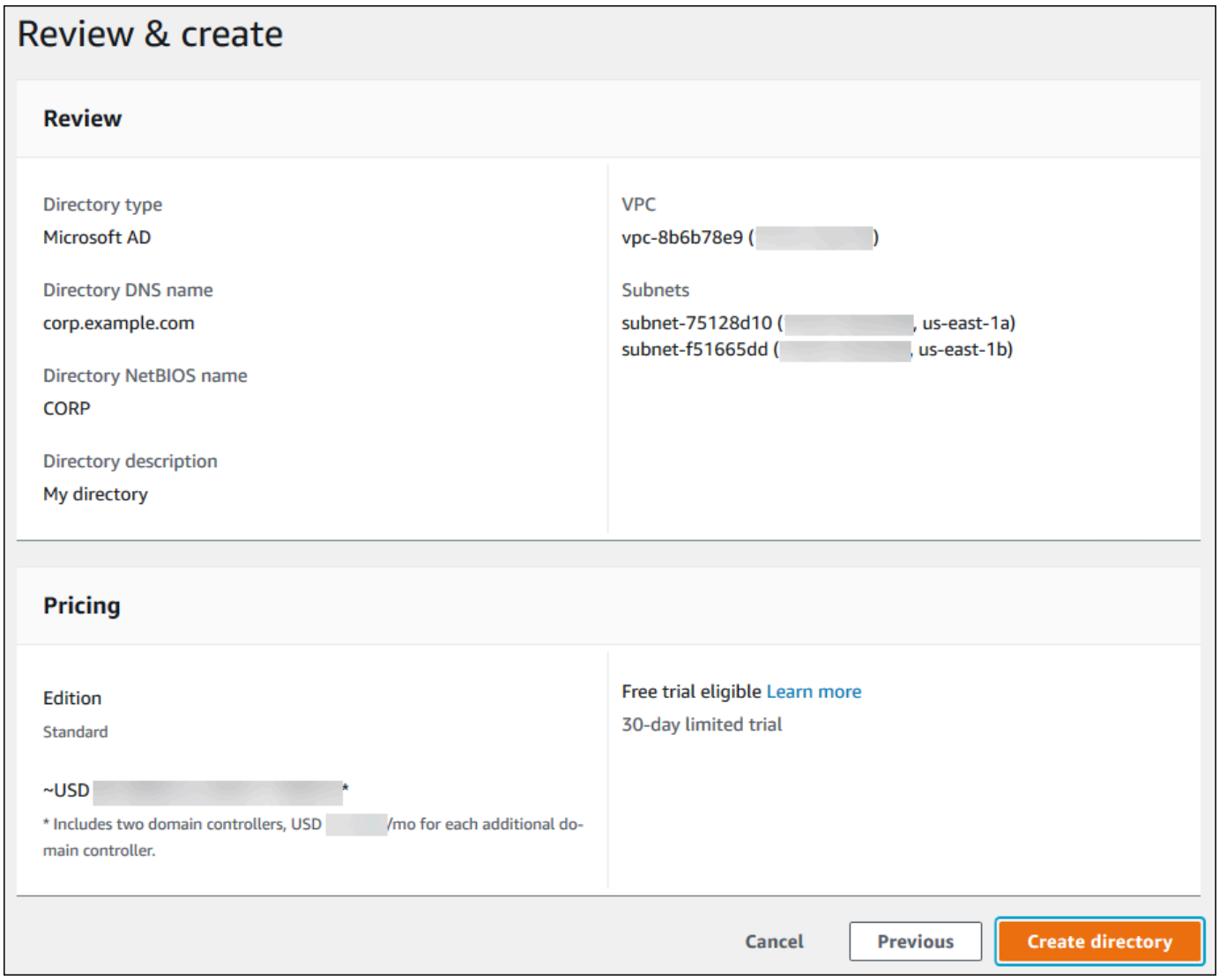
5. Elija Next (Siguiente).
6. En la página Choose VPC and subnets (Elegir la VPC y las subredes), proporcione la información siguiente:

   VPC

   Elija la VPC del directorio. Puede crear el clúster de base de datos de PostgreSQL en esta misma VPC o en una VPC diferente.

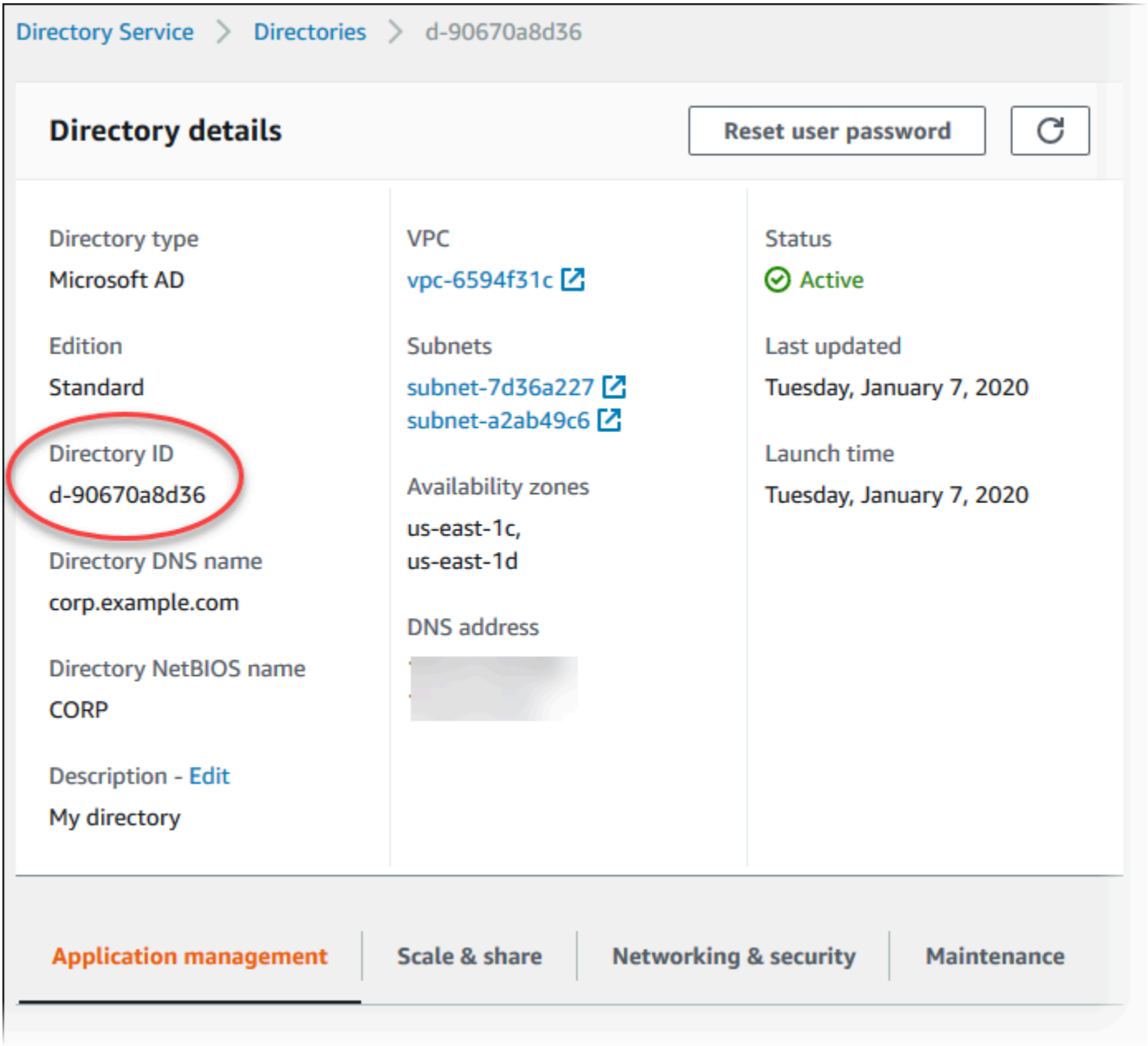
   Subredes

   Elija las subredes de los servidores del directorio. Las dos subredes deben estar en diferentes zonas de disponibilidad.
7. Elija Next (Siguiente).
8. Revise la información del directorio. Si es necesario realizar algún cambio, seleccione Previous (Anterior) y realizar los cambios. Cuando la información sea correcta, seleccione Create directory (Crear directorio).



La creación del directorio tarda varios minutos. Cuando se haya creado correctamente, el valor de Status (Estado) cambiará a Active (Activo).

Para ver información acerca de su directorio, seleccione el ID del directorio en la lista de directorios. Anote el valor de Directory ID (ID de directorio). Necesita este valor cuando cree o modifique su instancia de base de datos de PostgreSQL.



Paso 2: (opcional) crear una relación de confianza entre su Active Directory en las instalaciones y Directory Service

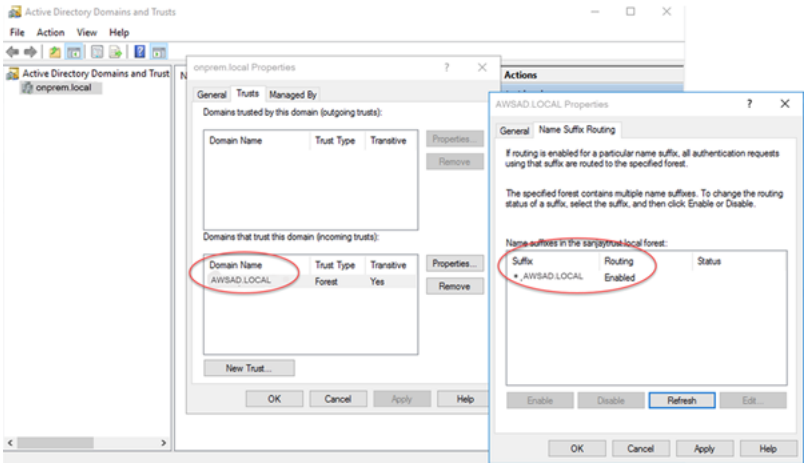
Si no planea usar su propio Microsoft Active Directory local, vaya a Paso 3: crear un rol de IAM para que Amazon Aurora acceda a Directory Service.

Para obtener la autenticación Kerberos mediante Active Directory en las instalaciones, debe crear una relación de dominio de confianza entre Microsoft Active Directory en las instalaciones y el directorio AWS Managed Microsoft AD (creado en Paso 1: crear un directorio con AWS Managed Microsoft AD). La confianza puede ser unidireccional, donde el directorio AWS Managed Microsoft

AD confía en Microsoft Active Directory local. La confianza también puede ser bidireccional, donde ambos Active Directories confían entre sí. Para obtener más información acerca de la configuración de relaciones de confianza con Directory Service, consulte Cuándo crear una relación de confianza en la guía de administración de AWS Directory Service.

> **Note**
>
> Si utiliza Microsoft Active Directory en las instalaciones:
>
> - Los clientes de Windows deben conectarse con el nombre de dominio del Directory Service en el punto de conexión en lugar de con rds.amazonaws.com. Para obtener más información, consulte Conexión a PostgreSQL con autenticación Kerberos.
> - Los clientes de Windows no pueden conectarse con utilizando punto de conexión personalizados de Aurora. Para obtener más información, consulte Conexiones de puntos de conexión de Amazon Aurora.
> - Para bases de datos globales:
>   - Los clientes de Windows solo pueden conectarse mediante los puntos de enlace de la instancia o los del clúster en la Región de AWS principal de la base de datos global.
>   - Los clientes de Windows no pueden conectarse mediante los puntos de enlace del clúster en Regiones de AWS secundarias.

Asegúrese de que el nombre de dominio local de Microsoft Active Directory incluya un enrutamiento de sufijo DNS que corresponda a la relación de confianza recién creada. En la siguiente captura de pantalla, se muestra un ejemplo.



Paso 3: crear un rol de IAM para que Amazon Aurora acceda a Directory Service

Para que Amazon Aurora llame a Directory Service en su nombre, su cuenta de AWS se precisa un rol de IAM que utilice la política de IAM administrada `AmazonRDSDirectoryServiceAccess`. Este rol permite que Amazon Aurora llame a Directory Service. (Tenga en cuenta que este rol de IAM es diferente para acceder a Directory Service al rol de IAM utilizado para Autenticación de bases de datos de IAM .)

Cuando se crea una instancia de base de datos con la Consola de administración de AWS y la cuenta de usuario de la consola tiene el permiso `iam:CreateRole`, la consola crea automáticamente el rol de IAM necesario. En este caso, el nombre del rol es `rds-directoryservice-kerberos-access-role`. De no ser así, debe crear el rol de IAM manualmente. Cuando cree este rol de IAM, elija `Directory Service` y asocie la política administrada de AWS `AmazonRDSDirectoryServiceAccess` a este.

A fin de obtener más información acerca de la creación de roles de IAM para un servicio, consulte Creación de un rol para delegar permisos a un servicio de AWS en la guía del usuario de IAM.

> **Note**
>
> El rol de IAM utilizado para la autenticación de Windows para RDS para Microsoft SQL Server no puede ser utilizado por Amazon Aurora.

Como alternativa al uso de la política administrada de `AmazonRDSDirectoryServiceAccess`, puede crear políticas con los permisos necesarios. En este caso, el rol de IAM debe tener la siguiente política de confianza de IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
```

```
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

El rol debe también tener la siguiente política de rol de IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Para suscribirse a Regiones de AWS, utilice las entidades principales de servicios específicos de la región en las políticas de confianza de roles de IAM. Cuando cree una política de confianza para servicios en estas regiones, especifique el código de región en la entidad principal de servicio.

En el siguiente ejemplo se muestra una política de confianza que incluye entidades principales de servicio específicas de la región:

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Reemplace REGION-CODE por el código de la región específica. Por ejemplo, utilice las siguientes entidades principales de servicio para la región Asia-Pacífico (Melbourne):

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

Paso 4: crear y configurar usuarios

Puede crear usuarios usando la herramienta Usuarios y equipos de Active Directory. Es una de las herramientas Active Directory Domain Services y Active Directory Lightweight Directory Services. Para obtener más información, consulte Agregar usuarios y equipos al dominio de Active Directory en la documentación de Microsoft. En este caso, los usuarios son individuos u otras entidades, como sus equipos, que forman parte del dominio y cuyas identidades se mantienen en el directorio.

Para crear usuarios en un directorio de Directory Service, debe estar conectado a una instancia de Amazon EC2 con Windows que sea miembro del directorio de Directory Service. Al mismo tiempo, debe iniciar sesión como usuario con privilegios para crear usuarios. Para obtener más información, consulte Crear un usuario en la Guía de administración de AWS Directory Service.

Paso 5: habilitar el tráfico entre VPC entre el directorio y la instancia de base de datos

Si prevé ubicar el directorio y el clúster de base de datos en la misma VPC, omita este paso y continúe con Paso 6: crear o modificar un clúster de base de datos de PostgreSQL.

Si tiene previsto ubicar el directorio y la instancia de base de datos en diferentes VPC, configure el tráfico entre VPC mediante la interconexión de VPC o AWSTransit Gateway.

El siguiente procedimiento permite el tráfico entre VPC mediante la interconexión de VPC. Siga las instrucciones de ¿Qué es una interconexión de VPC? en la Guía de interconexión de Amazon Virtual Private Cloud.

Para habilitar el tráfico entre VPC mediante la interconexión de VPC

1. Configure las reglas de enrutamiento de VPC adecuadas para garantizar que el tráfico de red pueda fluir en ambos sentidos.
2. Asegúrese de que el grupo de seguridad de la instancia de base de datos pueda recibir tráfico de entrada del grupo de seguridad del directorio.
3. Asegúrese de que no haya una regla de lista de control de acceso (ACL) a la red para bloquear el tráfico.

Si una cuenta de AWS distinta es la propietaria del directorio, debe compartirlo.

Para compartir el directorio entre cuentas de AWS

1. Comience a compartir el directorio con la cuenta de AWS en la que se creará la instancia de base de datos mediante las instrucciones de Tutorial: Uso compartido del directorio de AWS Managed Microsoft AD para realizar la unión al dominio de EC2 sin problemas en la Guía de administración de Directory Service.
2. Inicie sesión en la consola de Directory Service utilizando la cuenta para la instancia de base de datos y asegúrese de que el dominio tiene el estado `SHARED` antes de continuar.
3. Una vez iniciada sesión en la consola de Directory Service utilizando la cuenta de la instancia de base de datos, anote el valor de Directory ID (ID de directorio). Utilice este identificador de directorio para unir la instancia de base de datos al dominio.

Paso 6: crear o modificar un clúster de base de datos de PostgreSQL

Cree o modifique un clúster de base de datos de PostgreSQL para usarla con su directorio. Puede utilizar la consola, la CLI o la API de RDS para asociar un clúster de base de datos con un directorio. Puede hacerlo de una de las siguientes formas:

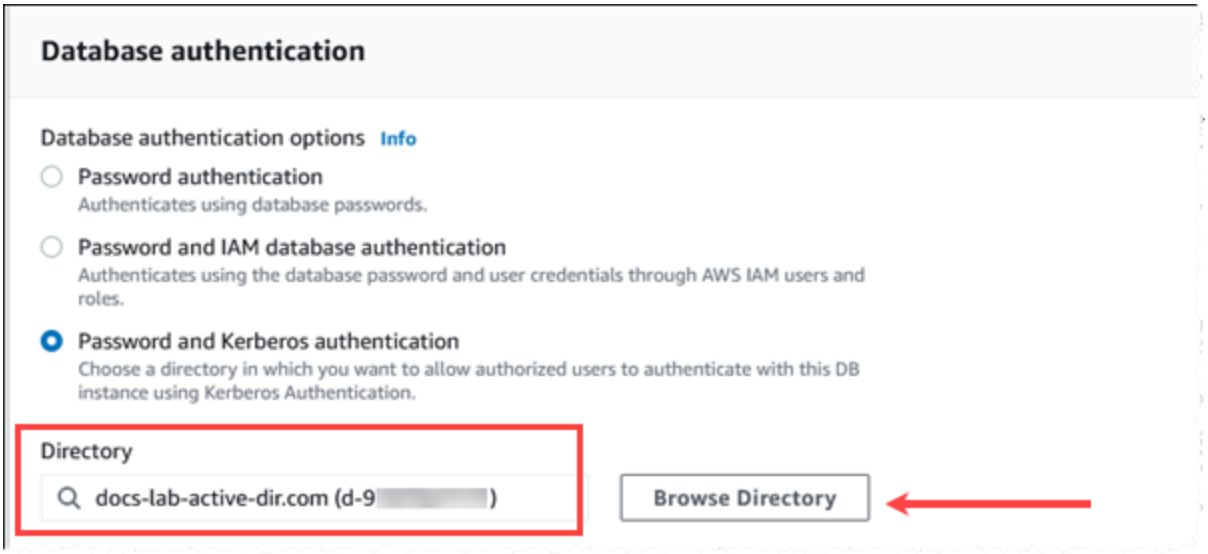
- Cree un nuevo clúster de base de datos de PostgreSQL con la consola, el comando create-db-cluster de la CLI o la operación CreateDBCluster de la API de RDS. Para obtener instrucciones, consulte Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él.
- Modifique un clúster de base de datos PostgreSQL existente mediante la consola, el comando modify-db-cluster de la CLI o la operación ModifyDBCluster de la API de RDS. Para obtener instrucciones, consulte Modificación de un clúster de base de datos de Amazon Aurora.
- Restaure un clúster de base de datos de PostgreSQL a partir de una instantánea de base de datos con la consola, el comando restore-db-cluster-from-db-snapshot de la CLI o la operación RestoreDBClusterFromDBSnapshot de la API de RDS. Para obtener instrucciones, consulte Restauración de una instantánea de clúster de base de datos.
- Restaure un clúster de base de datos de PostgreSQL a un punto en el tiempo con la consola, el comando restore-db-instance-to-point-in-time de la CLI o la operación RestoreDBInstanceToPointInTime de la API de RDS. Para obtener instrucciones, consulte Restauración de un clúster de base de dato a un momento indicado.

La autenticación de Kerberos solo es compatible con instancias de base de datos de PostgreSQL en una VPC. El clúster de DB puede estar en la misma VPC que el directorio o en una VPC diferente. El clúster de base de datos debe usar un grupo de seguridad que permita el ingreso y la salida dentro de la VPC del directorio, de modo que el clúster de base de datos pueda comunicarse con el directorio.

> **Note**
>
> Actualmente, el clúster de base de datos de Aurora PostgreSQL no permite habilitar la autenticación Kerberos durante la migración desde RDS para PostgreSQL. Solo puede habilitar la autenticación Kerberos en un clúster de base de datos de Aurora PostgreSQL independiente.

Consola

Si utiliza la consola para crear, modificar o restaurar un clúster de base de datos, elija Kerberos authentication (Autenticación de Kerberos) en la sección Database authentication (Autenticación de base de datos). Luego, elija Browse Directory (Examinar directorio). Seleccione el directorio o elija Create a new directory (Crear un nuevo directorio) para utilizar Directory Service.



AWS CLI

Cuando utilice la AWS CLI, se necesitan los siguientes parámetros para que el clúster de base de datos pueda usar el directorio que ha creado:

- Para el parámetro `--domain`, utilice el identificador de dominio (identificador "d-*") que se generó cuando creó el directorio.
- Para el parámetro `--domain-iam-role-name`, utilice el rol que creó que usa la política `AmazonRDSDirectoryServiceAccess` de IAM administrada.

Por ejemplo, el siguiente comando de CLI modifica un clúster de base de datos para que use un directorio.

```
aws rds  modify-db-cluster --db-cluster-identifier mydbinstance --domain d-Directory-ID
 --domain-iam-role-name role-name
```

> **Important**
>
> Si modifica un clúster de base de datos para habilitar la autenticación Kerberos, reinicie el clúster de base de datos después de hacer el cambio.

Paso 7: crear usuarios de PostgreSQL para las entidades principales de Kerberos

En este punto, su clúster de base de datos de Aurora PostgreSQL se une al dominio AWS Managed Microsoft AD. Los usuarios que haya creado en el directorio en el Paso 4: crear y configurar usuarios deben configurarse como usuarios de la base de datos de PostgreSQL y tener privilegios para iniciar sesión en la base de datos. Para ello, inicie sesión como usuario de la base de datos con privilegios `rds_superuser`. Por ejemplo, si ha aceptado los valores predeterminados al crear el clúster de base de datos de Aurora PostgreSQL , utiliza `postgres`, tal como se muestra en los pasos siguientes.

Para crear usuarios de la base de datos de PostgreSQL para las entidades principales de Kerberos

1. Use `psql` para conectarse a su punto de conexión de la instancia de base de datos del clúster de base de datos de Aurora PostgreSQL mediante `psql`. En el siguiente ejemplo, se usa la cuenta de `postgres` predeterminada del rol de `rds_superuser`.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Cree un nombre de usuario de base de datos para cada entidad principal de Kerberos (nombre de usuario de Active Directory) al que desee otorgar acceso a la base de datos. Utilice el nombre de usuario canónico (identidad) tal como se define en la instancia de Active Directory, es decir, un `alias` en minúsculas (nombre de usuario en Active Directory) y el nombre en mayúsculas del dominio de Active Directory para ese nombre de usuario. El nombre de usuario de Active Directory es un usuario autenticado externamente, así que utilice comillas alrededor del nombre tal como se muestra a continuación.

   ```
   postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
   CREATE ROLE
   ```

3. Otorgue el rol `rds_ad` al usuario de la base de datos.

   ```
   postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
   ```

```
GRANT ROLE
```

Cuando termine de crear todos los usuarios de PostgreSQL para sus identidades de usuario de Active Directory, los usuarios podrán acceder al clúster de base de datos de Aurora PostgreSQL con sus credenciales de Kerberos.

Es necesario que los usuarios de bases de datos que se autentican mediante Kerberos lo hagan desde máquinas de cliente que sean miembros del dominio de Active Directory.

Los usuarios de bases de datos a los que se les ha otorgado el rol `rds_ad` tampoco pueden tener el rol `rds_iam`. Esto se aplica también a las membresías anidadas. Para obtener más información, consulte Autenticación de bases de datos de IAM .

Configuración del clúster de base de datos de Aurora PostgreSQL para nombres de usuario que no distinguen mayúsculas de minúsculas

Las versiones 14.5, 13.8, 12.12 y 11.17 de Aurora PostgreSQL admiten el parámetro `krb_caseins_users` de PostgreSQL. Este parámetro admite nombres de usuario de Active Directory que no distinguen mayúsculas de minúsculas. De forma predeterminada, este parámetro está configurado en false, por lo que Aurora PostgreSQL interpreta los nombres de usuario con distinción entre mayúsculas y minúsculas. Este es el comportamiento predeterminado en todas las versiones anteriores de Aurora PostgreSQL. Sin embargo, puede establecer este parámetro en `true` en el grupo de parámetros del clúster de base de datos personalizado y permitir que su clúster de base de datos de Aurora PostgreSQL interprete los nombres de usuario sin distinguir mayúsculas de minúsculas. Considere la posibilidad de hacerlo para facilitar la vida a los usuarios de la base de datos, ya que a veces pueden escribir mal las mayúsculas y minúsculas de su nombre de usuario al autenticarse mediante Active Directory.

Para cambiar el parámetro `krb_caseins_users`, su clúster de base de datos de Aurora PostgreSQL debe utilizar un grupo de parámetros del clúster de base de datos personalizado. Para obtener más información acerca de cómo crear un grupo de parámetros del clúster de base de datos personalizado, consulte Grupos de parámetros para Amazon Aurora.

Puede utilizar la AWS CLI o la Consola de administración de AWS para cambiar la configuración. Para obtener más información, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Paso 8: configurar un cliente de PostgreSQL

Para configurar un cliente de PostgreSQL, siga los pasos siguientes:

- Cree un archivo krb5.conf (o equivalente) para apuntar al dominio.
- Verifique que el tráfico puede fluir entre el host cliente y Directory Service. Use una utilidad de red como, por ejemplo, Netcat para lo siguiente:
  - Verificar el tráfico sobre DNS para el puerto 53.
  - Verificar el tráfico sobre TCP/UDP para el puerto 52 y para Kerberos, lo que incluye los puertos 88 y 464 para Directory Service.
- Verifique que el tráfico puede fluir entre el host cliente y la instancia de base de datos sobre el puerto de base de datos. Por ejemplo, utilice psql para conectarse y acceder a la base de datos.

A continuación, se muestra el contenido krb5.conf para AWS Managed Microsoft AD.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

A continuación, se muestra el contenido krb5.conf de ejemplo para un Microsoft Active Directory local.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
```

```
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Administración de un clúster de base de datos de Aurora PostgreSQL en un dominio de Active Directory

Puede usar la consola, la CLI o la API de RDS para administrar el clúster de base de datos y su relación con Microsoft Active Directory. Puede, por ejemplo, asociar un Active Directory para habilitar la autenticación Kerberos. También puede eliminar la asociación para un Active Directory para deshabilitar la autenticación Kerberos. También puede mover un clúster de base de datos que va a ser autenticada externamente por un Microsoft Active Directory a otro.

Por ejemplo, con la CLI, puede hacer lo siguiente:

- Para volver a intentar habilitar la autenticación Kerberos para una suscripción con error, use el comando modify-db-clúster de la CLI. Especifique el ID de directorio de pertenencia actual para la opción `--domain`.
- Para deshabilitar la autenticación Kerberos en una instancia de base de datos, utilice el comando modify-db-clúster de la CLI. Especifique `none` para la opción `--domain`.
- Para mover una instancia de base de datos desde un dominio a otro, utilice el comando modify-db-clúster de la CLI. Especifique el identificador de dominio del nuevo dominio para la opción `--domain`.

Descripción de la pertenencia a los dominios

Después de crear o modificar el clúster de base de datos, las instancias de base de datos se convierten en miembros de un dominio. Puede ver el estado de la suscripción al dominio en la consola o ejecutando el comando describe-db-instances de la CLI. El estado de la instancia de base de datos puede ser uno de los siguientes:

- `kerberos-enabled`: la instancia de base de datos tiene habilitada la autenticación Kerberos.

- `enabling-kerberos` - AWS está en proceso de habilitar la autenticación Kerberos en esta instancia de base de datos.
- `pending-enable-kerberos`: la habilitación de la autenticación Kerberos está pendiente en esta instancia de base de datos.
- `pending-maintenance-enable-kerberos` - AWS intentará habilitar la autenticación Kerberos en la instancia de base de datos durante el próximo periodo de mantenimiento programado.
- `pending-disable-kerberos`: la deshabilitación de la autenticación Kerberos está pendiente en esta instancia de base de datos.
- `pending-maintenance-disable-kerberos` - AWS intentará desactivar la autenticación Kerberos en la instancia de base de datos durante el próximo periodo de mantenimiento programado.
- `enable-kerberos-failed` - Un problema de configuración ha impedido que AWS habilite la autenticación Kerberos en la instancia de base de datos. Corrija el problema de configuración antes de volver a ejecutar el comando para modificar la instancia de base de datos.
- `disabling-kerberos` - AWS está en proceso de desactivar la autenticación Kerberos en esta instancia de base de datos.

Una solicitud para habilitar la autenticación Kerberos puede generar un error a causa de un problema de conectividad de la red o de un rol de IAM incorrecto. En algunos casos, el intento de habilitar la autenticación Kerberos podría producir un error al crear o modificar un clúster de base de datos. En tal caso, asegúrese de que está utilizando el rol de IAM correcto, a continuación modifique el clúster de base de datos para unirse al dominio.

Conexión a PostgreSQL con autenticación Kerberos

Puede conectarse a PostgreSQL con autenticación Kerberos con la interfaz pgAdmin o con una interfaz de línea de comandos como, por ejemplo, psql. Para obtener más información acerca de las conexiones, consulte Conexión a un clúster de base de datos Amazon Aurora PostgreSQL. Para obtener información sobre cómo obtener el punto de conexión, el número de puerto y otros detalles necesarios para la conexión, consulte Visualización de los puntos de enlace para un clúster de Aurora.

> **Note**
>
> La autenticación y el cifrado GSSAPI en PostgreSQL están implementados por la biblioteca Kerberos `libkrb5.so`. Características como `postgres_fdw` y `dblink` también se basan en esta misma biblioteca para las conexiones salientes con autenticación o cifrado Kerberos.

pgAdmin

Para utilizar pgAdmin para conectarse a PostgreSQL con la autenticación Kerberos, siga estos pasos:

1. Lance la aplicación pgAdmin en su equipo cliente.
2. En la pestaña Dashboard (Panel), elija Add New Server (Añadir nuevo servidor).
3. En el cuadro de diálogo Crear - Servidor, escriba un nombre en la pestaña General para identificar el servidor en pgAdmin.
4. En la pestaña Connection (Conexión), introduzca la siguiente información de su base de datos de Aurora PostgreSQL:
   - En Host, introduzca el punto de conexión de la Instancia de escritura de su clúster de base de datos de Aurora PostgreSQL. Un punto de conexión tiene un aspecto similar al siguiente:

     ```
     AUR-cluster-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Para conectarse a un Microsoft Active Directory en las instalaciones desde un cliente de Windows, utilice el nombre de dominio del Active Directory administrado por AWS en lugar de `rds.amazonaws.com` en el punto de conexión del host. Por ejemplo, suponga que el nombre de dominio de AWS Managed Active Directory es `corp.example.com`. Luego, para Host, el punto de conexión se especificaría de la siguiente manera:

     ```
     AUR-cluster-instance.111122223333.aws-region.corp.example.com
     ```

   - En Puerto, escriba el puerto asignado.
   - En Base de datos de mantenimiento, escriba el nombre de la base de datos inicial a la que se conectará el cliente.
   - En Nombre de usuario, escriba el nombre de usuario que especificó para la autenticación Kerberos en Paso 7: crear usuarios de PostgreSQL para las entidades principales de Kerberos.

5. Seleccione Save.

Psql

Para utilizar psql para conectar a PostgreSQL con autenticación Kerberos, siga los pasos siguientes:

1. En el símbolo del sistema, ejecute el siguiente comando.

   ```
   kinit username
   ```

   Sustituya *username* por el nombre de usuario. En el símbolo del sistema, introduzca la contraseña almacenada en Microsoft Active Directory para el usuario.

2. Si el clúster de base de datos de PostgreSQL utiliza una VPC accesible públicamente, coloque una dirección IP para su punto de conexión de clúster de base de datos en su archivo `/etc/hosts` en el cliente EC2. Por ejemplo, los comandos siguientes obtienen la dirección IP y, a continuación, la ponen en el archivo `/etc/hosts`.

   ```
   % dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
   ;; Truncated, retrying in TCP mode.
   ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
   34.210.197.118

   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
   hosts
   ```

   Si utiliza un Microsoft Active Directory en las instalaciones desde un cliente de Windows, tiene que conectarse mediante un punto de enlace especializado. En lugar de utilizar el dominio de Amazon `rds.amazonaws.com` en el punto de conexión del host, utilice el nombre de dominio de AWS Managed Active Directory.

   Por ejemplo, suponga que el nombre de dominio de su AWS Managed Active Directory es `corp.example.com`. A continuación, use el formato *PostgreSQL-endpoint.AWS-Region.corp.example.com* para el punto de enlace y colóquelo en el archivo `/etc/hosts`.

   ```
   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
   hosts
   ```

3. Utilice el comando psql siguiente para iniciar sesión en una de clúster de base de datos de PostgreSQL que está integrada con Active Directory. Utilice un punto de enlace de clúster o instancia.

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.rds.amazonaws.com postgres
```

   Para iniciar sesión en el clúster de base de datos de PostgreSQL desde un cliente de Windows utilizando un Active Directory en las instalaciones, utilice el siguiente comando psql con el nombre de dominio del paso anterior (`corp.example.com`):

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

Uso de grupos de seguridad de AD para el control de acceso de Aurora PostgreSQL

A partir de las versiones 14.10 y 15.5 de Aurora PostgreSQL, el control de acceso de Aurora PostgreSQL se puede administrar mediante AWS Directory Service para los grupos de seguridad de Microsoft Active Directory (AD). Las versiones anteriores de Aurora PostgreSQL admiten la autenticación basada en Kerberos con AD solo para usuarios individuales. Cada usuario de AD tenía que estar aprovisionado de forma explícita en el clúster de base de datos para poder obtener el acceso.

En lugar de aprovisionar explícitamente a cada usuario de AD en un clúster de base de datos en función de las necesidades empresariales, puede aprovechar los grupos de seguridad de AD, tal como se explica a continuación:

- Los usuarios de AD son miembros de varios grupos de seguridad de AD en un Active Directory. No los determina el administrador del clúster de base de datos, sino que se basan en los requisitos empresariales y los gestiona un administrador de AD.
- Los administradores de clústeres de bases de datos crean roles de base de datos en las instancias de base de datos en función de los requisitos empresariales. Estos roles de base de datos pueden tener permisos o privilegios diferentes.
- Los administradores de clústeres de bases de datos configuran un mapeo entre los grupos de seguridad de AD y los roles de base de datos por clúster de base de datos.
- Los usuarios de bases de datos pueden acceder a los clústeres de bases de datos con sus credenciales de AD. El acceso se basa en la pertenencia a un grupo de seguridad de AD. Los usuarios de AD obtienen o pierden el acceso automáticamente en función de su pertenencia a grupos de AD.

Requisitos previos

Asegúrese de tener lo siguiente antes de configurar la extensión para los grupos de seguridad de AD:

- Configuración de la autenticación Kerberos para clústeres de base de datos de PostgreSQL. Para obtener más información, consulte Configuración de autenticación Kerberos para clústeres de base de datos de PostgreSQL.

> **Note**
>
> Para los grupos de seguridad de AD, omita el paso 7: creación de usuarios de PostgreSQL para las entidades principales de Kerberos de este procedimiento de configuración.

- Administración de un clúster de base de datos en un dominio Para obtener más información, consulte Administración de un clúster de base de datos en un dominio.

Configuración de la extensión pg_ad_mapping

Aurora PostgreSQL ahora ofrece una extensión `pg_ad_mapping` para administrar el mapeo entre los grupos de seguridad de AD y los roles de base de datos en el clúster de Aurora PostgreSQL. Para obtener más información sobre las funciones que proporciona `pg_ad_mapping`, consulte Uso de las funciones de la extensión `pg_ad_mapping`.

Para configurar la extensión `pg_ad_mapping` en el clúster de base de datos de Aurora PostgreSQL, primero debe agregar `pg_ad_mapping` a las bibliotecas compartidas en el grupo de parámetros del clúster de base de datos personalizado para su clúster de base de datos de Aurora PostgreSQL. Para obtener información acerca de cómo crear un grupo de parámetros del clúster de base de datos personalizado, consulte Grupos de parámetros para Amazon Aurora. A continuación, instale la extensión `pg_ad_mapping`. Los procedimientos de esta sección le muestran cómo hacerlo. Puede utilizar la Consola de administración de AWS o la AWS CLI.

Debe tener permisos como el rol `rds_superuser` para realizar todas estas tareas.

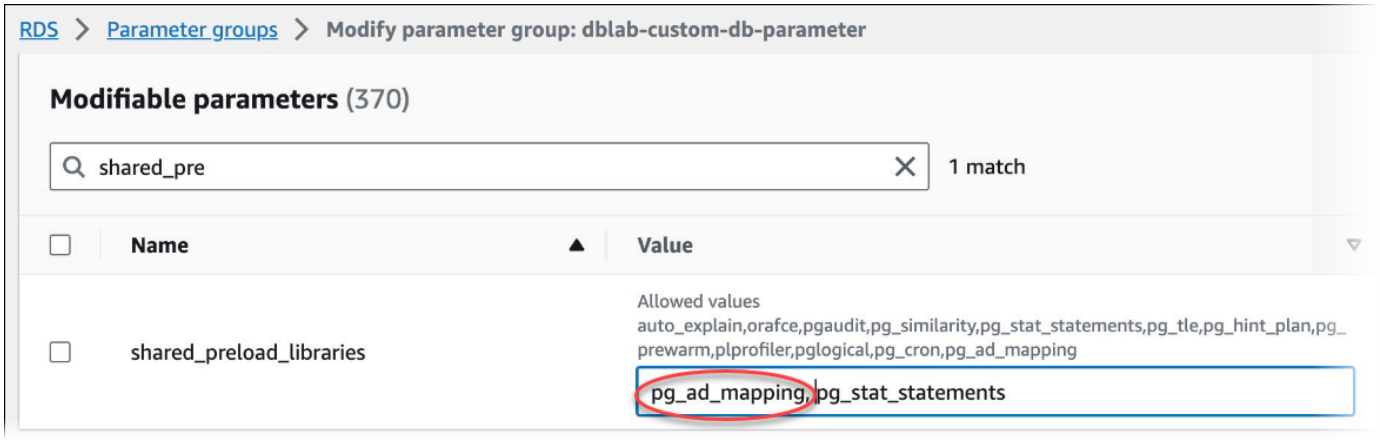
En los pasos siguientes se supone que su clúster de base de datos de Aurora PostgreSQL está asociado a un grupo de parámetros de clúster de base de datos personalizado.

Consola

Para configurar la extensión **`pg_ad_mapping`**

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija la instancia del escritor del clúster de base de datos de Aurora PostgreSQL.

3. Abra la pestaña Configuración para su instancia de escritor del clúster de base de datos de Aurora PostgreSQL. Entre los detalles de la instancia, busque el enlace del grupo de parámetros.
4. Elija el enlace para abrir los parámetros personalizados asociados al clúster de base de datos de Aurora PostgreSQL.
5. En el campo de búsqueda Parametes (Parámetros), escriba `shared_pre` para buscar el parámetro `shared_preload_libraries`.
6. Seleccione Edit parameters (Editar parámetros) para acceder a los valores de las propiedades.
7. Añada `pg_ad_mapping` a la lista en el campo Values (Valores). Utilice una coma para separar los elementos de la lista de valores.



8. Reinicie la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que se aplique el cambio en el parámetro `shared_preload_libraries`.
9. Cuando la instancia esté disponible, verifique si se ha inicializado `pg_ad_mapping`. Use `psql` para conectarse a la instancia de escritor de su clúster de bases de datos de Aurora PostgreSQL y, a continuación, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_ad_mapping
(1 row)
```

10. Con `pg_ad_mapping` inicializado, ahora ya puede crear la extensión. Debe crear la extensión después de inicializar la biblioteca para empezar a utilizar las funciones que proporciona la extensión.

```
CREATE EXTENSION pg_ad_mapping;
```

11. Cierre la sesión de `psql`.

```
labdb=> \q
```

AWS CLI

Para configurar pg_ad_mapping

Para configurar pg_ad_mapping mediante la AWS CLI, llame a la operación modify-db-parameter-group para agregar este parámetro a su grupo de parámetros personalizado, tal como se muestra en el siguiente procedimiento.

1. Utilice el siguiente comando AWS CLI para añadir `pg_ad_mapping` al parámetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_ad_mapping,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilice el siguiente comando AWS CLI para reiniciar la instancia de escritura del clúster de base de datos de Aurora PostgreSQL para que se inicialice la biblioteca pg_ad_mapping.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Cuando la instancia esté disponible, puede verificar si `pg_ad_mapping` se ha inicializado. Use `psql` para conectarse a la instancia de escritor de su clúster de bases de datos de Aurora PostgreSQL y, a continuación, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_ad_mapping
```

```
(1 row)
```

Con pg_ad_mapping inicializado, ahora ya puede crear la extensión.

```
CREATE EXTENSION pg_ad_mapping;
```

4. Cierre la sesión de `psql` para poder utilizar AWS CLI.

```
labdb=> \q
```

Recuperación del SID del grupo de Active Directory en PowerShell

Se utiliza un identificador de seguridad (SID) para identificar de forma exclusiva una entidad principal de seguridad o un grupo de seguridad. Siempre que se crea un grupo o una cuenta de seguridad en Active Directory, se le asigna un SID. Para obtener el SID del grupo de seguridad de AD desde Active Directory, puede usar el cmdlet Get-ADGroup desde un equipo cliente Windows que esté unido a ese dominio de Active Directory. El parámetro Identity especifica el nombre del grupo de Active Directory para obtener el SID correspondiente.

En el siguiente ejemplo, se devuelve el SID del grupo de AD *`adgroup1`*.

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID

             SID
-----------------------------------------------
S-1-5-21-3168537779-1985441202-1799118680-1612
```

Asignación del rol de base de datos al grupo de seguridad de AD

Debe aprovisionar explícitamente los grupos de seguridad de AD en la base de datos como un rol de base de datos de PostgreSQL. Un usuario de AD que forme parte de al menos un grupo de seguridad de AD aprovisionado tendrá acceso a la base de datos. No debe conceder `rds_ad role` a un rol de base de datos basado en un grupo de seguridad de AD. La autenticación Kerberos para el grupo de seguridad se activará mediante el sufijo del nombre de dominio, como *`user1@example.com`*. Este rol de base de datos no puede usar la autenticación por contraseña o IAM para acceder a la base de datos.

> **Note**
>
> Los usuarios de AD que tengan un rol de base de datos correspondiente en la base de datos y el rol `rds_ad` concedido no pueden iniciar sesión como parte del grupo de seguridad de AD. Obtendrán acceso a través del rol de base de datos como usuarios individuales.

Por ejemplo, accounts-group es un grupo de seguridad de AD y querría aprovisionar este grupo de seguridad en Aurora PostgreSQL como accounts-role.

Grupo de seguridad de AD	Rol de base de datos de PosgreSQL
accounts-group	accounts-role

Al asignar el rol de base de datos al grupo de seguridad de AD, debe asegurarse de que el rol de la base de datos tenga el atributo LOGIN establecido y el privilegio CONNECT en la base de datos de inicio de sesión requerida.

```
postgres => alter role accounts-role login;

ALTER ROLE
postgres => grant connect on database accounts-db to accounts-role;
```

El administrador ahora puede proceder a crear la asignación entre el grupo de seguridad de AD y el rol de base de datos de PostgreSQL.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', <SID>, <Weight>);
```

Para obtener información sobre cómo recuperar el SID del grupo de seguridad de AD, consulte Recuperación del SID del grupo de Active Directory en PowerShell.

Puede haber casos en los que un usuario de AD pertenezca a varios grupos; en ese caso, el usuario de AD heredará los privilegios del rol de la base de datos, que se aprovisionó con una mayor ponderación. Si los dos roles tienen la misma ponderación, el usuario de AD heredará los privilegios del rol de base de datos correspondiente a la asignación que se efectuase más recientemente. La recomendación es especificar ponderaciones que reflejen los permisos y privilegios relativos de los roles de base de datos individuales. Cuanto mayores sean los permisos o privilegios de un rol de

base de datos, mayor será la ponderación que se le deberá asociar a la entrada de la asignación. Esto evitará la ambigüedad de que dos asignaciones tengan la misma ponderación.

En la siguiente tabla se muestra un ejemplo de asignación de grupos de seguridad de AD a roles de base de datos de Aurora PostgreSQL.

Grupo de seguridad de AD	Rol de base de datos de PosgreSQL	Peso
accounts-group	accounts-role	7
sales-group	sales-role	10
dev-group	dev-role	7

En el siguiente ejemplo, `user1` heredará los privilegios de sales-role, ya que tiene mayor ponderación, mientras que `user2` heredará los privilegios de `dev-role`, ya que la asignación para este rol se creó después de `accounts-role`, que comparte la misma ponderación que `accounts-role`.

Nombre de usuario	Pertenencia al grupo de seguridad
user1	accounts-group sales-group
user2	accounts-group dev-group

A continuación se muestran los comandos psql para establecer, enumerar y borrar las asignaciones. En la actualidad, no es posible modificar una única entrada de asignación. Es necesario eliminar la entrada existente y volver a crear la asignación.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', 'S-1-5-67-890',
 7);
admin=>select pgadmap_set_mapping('sales-group', 'sales-role', 'S-1-2-34-560', 10);
admin=>select pgadmap_set_mapping('dev-group', 'dev-role', 'S-1-8-43-612', 7);

admin=>select * from pgadmap_read_mapping();
```

```
ad_sid       | pg_role        | weight | ad_grp
-------------+----------------+--------+--------------
S-1-5-67-890 | accounts-role | 7      | accounts-group
S-1-2-34-560 | sales-role    | 10     | sales-group
S-1-8-43-612 | dev-role      | 7      | dev-group
(3 rows)
```

Registro/auditoría de la identidad de los usuarios de AD

Utilice el siguiente comando para determinar el rol de base de datos heredado por el usuario actual o de la sesión:

```
postgres=>select session_user, current_user;

session_user | current_user
-------------+--------------
dev-role     | dev-role

(1 row)
```

Para determinar la identidad de la entidad principal de seguridad de AD, utilice el siguiente comando:

```
postgres=>select principal from pg_stat_gssapi where pid = pg_backend_pid();

 principal
------------------------
 user1@example.com

(1 row)
```

Actualmente, la identidad del usuario de AD no está visible en los registros de auditoría. El parámetro `log_connections` se puede habilitar para registrar el establecimiento de la sesión de base de datos. Consulte log_connections para obtener más información. El resultado de esto incluye la identidad del usuario de AD, como se muestra a continuación. A continuación, el PID del backend asociado a este resultado puede ayudar a atribuir las acciones al usuario real de AD.

```
pgrole1@postgres:[615]:LOG: connection authorized: user=pgrole1
 database=postgres application_name=psql GSS (authenticated=yes, encrypted=yes,
 principal=Admin@EXAMPLE.COM)
```

Limitaciones

- El ID de Microsoft Entra conocido como Azure Active Directory no es compatible.

Uso de las funciones de la extensión **pg_ad_mapping**

La extensión pg_ad_mapping ofrece compatibilidad con las siguientes funciones:

pgadmap_set_mapping

Esta función establece la asignación entre el grupo de seguridad de AD y el rol de base de datos con una ponderación asociada.

Sintaxis

```
pgadmap_set_mapping(
ad_group,
db_role,
ad_group_sid,
weight)
```

Argumentos

Parámetro	Descripción
ad_group	Nombre del grupo de AD. El valor no puede ser una cadena vacía o nula.
db_role	El rol de la base de datos que se asignará al grupo de AD especificado. El valor no puede ser una cadena vacía o nula.
ad_group_sid	Identificador de seguridad que se utiliza para identificar de forma exclusiva al grupo de AD. El valor comienza por 'S-1-' y no puede ser una cadena vacía o nula. Para obtener más informaci

Parámetro	Descripción
	ón, consulte Recuperación del SID del grupo de Active Directory en PowerShell.
weight	Ponderación asociada al rol de la base de datos. El rol con mayor ponderación tiene prioridad cuando el usuario es miembro de varios grupos. El valor de ponderación predeterm inado es 1.

Tipo de retorno

None

Notas de uso

Esta función agrega una nueva asignación del grupo de seguridad de AD al rol de la base de datos. Solo lo puede ejecutar en la instancia de base de datos principal del clúster de base de datos un usuario que tenga el privilegio rds_superuser.

Ejemplos

```
postgres=> select pgadmap_set_mapping('accounts-group','accounts-
role','S-1-2-33-12345-67890-12345-678',10);

pgadmap_set_mapping

(1 row)
```

pgadmap_read_mapping

Esta función muestra las asignaciones entre el grupo de seguridad de AD y el rol de base de datos que se establecieron mediante la función `pgadmap_set_mapping`.

Sintaxis

```
pgadmap_read_mapping()
```

Argumentos

None

Tipo de retorno

Parámetro	Descripción
ad_group_sid	Identificador de seguridad que se utiliza para identificar de forma exclusiva al grupo de AD. El valor comienza por 'S-1-' y no puede ser una cadena vacía o nula. Para obtener más informaci ón, consulte Recuperación del SID del grupo de Active Directory en PowerShell.accounts-role@example.com
db_role	El rol de la base de datos que se asignará al grupo de AD especificado. El valor no puede ser una cadena vacía o nula.
weight	Ponderación asociada al rol de la base de datos. El rol con mayor ponderación tiene prioridad cuando el usuario es miembro de varios grupos. El valor de ponderación predeterm inado es 1.
ad_group	Nombre del grupo de AD. El valor no puede ser una cadena vacía o nula.

Notas de uso

Llame a esta función para obtener una lista de todas las asignaciones disponibles entre el grupo de seguridad de AD y el rol de base de datos.

Ejemplos

```
postgres=> select * from pgadmap_read_mapping();

ad_sid                              | pg_role       | weight | ad_grp
------------------------------------+---------------+--------+-----------------
S-1-2-33-12345-67890-12345-678      | accounts-role | 10     | accounts-group
(1 row)

(1 row)
```

pgadmap_reset_mapping

Esta función restablece una o todas las asignaciones que se establecieron mediante la función `pgadmap_set_mapping`.

Sintaxis

```
pgadmap_reset_mapping(
ad_group_sid,
db_role,
weight)
```

Argumentos

Parámetro	Descripción
ad_group_sid	Identificador de seguridad que se utiliza para identificar de forma exclusiva al grupo de AD.
db_role	El rol de la base de datos que se asignará al grupo de AD especificado.
weight	Ponderación asociada al rol de la base de datos.

Si no se proporcionan argumentos, se restablecen todas las asignaciones de roles de base de datos de los grupos de AD. Es necesario proporcionar todos los argumentos o ninguno.

Tipo de retorno

`None`

Notas de uso

Llame a esta función para eliminar la asignación de roles de un grupo de AD específico a la base de datos o para restablecer todas las asignaciones. Esta función solo la puede ejecutar en la instancia de base de datos principal del clúster de base de datos un usuario que tenga el privilegio `rds_superuser`.

Ejemplos

```
postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+-------------------
 S-1-2-33-12345-67890-12345-678 | accounts-role| 10          | accounts-group
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(2 rows)
postgres=> select pgadmap_reset_mapping('S-1-2-33-12345-67890-12345-678', 'accounts-
role', 10);

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+---------------
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(1 row)
postgres=> select pgadmap_reset_mapping();

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+--------------
 (0 rows)
```

Migración de datos a Amazon Aurora con compatibilidad con PostgreSQL

Tiene varias opciones para migrar datos desde una base de datos a un clúster de base de datos de Edición compatible con Amazon Aurora PostgreSQL. Las opciones de migración dependen también

de la base de datos desde la que se realiza la migración y del tamaño de los datos que se van a migrar. A continuación se muestran sus opciones:

Migración de una instancia de base de datos de RDS for PostgreSQL mediante una instantánea

Puede migrar datos directamente a partir de una instantánea de base de datos RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL.

Migración de una instancia de base de datos de RDS for PostgreSQL mediante una réplica de lectura de Aurora

También puede migrar desde una instancia de base de datos de RDS for PostgreSQL creando una réplica de lectura de Aurora PostgreSQL de una instancia de base de datos de RDS for PostgreSQL. Cuando el retraso de réplica entre la instancia de base de datos de RDS for PostgreSQL y la réplica de lectura de Aurora PostgreSQL es cero, puede detener la reproducción. En este momento, puede convertir la réplica de lectura de Aurora en un clúster de base de datos de Aurora PostgreSQL independiente de lectura y escritura.

Importación de datos de Amazon S3 a RDS para Aurora PostgreSQL

Puede migrar datos al importarlos desde una tabla perteneciente Amazon S3 a un clúster de Aurora PostgreSQL basede datos.

Migración desde una base de datos no compatible con PostgreSQL

Puede utilizar AWS Database Migration Service (AWS DMS) para migrar datos desde una base de datos que no sea compatible con PostgreSQL. Para obtener más información acerca de AWS DMS, consulte ¿Qué es el Database Migration Service de AWS? en la guía del usuario de AWS Database Migration Service.

> **Note**
>
> Actualmente, el clúster de base de datos de Aurora PostgreSQL no permite habilitar la autenticación Kerberos durante la migración desde RDS para PostgreSQL. Solo puede habilitar la autenticación Kerberos en un clúster de base de datos de Aurora PostgreSQL independiente.

Para obtener una lista de las Regiones de AWS en las que está disponible Aurora, consulte Amazon Aurora en la Referencia general de AWS.

> **Important**
>
> Si planea migrar una instancia de base de datos de RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL en un futuro próximo, se recomienda encarecidamente que desactive las actualizaciones automáticas de versiones secundarias para la instancia de base de datos al principio de la fase de planificación de la migración. La migración a Aurora PostgreSQL podría retrasarse si la versión de RDS para PostgreSQL aún no es compatible con Aurora PostgreSQL.
> Para obtener información acerca de Aurora PostgreSQL las versiones, vea Versiones del motor para Amazon Aurora PostgreSQL.

Migración de una instantánea de una instancia de base de datos de RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL

Para crear un clúster de base de datos de Aurora PostgreSQL, puede migrar una instantánea de base de datos de una instancia de base de datos de RDS for PostgreSQL. El nuevo clúster de base de datos de Aurora PostgreSQL se completa con los datos de la instancia de base de datos de RDS for PostgreSQL original. Para obtener más información acerca de la creación de una instantánea de base de datos, consulte Creación de una instantánea de base de datos.

En algunos casos, la instantánea de base de datos puede no estar en la Región de AWS en la que desee ubicar los datos. Si es así, utilice la consola de Amazon RDS para copiar la instantánea de base de datos en esa Región de AWS. Para obtener más información acerca de la copia de una instantánea de base de datos, consulte Copia de una instantánea de base de datos.

Puede migrar instantáneas de RDS for PostgreSQL compatibles con las versiones de Aurora PostgreSQL disponibles en la Región de AWS determinada. Por ejemplo, una instantánea de una instancia de base de datos de RDS for PostgreSQL 11.1 se puede migrar a las versiones 11.4, 11.7, 11.8 u 11.9 de Aurora PostgreSQL en la región Oeste de EE. UU. (Norte de California). Una instantánea de RDS for PostgreSQL 10.11 se puede migrar a Aurora PostgreSQL 10.11, 10.12, 10.13 y 10.14. En otras palabras, la instantánea de RDS for PostgreSQL debe usar la misma versión secundaria o inferior que la versión de Aurora PostgreSQL.

También puede elegir que el nuevo clúster de base de datos de Aurora PostgreSQL se cifre en reposo utilizando una AWS KMS key. Esta opción solo está disponible para las instantáneas de base de datos no cifradas.

Para migrar una instantánea de base de datos de RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL, puede usar la Consola de administración de AWS, la AWS CLI o la API de RDS. Cuando se utiliza la Consola de administración de AWS, la consola realiza las acciones necesarias para crear tanto el clúster de base de datos como la instancia principal.

Consola

Para migrar una instantánea de base de datos PostgreSQL con la consola de RDS, realice el siguiente procedimiento:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Snapshots (Instantáneas).
3. En la página Snapshots (Instantáneas), elija la instantánea de RDS for PostgreSQL que desea migrar a un clúster de base de datos de Aurora PostgreSQL.
4. Elija Actions (Acciones) y elija Migrate snapshot (Migrar instantánea).
5. Defina los siguientes valores en la página Migrate Database (Migrar base de datos):

   - DB engine version (Versión del motor de base de datos): elija una versión del motor de base de datos que desee utilizar para la nueva instancia migrada.
   - DB Instance Identifier (Identificador de instancias de bases de datos): ingrese un nombre para el clúster de base de datos que sea único para su cuenta en la Región de AWS que eligió. Este identificador se utiliza en las direcciones de punto de enlace para las instancias del clúster de base de datos. Puede optar por agregar al nombre información como la Región de AWS y el motor de base de datos que eligió, por ejemplo, **`aurora-cluster1`**.

     El identificador de instancias de bases de datos tiene las siguientes limitaciones:

     - Debe incluir entre 1 y 63 caracteres alfanuméricos o guiones.
     - El primer carácter debe ser una letra.
     - No puede terminar con un guion ni contener dos guiones consecutivos.
     - Debe ser único para todas las instancias de base de datos por cada cuenta de AWS y por cada Región de AWS.
   - Clase de instancia de base de datos: elija una clase de instancia de base de datos que tenga el almacenamiento y la capacidad requeridos para la base de datos, por ejemplo `db.r6g.large`. Los volúmenes de clúster de Aurora crecen automáticamente a medida que se incrementa la cantidad de datos de la base de datos. Por lo tanto, solo tiene que

elegir una clase de instancia de base de datos que se adapte a sus necesidades actuales de almacenamiento. Para obtener más información, consulte Información general del almacenamiento de Amazon Aurora.

- Virtual Private Cloud (VPC): si ya dispone de una VPC, puede utilizarla con su clúster de base de datos de Aurora PostgreSQL seleccionando el identificador de la VPC, por ejemplo, `vpc-a464d1c1`. Para obtener más información acerca de la creación de una VPC, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

  De lo contrario, puede optar por hacer que Amazon RDS cree una VPC por usted eligiendo Create a new VPC (Crear una VPC nueva).

- DB Subnet Group (Grupo de subred de BD): si dispone de un grupo de subred existente, puede utilizarlo con su clúster de base de datos Aurora PostgreSQL eligiendo el identificador del grupo de subred, por ejemplo, `gs-subnet-group1`.

- Public Access (Acceso público): elija No para especificar que solo pueden obtener acceso a las instancias de su clúster de base de datos los recursos que se encuentran dentro de su VPC. Elija Yes (Sí) para especificar que los recursos de la red pública pueden obtener acceso a las instancias de su clúster de base de datos.

  > **Note**
  >
  > No es necesario que su clúster de base de datos de producción esté en una subred pública, ya que solo los servidores de su aplicación necesitan acceso a su clúster de base de datos. Si no es necesario que su clúster de base de datos esté en una subred pública, defina Public Access (Acceso público) como No.

- VPC security group (Grupo de seguridad de VPC): elija un grupo de seguridad de VPC para permitir el acceso a la base de datos.

- Availability Zone (Zona de disponibilidad): elija la zona de disponibilidad para alojar la instancia principal de su clúster de base de datos Aurora PostgreSQL. Para hacer que Amazon RDS elija un valor de Availability Zone (Zona de disponibilidad), elija No Preference (Sin preferencia).

- Database Port (Puerto de base de datos): especifique el puerto predeterminado que se utilizará al conectar a instancias del clúster de base de datos Aurora PostgreSQL. El valor predeterminado es `5432`.

> **Note**
>
> Es posible que se encuentre detrás de un firewall de una compañía que no permite el acceso a los puertos predeterminados, como el puerto predeterminado de PostgreSQL, el 5432. En este caso, proporcione un valor de puerto permitido por el firewall corporativo. Recuerde el valor del puerto cuando se conecte más adelante al clúster de base de datos de Aurora PostgreSQL.

- Enable Encryption (Habilitar cifrado): elija Enable Encryption (Habilitar cifrado) para que el nuevo clúster de base de datos de Aurora PostgreSQL se cifre en reposo. Elija también una clave de KMS como valor de AWS KMS key.
- Auto Minor Version Upgrade (Actualización automática de versiones secundarias): seleccione Enable auto minor version upgrade (Habilitar actualización automática de versiones secundarias) si desea habilitar su clúster de base de datos Aurora PostgreSQL para recibir actualizaciones de las versiones secundarias del motor de base de datos PostgreSQL automáticamente cuando estén disponibles.

  La opción Auto Minor Version Upgrade (Actualización automática a versiones secundarias) solo es válida para las actualizaciones secundarias de las versiones del motor de PostgreSQL para su clúster de base de datos Aurora PostgreSQL. No tiene validez para los parches periódicos que se utilizan para mantener la estabilidad del sistema.

6. Elija Migrate (Migrar) para migrar la instantánea de base de datos.
7. Elija Databases (Bases de datos) para ver el nuevo clúster de base de datos. Elija el nuevo clúster de base de datos para supervisar el progreso de la migración. Cuando se complete la migración, el estado del clúster será Available (Disponible). En la pestaña Connectivity & security (Conectividad y seguridad), puede encontrar el punto de enlace del clúster que se va a utilizar para conectarse a la instancia de escritor principal del clúster de base de datos. Para obtener más información acerca de la conexión a un clúster de base de datos de Aurora PostgreSQL, consulte Conexión a un clúster de base de datos Amazon Aurora.

AWS CLI

El uso de AWS CLI para migrar una instantánea de base de datos de RDS for PostgreSQL a una de Aurora PostgreSQL implica dos comandos AWS CLI separados. En primer lugar, se usa el comando `restore-db-cluster-from-snapshot` de AWS CLI para crear un nuevo clúster de bases de datos de Aurora PostgreSQL. A continuación, se usa el comando `create-db-instance` para crear

la instancia de base de datos principal en el nuevo clúster para completar la migración. El siguiente procedimiento crea un clúster de base de datos de Aurora PostgreSQL con una instancia de base de datos principal que tiene la misma configuración que la instancia de base de datos usada para crear la instantánea.

Para migrar una instantánea de base de datos de RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL

1. Use el comando describe-db-snapshots para obtener información sobre la instantánea de base de datos que desea migrar. Puede especificar el parámetro `--db-instance-identifier` o el `--db-snapshot-identifier` en el comando. Si no especifica uno de estos parámetros, obtendrá todas las instantáneas.

   ```
   aws rds describe-db-snapshots --db-instance-identifier <your-db-instance-name>
   ```

2. El comando muestra todos los detalles de configuración de las instantáneas creadas a partir de la instancia de base de datos especificada. En la respuesta, busque la instantánea que desea migrar y localice el nombre de recurso de Amazon (ARN). Para obtener más información sobre los ARN de Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS). Un ARN tiene un aspecto similar a la siguiente imagen.

   ```
   “DBSnapshotArn": "arn:aws:rds:aws-region:111122223333:snapshot:<snapshot_name>"
   ```

   También en la respuesta puede encontrar detalles de configuración para la instancia de base de datos de RDS for PostgreSQL, como la versión del motor, el almacenamiento asignado, si la instancia de base de datos está cifrada o no, etc.

3. Use el comando restore-db-cluster-from-snapshot para iniciar la migración. Especifique los siguientes parámetros:

   - `--db-cluster-identifier`: el nombre que desea dar al clúster de base de datos de Aurora PostgreSQL. Este clúster de base de datos de Aurora es el destino de la migración de la instantánea de base de datos.
   - `--snapshot-identifier`: el nombre de recurso de Amazon (ARN) de la instantánea de base de datos que se va a migrar.
   - `--engine`: especifica `aurora-postgresql` para el motor del clúster de bases de datos de Aurora.

- `--kms-key-id`: este parámetro opcional le permite crear un clúster de base de datos de Aurora PostgreSQL cifrado a partir de una instantánea de base de datos sin cifrar. También le permite elegir una clave de cifrado diferente para el clúster de base de datos que la clave utilizada para la instantánea de base de datos.

> **Note**
>
> No puede crear un clúster de base de datos de Aurora PostgreSQL sin cifrar a partir de una instantánea de base de datos cifrada.

Sin el parámetro `--kms-key-id` especificado como se muestra a continuación, el comando de la AWS CLI restore-db-cluster-from-snapshot crea un clúster de base de datos de Aurora PostgreSQL vacío que está cifrado con la misma clave que la instantánea de base de datos o no está cifrado si la instantánea de base de datos de origen no está cifrada.

Para Linux, macOS o:Unix

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier cluster-name \
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name \
    --engine aurora-postgresql
```

En:Windows

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier new_cluster ^
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name ^
    --engine aurora-postgresql
```

4. El comando muestra detalles sobre el clúster de base de datos de Aurora PostgreSQL que se creó para la migración. Puede comprobar el estado del clúster de base de datos de Aurora PostgreSQL con el comando de la AWS CLI describe-db-clusters.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-name
```

5. Cuando el clúster de base de datos esté “disponible”, use el comando create-db-instance para rellenar el clúster de base de datos de Aurora PostgreSQL con la instancia de base de datos basada en su instantánea de base de datos de Amazon RDS. Especifique los siguientes parámetros:

   - `--db-cluster-identifier`: el nombre del nuevo clúster de base de datos Aurora PostgreSQL que creó en el paso anterior.
   - `--db-instance-identifier`: el nombre que desea dar a la instancia de base de datos. Esta instancia se convierte en el nodo principal de su clúster de base de datos de Aurora PostgreSQL.
   - `----db-instance-class` : especifica la clase de instancia de base de datos que se va a usar. Elija una de las clases de instancia de base de datos admitidas por la versión de Aurora PostgreSQL a la que va a migrar. Para obtener más información, consulte Tipos de clase de instancia de base de datos y Motores de base de datos compatibles para clases de instancia de base de datos.
   - `--engine`: especifica `aurora-postgresql` para la instancia de base de datos.

   También puede crear la instancia de base de datos con una configuración diferente a la de la instantánea de base de datos de origen si pasa las opciones adecuadas en el comando `create-db-instance` de la AWS CLI. Para obtener más información, consulte el comando create-db-instance.

   Para Linux, macOS o:Unix

   ```
   aws rds create-db-instance \
       --db-cluster-identifier cluster-name \
       --db-instance-identifier --db-instance-class db.instance.class \
       --engine aurora-postgresql
   ```

   En:Windows

   ```
   aws rds create-db-instance ^
       --db-cluster-identifier cluster-name ^
       --db-instance-identifier --db-instance-class db.instance.class ^
       --engine aurora-postgresql
   ```

Cuando finaliza el proceso de migración, el clúster de Aurora PostgreSQL tiene una instancia de base de datos principal rellenada.

Migración de datos desde una instancia de base de datos de RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL utilizando una réplica de lectura de Aurora

Para realizar el proceso de migración, puede utilizar una instancia de base de datos de RDS for PostgreSQL como base para un nuevo clúster de base de datos de Aurora PostgreSQL mediante el uso de una réplica de lectura de Aurora. La opción de réplica de lectura de Aurora solo está disponible para migrar dentro de la misma Región de AWS y cuenta, y solo está disponible si la región ofrece una versión compatible de Aurora PostgreSQL para su instancia de base de datos de RDS for PostgreSQL. Compatible significa que la versión de Aurora PostgreSQL es la misma que la versión de RDS for PostgreSQL o que es una versión secundaria superior de la misma familia de versiones principales.

Por ejemplo, para utilizar esta técnica de migración de una instancia de base de datos de RDS for PostgreSQL 11.14, la región debe ofrecer la versión 11.14 de Aurora PostgreSQL o una versión secundaria superior de la familia PostgreSQL, versión 11.

Temas

- Información general de la migración de datos mediante una réplica de lectura de Aurora
- Preparación para la migración de datos mediante una réplica de lectura de Aurora
- Creación de una réplica de lectura de Aurora
- Promoción de una réplica de lectura de Aurora

Información general de la migración de datos mediante una réplica de lectura de Aurora

La migración desde una instancia de base de datos de RDS for PostgreSQL a un clúster de base de datos de Aurora PostgreSQL es un procedimiento que consta de varios pasos. En primer lugar, cree una réplica de lectura de Aurora de su instancia de base de datos de RDS for PostgreSQL de origen. Esto inicia un proceso de replicación desde la instancia de base de datos de RDS for PostgreSQL a un clúster de base de datos específico conocido como clúster de réplicas. El clúster de réplicas consiste únicamente en una réplica de lectura de Aurora (una instancia de lector).

> **Note**
>
> La migración puede tardar varias horas por tebibyte de datos en completarse.

Promoción de una réplica de Aurora PostgreSQL

Tras crear un clúster de base de datos de Aurora PostgreSQL, siga estos pasos para promocionar la réplica de Aurora:

1. Detenga toda la carga de trabajo de escritura de base de datos en la instancia de base de datos de RDS para PostgreSQL de origen.
2. Obtenga el `WAL LSN` actual de la instancia de base de datos de RDS para PostgreSQL de origen:

   ```
   SELECT pg_current_wal_lsn();
   pg_current_wal_lsn
   --------------------
   0/F0000318
   (1 row)
   ```

3. En el clúster de réplica de Aurora PostgreSQL, compruebe que el LSN reproducido sea mayor que el LSN del paso 2:

   ```
   SELECT pg_last_wal_replay_lsn();
   pg_last_wal_replay_lsn
   ------------------------
   0/F0000400
   (1 row)
   ```

   También puede usar la consulta siguiente en la instancia de base de datos de RDS para PostgreSQL de origen:

   ```
   SELECT restart_lsn FROM pg_replication_slots;
   ```

4. Promueva el clúster de réplica de Aurora PostgreSQL.

   La replicación se detiene, el clúster de réplica se convierte en un clúster de base de datos de Aurora PostgreSQL independiente y el lector se convierte en una instancia de escritor para el clúster. A partir de este momento, ya puede agregar instancias al clúster de base de datos de

Aurora PostgreSQL para adaptarlo a su caso de uso. Si ya no necesita la instancia de base de datos de RDS para PostgreSQL original, puede eliminarla.

No puede crear una réplica de lectura de Aurora si su instancia de base de datos de RDS for PostgreSQL ya tiene una réplica de lectura de Aurora o si tiene una réplica de lectura entre regiones.

Preparación para la migración de datos mediante una réplica de lectura de Aurora

> **Note**
>
> Al prepararse para migrar datos a Aurora PostgreSQL, es importante identificar y gestionar adecuadamente las tablas no registradas. Para obtener más información, consulte Gestión de tablas no registradas durante la migración.

Durante el proceso de migración mediante la réplica de lectura de Aurora, las actualizaciones que se efectúan en la instancia de base de datos de RDS for PostgreSQL de origen se replican de forma asíncrona en la réplica de lectura de Aurora del clúster de réplicas. El proceso utiliza la funcionalidad de replicación de streaming nativa de PostgreSQL, que almacena segmentos de registros de escritura anticipada (WAL) en la instancia de origen. Antes de iniciar este proceso de migración, asegúrese de que su instancia disponga de una capacidad de almacenamiento suficiente mediante la verificación de los valores de las métricas enumeradas en la tabla.

Métrica	Descripción
`FreeStorageSpace`	Espacio de almacenamiento disponible. Unidades: bytes
`OldestReplicationSlotLag`	Tamaño de retardo de datos de WAL en la replica con mayor retardo. Unidades: megabytes
`RDSToAuroraPostgreSQLReplicaLag`	Cantidad de tiempo en segundos de retardo de un clúster de base de datos de Aurora PostgreSQL con respecto a la instancia de base de datos de RDS de origen.

Métrica	Descripción
`TransactionLogsDiskUsage`	Espacio en disco utilizado por los logs de transacciones. Unidades: megabytes

Para obtener más información sobre la monitorización de una instancia de RDS, consulte Monitoring (Monitorización) en la Guía del usuario de Amazon RDS.

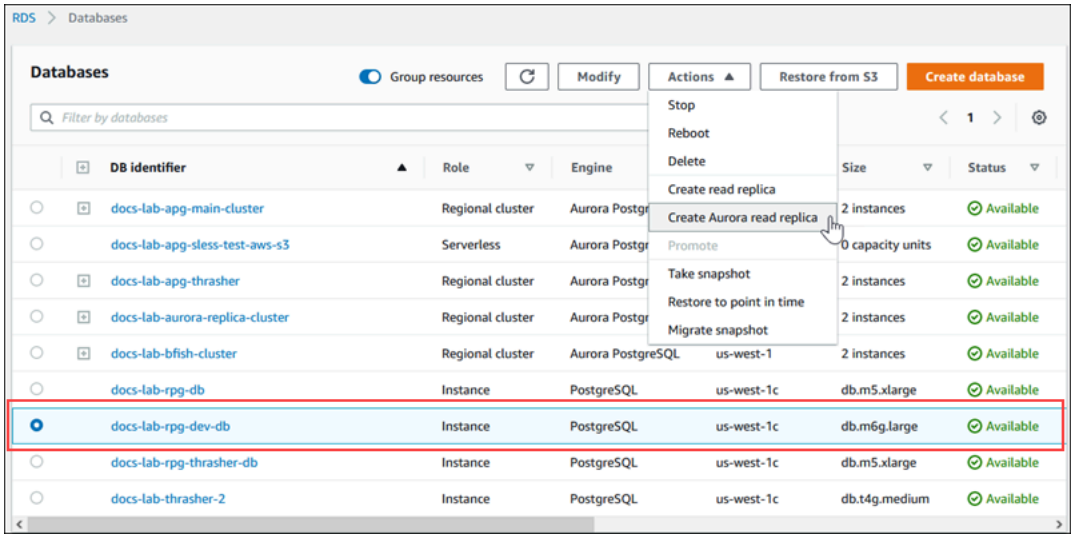
Creación de una réplica de lectura de Aurora

Puede crear una réplica de lectura de Aurora para una instancia de base de datos de RDS for PostgreSQL a través de la Consola de administración de AWS y la AWS CLI. La opción de crear una réplica de lectura de Aurora mediante la Consola de administración de AWS solo está disponible si la Región de AWS ofrece una versión compatible de Aurora PostgreSQL. Es decir, solo se encuentra disponible si existe una versión de Aurora PostgreSQL que sea la misma que la versión de RDS for PostgreSQL, o bien, que sea una versión secundaria superior de la misma familia de versiones principales.

Consola

Para crear una réplica de lectura de Aurora a partir de una instancia de base de datos PostgreSQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija la instancia de base de datos de RDS for PostgreSQL que quiera usar como origen para su réplica de lectura de Aurora. En Actions (Acciones), elija Create Aurora read replica (Crear réplica de lectura de Aurora). Si esta opción no se muestra, significa que una versión compatible de Aurora PostgreSQL no está disponible en la región.



4. En la página de configuración Create Aurora read replica (Crear réplica de lectura de Aurora), se configuran las propiedades del clúster de base de datos de Aurora PostgreSQL, tal como se muestra en la siguiente tabla. El clúster de la base de datos de réplica se crea a partir de una instantánea de la instancia de base de datos de origen con el mismo nombre de usuario y contraseña “maestros” que el origen, por lo que no puede cambiarlos en este momento.

Opción	Descripción
Clase de instancia de la base de datos	Elija una clase de instancia de base de datos que cumpla con los requisitos de procesamiento y memoria para la instancia principal del clúster de base de datos. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.
Multi-AZ deployment (Implementación Multi-AZ	No se encuentra disponible durante la migración
Identificador de la instancia de la base de datos	Ingrese el nombre que desea dar a la instancia de base de datos. Este identificador se utiliza en la dirección del punto de enlace de la instancia principal del nuevo clúster de base de datos. El identificador de instancias de bases de datos tiene las siguientes limitaciones:

Opción	Descripción
	• Debe incluir entre 1 y 63 caracteres alfanuméricos o guiones. • El primer carácter debe ser una letra. • No puede terminar con un guion ni contener dos guiones consecutivos. • Debe ser único para todas las instancias de base de datos de cada cuenta de AWS y para cada Región de AWS.
Virtual Private Cloud (VPC) (Nube virtual privada)	Elija la VPC que alojará el clúster de base de datos. Elija Create new VPC (Crear una VPC) para que Amazon RDS cree una VPC automáticamente. Para obtener más información, consulte Requisitos previos de clúster de base de datos.
Grupo de subred de base de datos	Elija el grupo de subredes de base de datos que desea utilizar para el clúster de base de datos. Elija Create new DB subnet group (Crear nuevo grupo de subredes de base de datos) para que Amazon RDS cree un grupo de subredes de base de datos. Para obtener más información, consulte Requisitos previos de clúster de base de datos.
Public accessibility (Accesibilidad pública	Elija Yes (Sí) para asignar al clúster de base de datos una dirección IP pública; de lo contrario, elija No. Las instancias en el clúster de base de datos pueden ser una combinación de instancias de base de datos tanto públicas como privadas. Para obtener más informaci ón acerca del modo de ocultar instancias al acceso público, consulte Cómo ocultar un clúster de base de datos en una VPC desde Internet..

Opción	Descripción
Availability zone	Determine si desea especificar una zona de disponibilidad concreta. Para obtener más información acerca de las zonas de disponibilidad, consulte Regiones y zonas de disponibilidad.
Grupos de seguridad de la VPC	Elija uno o varios grupos de seguridad de VPC para proteger el acceso de red al clúster de base de datos. Elija Create new VPC security group (Crear un grupo de seguridad de VPC) para que Amazon RDS cree un grupo de seguridad de VPC automáticamente. Para obtener más información, consulte Requisitos previos de clúster de base de datos.
Database port (Puerto de base de datos	Especifique el puerto que deben usar las aplicaciones y las utilidades para obtener acceso a la base de datos. Los clústeres de base de datos de Aurora PostgreSQL utilizan de forma predeterminada el puerto 5432 de PostgreSQL. Los firewalls de algunas compañías bloquean las conexiones a este puerto. Si el firewall de su compañía bloquea el puerto predeterminado, elija otro puerto para el nuevo clúster de base de datos.
Grupo de parámetros de base de datos	Elija un grupo de parámetros de base de datos para el clúster de base de datos de Aurora PostgreSQL. Aurora tiene un grupo de parámetros de base de datos predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros de base de datos. Para obtener más información acerca de los grupos de parámetros de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Opción	Descripción
Grupo de parámetros de clúster de base de datos	Elija un grupo de parámetros de clúster de base de datos para el clúster de base de datos de Aurora PostgreSQL. Aurora cuenta con un grupo de parámetros de clúster de base de datos predeterminado que puede utilizar. Si lo prefiere, puede crear su propio grupo de parámetros de clúster de base de datos. Para obtener más información acerca de los grupos de parámetros de clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora.
Cifrado	Elija Enable encryption (Habilitar cifrado) para que el nuevo clúster de base de datos de Aurora se cifre en reposo. Si elige Enable encryption (Habilitar cifrado), elija también una clave de KMS como valor de AWS KMS key.
Prioridad	Elija una prioridad de conmutación por error para el clúster de base de datos. Si no elije un valor, el ajuste predeterminado es tier-1. Esta prioridad determina el orden en que se promueven las réplicas de Aurora cuando el sistema se recupera de un error en la instancia principal. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.
Backup retention period	Elija el tiempo (entre 135 y 35 días) durante el que Aurora conservará las copias de seguridad de la base de datos. Los backups se pueden utilizar para las restauraciones a un momento dado (PITR) de la base de datos con una precisión de segundos.

Opción	Descripción
Enhanced monitoring (Supervisión mejorada	Elija Enable enhanced monitoring (Habilitar monitorización mejorada) a fin de habilitar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta su clúster de base de datos. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
Monitoring Role (Rol de supervisión	Solo está disponible si eligió Enable Enhanced Monitoring (Habilitar monitorización mejorada). El rol de AWS Identity and Access Management (IAM) que se debe usar para Enhanced Monitoring. Para obtener más información, consulte Configuración y habilitación del monitoreo mejorado.
Granularity (Grado de detalle	Solo está disponible si eligió Enable Enhanced Monitoring (Habilitar monitorización mejorada). Defina el intervalo, en segundos, en el que se recopilan las métricas para el clúster de base de datos.
Auto minor version upgrade (Actualización automática de versiones secundarias	Elija Yes (Sí) para habilitar el clúster de base de datos de Aurora PostgreSQL para recibir actualizaciones de las versiones secundarias del motor de base de datos PostgreSQL automáticamente cuando estén disponibles. La opción Auto Minor Version Upgrade (Actualización automática a versiones secundarias) solo es válida para las actualizaciones secundarias de las versiones del motor de PostgreSQL para su clúster de base de datos Aurora PostgreSQL. No tiene validez para los parches periódicos que se utilizan para mantener la estabilidad del sistema.
Periodo de mantenimiento	Elija el intervalo de tiempo semanal durante el que se puede llevar a cabo el mantenimiento del sistema.

5. Elija Create read replica (Crear réplica de lectura).

AWS CLI

Para crear una réplica de lectura de Aurora a partir de una instancia de base de datos de RDS for PostgreSQL de origen mediante la AWS CLI, use el comando de la CLI create-db-cluster para crear un clúster de base de datos de Aurora vacío. Cuando exista el clúster de base de datos, utilice el comando create-db-instance para crear la instancia principal del clúster de base de datos. La instancia principal es la primera instancia que se crea en un clúster de bases de datos de Aurora. En este caso, se crea inicialmente como una réplica de lectura de Aurora de su instancia de base de datos de RDS for PostgreSQL. Cuando concluya el proceso, la instancia de base de datos de RDS for PostgreSQL se habrá migrado de forma efectiva a un clúster de base de datos de Aurora PostgreSQL.

No tiene que especificar la cuenta de usuario principal (normalmente, `postgres`), su contraseña o el nombre de la base de datos. La réplica de lectura de Aurora las obtiene automáticamente a partir de la instancia de base de datos de RDS for PostgreSQL de origen que identifica al invocar los comandos AWS CLI.

Necesita especificar la versión del motor que se va a utilizar para el clúster de base de datos de Aurora PostgreSQL y la instancia de base de datos. La versión que especifique debe coincidir con la instancia de base de datos de RDS for PostgreSQL de origen. Si la instancia de base de datos de RDS for PostgreSQL de origen está cifrada, debe especificar también el cifrado para la instancia principal del clúster de base de datos de Aurora PostgreSQL. No se admite la migración de una instancia cifrada a un clúster de base de datos Aurora sin cifrar.

Los siguientes ejemplos crean un clúster de base de datos de Aurora PostgreSQL denominado `my-new-aurora-cluster` que va a utilizar una instancia de base de datos RDS de origen sin cifrar. En primer lugar, cree el clúster de base de datos de Aurora PostgreSQL llamando al comando `create-db-cluster`de la CLI. En el ejemplo se muestra cómo utilizar el parámetro `--storage-encrypted` opcional para especificar que el clúster de base de datos debe cifrarse. Debido a que la base de datos de origen no está cifrada, `--kms-key-id` se utiliza para especificar la clave que se debe usar. Para obtener más información sobre los parámetros obligatorios y opcionales, consulte la lista que sigue al ejemplo.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster  \
```

```
    --db-cluster-identifier my-new-aurora-cluster \
    --db-subnet-group-name my-db-subnet
    --vpc-security-group-ids sg-11111111
    --engine aurora-postgresql \
    --engine-version same-as-your-rds-instance-version \
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db \
    --storage-encrypted \
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Para Windows:

```
aws rds create-db-cluster  ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-subnet-group-name my-db-subnet ^
    --vpc-security-group-ids sg-11111111 ^
    --engine aurora-postgresql ^
    --engine-version same-as-your-rds-instance-version ^
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db ^
    --storage-encrypted ^
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

En la siguiente lista encontrará más información sobre algunas de las opciones que se muestran en el ejemplo. A menos que se especifique lo contrario, estos parámetros son obligatorios.

- `--db-cluster-identifier`: debe asignar un nombre a su nuevo clúster de base de datos de Aurora PostgreSQL.
- `--db-subnet-group-name`: cree su clúster de base de datos de Aurora PostgreSQL en la misma subred de base de datos que la instancia de base de datos de origen.
- `--vpc-security-group-ids`: especifique el grupo de seguridad del clúster de base de datos de Aurora PostgreSQL.
- `--engine-version`: especifique la versión que se va a utilizar para el clúster de base de datos de Aurora PostgreSQL. Debe ser la misma versión secundaria que la que utiliza la instancia de base de datos RDS para PostgreSQL de origen o una superior.
- `--replication-source-identifier`: identifique la instancia de base de datos de RDS for PostgreSQL con su nombre de recurso de Amazon (ARN). Para obtener más información sobre

los ARN de Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS) en la Referencia general de AWS de su clúster de base de datos.

- `--storage-encrypted`: opcional. Úsela solo cuando sea necesario para especificar el cifrado de la siguiente manera:
  - Utilice este parámetro cuando la instancia de base de datos de origen tenga almacenamiento cifrado. La llamada a `create-db-cluster` falla si no utiliza este parámetro con una instancia de base de datos de origen que tenga almacenamiento cifrado. Si desea utilizar una clave distinta para el clúster de base de datos de Aurora PostgreSQL a la clave utilizada por la instancia de base de datos de origen, debe especificar también `--kms-key-id`.
  - Utilice este parámetro si el almacenamiento de la instancia de base de datos de origen no está cifrado pero desea que el clúster de base de datos de Aurora PostgreSQL utilice el cifrado. Si es así, también deberá identificar la clave de cifrado que se va a utilizar con el parámetro `--kms-key-id`.
- `--kms-key-id`: opcional. Si se utiliza, puede especificar la clave que se va a utilizar para el cifrado de almacenamiento (`--storage-encrypted`) mediante el ARN, el ID, el alias ARN o el nombre de alias de la clave. Este parámetro solo es necesario para las siguientes situaciones:
  - Para elegir una clave distinta para el clúster de base de datos de Aurora PostgreSQL a la utilizada por la instancia de base de datos de origen.
  - Para crear un clúster cifrado a partir de un origen sin cifrar. En este caso, debe especificar la clave que Aurora PostgreSQL debe utilizar para el cifrado.

Después de crear el clúster de base de datos de Aurora PostgreSQL, cree la instancia principal mediante el comando de la CLI `create-db-instance`, tal y como se muestra en la siguiente:

Para Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-instance-class db.x2g.16xlarge \
    --db-instance-identifier rpg-for-migration \
    --engine aurora-postgresql
```

Para Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-instance-class db.x2g.16xlarge ^
```

```
    --db-instance-identifier rpg-for-migration ^
    --engine aurora-postgresql
```

En la siguiente lista encontrará más información sobre algunas de las opciones que se muestran en el ejemplo.

- `--db-cluster-identifier`: especifique el nombre del nuevo clúster de base de datos de Aurora PostgreSQL que creó en el paso anterior con el comando `create-db-instance`.
- `--db-instance-class`: nombre de la clase de instancia de base de datos que se va a utilizar para la instancia principal, como `db.r4.xlarge`, `db.t4g.medium`, `db.x2g.16xlarge`, etc. Para obtener una lista de las clases de instancia de base de datos disponibles, consulte Tipos de clase de instancia de base de datos.
- `--db-instance-identifier`: especifique el nombre que desea asignar a la instancia principal.
- `--engine aurora-postgresql`: especifique `aurora-postgresql` para el motor.

API de RDS

Para crear una réplica de lectura de Aurora a partir de una instancia de base de datos de RDS for PostgreSQL de origen, utilice primero la operación de la API de RDS `CreateDBCluster` para crear un nuevo clúster de base de datos de Aurora para la réplica de lectura de Aurora que se crea a partir de su instancia de base de datos de RDS for PostgreSQL de origen. Cuando el clúster de base de datos de Aurora PostgreSQL esté disponible, debe utilizar la `CreateDBInstance` para crear la instancia principal del clúster de base de datos de Aurora.

No tiene que especificar la cuenta de usuario principal (normalmente `postgres`), su contraseña o el nombre de la base de datos. La réplica de lectura de Aurora las obtiene automáticamente a partir de la instancia de base de datos de RDS for PostgreSQL de origen especificada con `ReplicationSourceIdentifier`.

Necesita especificar la versión del motor que se va a utilizar para el clúster de base de datos de Aurora PostgreSQL y la instancia de base de datos. La versión que especifique debe coincidir con la instancia de base de datos de RDS for PostgreSQL de origen. Si la instancia de base de datos de RDS for PostgreSQL de origen está cifrada, debe especificar también el cifrado para la instancia principal del clúster de base de datos de Aurora PostgreSQL. No se admite la migración de una instancia cifrada a un clúster de base de datos de Aurora sin cifrar.

Para crear el clúster de base de datos de Aurora para la réplica de lectura de Aurora, utilice la operación de la API de RDS `CreateDBCluster` con los siguientes parámetros:

- `DBClusterIdentifier`: nombre del clúster de base de datos que se creará.
- `DBSubnetGroupName`: nombre del grupo de subredes de la base de datos que desea asociar con este clúster de base de datos.
- `Engine=aurora-postgresql`: nombre del motor que se debe utilizar.
- `ReplicationSourceIdentifier`: nombre de recurso de Amazon (ARN) de la instancia de base de datos de PostgreSQL de origen. Para obtener más información sobre los ARN de Amazon RDS, consulte Amazon Relational Database Service (Amazon RDS) en la Referencia general de Amazon Web Services. Si `ReplicationSourceIdentifier` identifica un origen cifrado, Amazon RDS utiliza la clave de KMS predeterminada a menos que especifique otra clave mediante la opción `KmsKeyId`.
- `VpcSecurityGroupIds`: lista de grupos de seguridad de VPC de Amazon EC2 que asociar a este clúster de base de datos.
- `StorageEncrypted`: indica si el clúster de base de datos está cifrado. Cuando utiliza este parámetro sin especificar también `ReplicationSourceIdentifier`, Amazon RDS utiliza la clave de KMS predeterminada.
- `KmsKeyId`: clave de un clúster cifrado. Si se utiliza, puede especificar la clave que se va a utilizar para el cifrado de almacenamiento mediante el ARN, el ID, el ARN de alias o el nombre de alias de la clave.

Para obtener más información, consulte `CreateDBCluster` en la Referencia de la API de Amazon RDS.

Cuando el clúster de base de datos de Aurora esté disponible, puede crear una instancia principal para él con la operación `CreateDBInstance` de la API de RDS y los siguientes parámetros:

- `DBClusterIdentifier`: nombre del clúster de base de datos.
- `DBInstanceClass`: nombre de la clase de instancia de base de datos que se va a utilizar para la instancia principal.
- `DBInstanceIdentifier`: nombre de la instancia principal.
- `Engine=aurora-postgresql`: nombre del motor que se debe utilizar.

Para obtener más información, consulte `CreateDBInstance` en la Referencia de la API de Amazon RDS.

Promoción de una réplica de lectura de Aurora

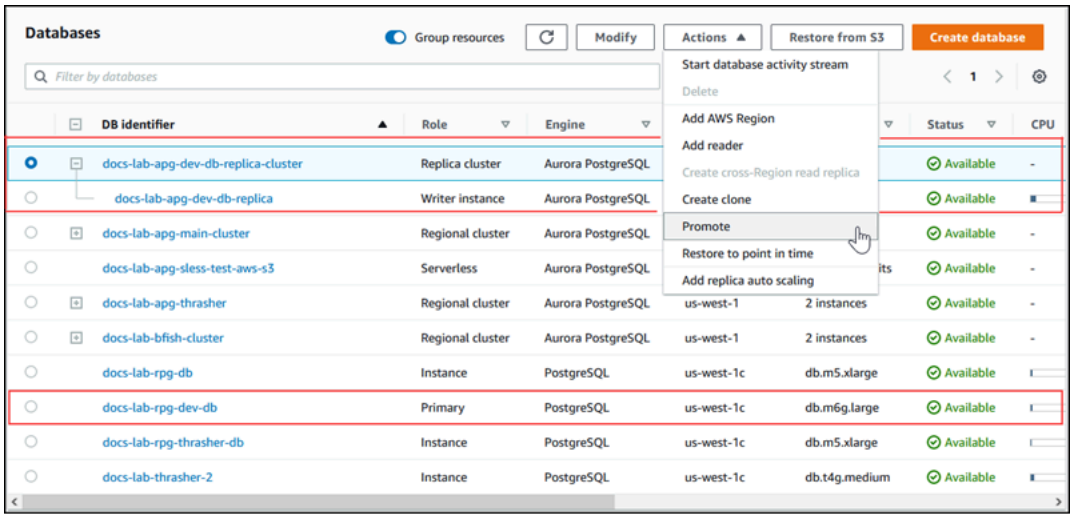
La migración a Aurora PostgreSQL no está completa hasta que se promocione el clúster de réplicas, así que todavía no elimine la instancia de base de datos de origen de RDS for PostgreSQL.

Antes de promocionar el clúster de réplicas, asegúrese de que la instancia de base de datos de RDS for PostgreSQL no tenga transacciones en proceso ni escritura de otra actividad en la base de datos. Cuando el retraso de réplica en la réplica de lectura de Aurora llegue a cero (0), puede promocionar el clúster de réplicas. Para obtener más información sobre la supervisión del retraso de réplica, consulte Monitoreo de replicación de Aurora PostgreSQL y Métricas de nivel de instancia para Amazon Aurora.

Consola

Para promover una réplica de lectura de Aurora a un clúster de base de datos Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster de réplicas.



4. En Actions (Acciones), seleccione Promote (Promover). Esto puede tardar unos minutos y causar tiempo de inactividad.

Cuando se completa el proceso, el clúster de réplica de Aurora es un clúster de base de datos regional de Aurora PostgreSQL, con una instancia de escritor que contiene los datos de la instancia de base de datos de RDS for PostgreSQL.

AWS CLI

Para promocionar una réplica de lectura de Aurora a un clúster de base de datos independiente, utilice el comando promote-read-replica-db-cluster de la AWS CLI.

Example

Para Linux, macOS o Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Para Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

API de RDS

Para promover una réplica de Aurora lectura a un clúster de base de datos independiente, utilice la operación de API de RDS PromoteAdReplicadbCluster .

Después de promocionar el clúster de réplicas, puede confirmar que la promoción se ha completado mediante la revisión del registro de eventos de la siguiente manera.

Para confirmar que se ha promocionado el clúster de réplicas de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Events.
3. En la página Events (Eventos), busque el nombre de su clúster en la lista Source (Origen). Cada evento tiene un origen, un tipo, una hora y un mensaje. Puede ver todos los eventos que se han producido en su Región de AWS para su cuenta. Una promoción exitosa genera el siguiente mensaje.

   ```
   Promoted Read Replica cluster to a stand-alone database cluster.
   ```

Una vez que haya finalizado la promoción, la instancia de base de datos de RDS for PostgreSQL de origen y el clúster de base de datos de Aurora se desvinculan. Puede dirigir sus aplicaciones

de cliente al punto de conexión para la réplica de lectura de Aurora. Para obtener más información acerca de los puntos de enlace de Aurora, consulte Conexiones de puntos de conexión de Amazon Aurora. En ese momento, podrá eliminar de forma segura la instancia de base de datos.

Optimización del rendimiento de las consultas en Aurora PostgreSQL

Optimizar el rendimiento de las consultas es fundamental porque ayuda a que las bases de datos se ejecuten de forma más rápida y eficiente, a la vez que se utilizan menos recursos, lo que se traduce en una mejor experiencia de usuario y en menores costos operativos. Amazon Aurora PostgreSQL ofrece varias características que ayudan a optimizar el rendimiento de las consultas para las cargas de trabajo de PostgreSQL.

Temas

- Mejora del rendimiento de las consultas de Aurora PostgreSQL con lecturas optimizadas de Aurora
- Optimización de subconsultas correlacionadas en Aurora PostgreSQL
- Mejora del rendimiento de las consultas mediante la unión adaptativa

Mejora del rendimiento de las consultas de Aurora PostgreSQL con lecturas optimizadas de Aurora

Puede conseguir un procesamiento de consultas más rápido para Aurora PostgreSQL con las lecturas optimizadas de Aurora. Una instancia de la base de datos Aurora PostgreSQL que utiliza las lecturas optimizadas de Aurora ofrece una reducción de la latencia de las consultas de hasta 8 veces y un ahorro de costes de hasta el 30 % para aplicaciones con conjuntos de datos de gran tamaño que superan la capacidad de memoria de una instancia de base de datos.

Temas

- Información general de las lecturas optimizadas de Aurora en PostgreSQL
- Uso de lecturas optimizadas de Aurora
- Casos de uso de lecturas optimizadas de Aurora
- Supervisión de instancias de base de datos que utilizan lecturas optimizadas de Aurora
- Prácticas recomendadas para lecturas optimizadas de Aurora

Información general de las lecturas optimizadas de Aurora en PostgreSQL

Las lecturas optimizadas de Aurora están disponibles de forma predeterminada cuando se crea un clúster de base de datos que utiliza instancias R6gd basadas en Graviton, R8gd e instancias R6id basadas en Intel con almacenamiento de memoria no volátil exprés (NVMe). Está disponible a partir de las siguientes versiones de PostgreSQL:

- versiones 14.12 y superiores, 15.7 y versiones superiores, 16.3 y versiones superiores, 17.4 y versiones superiores para instancias R8gd
- versiones 14.9 y superiores, 15.4 y versiones superiores, 16.1 y versiones superiores para instancias R6gd y R6id

Las lecturas optimizadas de Aurora admiten dos capacidades: caché por niveles y objetos temporales.

Caché por niveles habilitada para lecturas optimizadas: con la caché por niveles, puede multiplicar por 5 la capacidad de almacenamiento en caché de la instancia de base de datos. De este modo, la caché se mantiene automáticamente con los datos más recientes y coherentes a nivel de transacción, lo que libera a las aplicaciones de la sobrecarga que supone administrar la vigencia de los datos en soluciones de almacenamiento en caché externas basadas en conjuntos de resultados. Ofrece una latencia hasta 8 veces mejor para las consultas que anteriormente obtenían datos del almacenamiento de Aurora.

En Aurora, el valor de `shared_buffers` en el grupo de parámetros predeterminado suele estar establecido en torno al 75 % de la memoria disponible. Sin embargo, para los tipos de instancia r8gd, r6gd y r6id, Aurora reduce el espacio de `shared_buffers` un 4,5 % para alojar los metadatos de la caché de lecturas optimizadas.

Objetos temporales habilitados para lecturas optimizadas: mediante el uso de objetos temporales, puede lograr un procesamiento de consultas más rápido si ubica los archivos temporales generados por PostgreSQL en el almacenamiento NVMe local. Así se reduce el tráfico hacia Elastic Block Storage (EBS) a través de la red. Ofrece una latencia y un rendimiento hasta dos veces mejores para consultas avanzadas que ordenan, unen o fusionan grandes volúmenes de datos que no caben en la capacidad de memoria disponible en una instancia de base de datos.

En un clúster optimizado para E/S de Aurora, las lecturas optimizadas utilizan tanto la caché por niveles como los objetos temporales del almacenamiento NVMe. Con la función de caché por niveles habilitada para lecturas optimizadas, Aurora asigna el doble de la memoria de instancia para

objetos temporales, aproximadamente el 10 % del almacenamiento para operaciones internas y el almacenamiento restante como caché por niveles. En un clúster estándar de Aurora, las lecturas optimizadas utilizan únicamente objetos temporales.

Los clústeres optimizados para E/S de Aurora le permiten cambiar el tamaño del espacio asignado para los objetos temporales habilitados para lecturas optimizados mediante el parámetro dinámico `aurora_temp_space_size` en el nivel de instancia. Esta característica de cambio de tamaño disponible a partir de las siguientes versiones de PostgreSQL:

- Versión 16.8 y todas las versiones posteriores
- Versión 15.12 y otras 15 versiones posteriores
- Versión 14.17 y otras 14 versiones posteriores

Con este parámetro, puede redimensionar la capacidad de 2 a 6 veces la memoria de la instancia sin necesidad de reiniciar el motor de base de datos. Cuando expande el espacio de objetos temporales, el cambio surte efecto inmediatamente, independientemente de las cargas de trabajo concurrentes. Sin embargo, al reducir el espacio, el ajuste solo se completa cuando hay suficiente espacio no utilizado en los objetos temporales para adaptarse a la nueva solicitud de tamaño. Tras cambiar el tamaño de los objetos temporales habilitados para lecturas optimizadas, la caché por niveles se ajusta automáticamente para utilizar cualquier espacio disponible.

Motor	Configuración de almacenamiento en clústeres	Objetos temporales habilitados para lecturas optimizadas	Caché por niveles optimizada y habilitada para las lecturas optimizadas	Versiones admitidas
Aurora PostgreSQL-Compatible Edition	Estándar	Sí	No	• versiones 14.9 y superiores, 15.4 y versiones superiores, 16.1 y todas las versiones superiores para instancias r6gd y r6id
	Optimización de E/S	Sí	Sí	

Motor	Configuración de almacenamiento en clústeres	Objetos temporales habilitados para lecturas optimizadas	Caché por niveles optimizada y habilitada para las lecturas optimizadas	Versiones admitidas
				• versiones 14.12 y superiores, 15.7 y versiones superiores, 16.3 y versiones superiores, 17.4 y versiones superiores para instancias r8gd

> **Note**
>
> Al cambiar entre clústeres estándar y optimizados para E/S en una clase de instancia de base de datos basada en NVMe, se reinicia inmediatamente el motor de base de datos.

En Aurora PostgreSQL, utilice el parámetro `temp_tablespaces` para configurar el espacio de tabla donde se almacenan los objetos temporales.

Para comprobar si los objetos temporales están configurados, utilice el siguiente comando:

```
postgres=> show temp_tablespaces;
temp_tablespaces
---------------------
aurora_temp_tablespace
(1 row)
```

`aurora_temp_tablespace` es un espacio de tabla configurado por Aurora que apunta hacia el almacenamiento local de NVMe. No puede modificar este parámetro ni volver al almacenamiento de Amazon EBS.

Para comprobar si la caché de lecturas optimizada está activada, utilice el siguiente comando:

```
postgres=> show shared_preload_libraries;
                shared_preload_libraries
-------------------------------------------------------
rdsutils,pg_stat_statements,aurora_optimized_reads_cache
```

Uso de lecturas optimizadas de Aurora

Al aprovisionar una instancia de base de datos de Aurora PostgreSQL con instancias de base de datos basadas en NVMe, la instancia de base de datos utiliza automáticamente lecturas optimizadas de Aurora.

Para activar las lecturas optimizadas de Aurora, realice una de las siguientes acciones:

- Para crear un clúster de base de datos de Aurora PostgreSQL, utilice una de las clases de instancia de base de datos basadas en NVMe. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.
- Modifique un clúster de base de datos de Aurora PostgreSQL para utilizar una de las clases de instancia de base de datos basadas en NVMe. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Las lecturas optimizadas de Aurora están disponibles en todas las Regiones de AWS donde se admite una o más de las clases de instancia de base de datos con almacenamiento SSD NVMe local. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.

Para volver a una instancia de Aurora con lecturas no optimizadas, modifique la clase de instancia de base de datos de su instancia de Aurora por una clase de instancia similar sin almacenamiento efímero de NVMe para sus cargas de trabajo de base de datos. Por ejemplo, si la clase de instancia de base de datos actual es db.r6gd.4xlarge, elija db.r6g.4xlarge para volver atrás. Para obtener más información, consulte Modificación de una instancia de base de datos de Aurora.

Casos de uso de lecturas optimizadas de Aurora

Caché por niveles optimizada y habilitada para las lecturas optimizadas

Estos son algunos casos de uso que pueden beneficiarse de las lecturas optimizadas con caché por niveles:

- Aplicaciones de Internet como procesamiento de pagos, facturación, comercio electrónico con estrictos SLA de rendimiento.
- Paneles de informes en tiempo real que ejecutan cientos de consultas puntuales para recopilar métricas o datos.
- Aplicaciones de IA generativa con la extensión pgvector para buscar vecinos exactos o más cercanos entre millones de incrustaciones vectoriales.

Objetos temporales habilitados para lecturas optimizadas

Estos son algunos casos de uso que pueden beneficiarse de las lecturas optimizadas con objetos temporales:

- Consultas analíticas que incluyen expresiones comunes de tabla (CTE), tablas derivadas y operaciones de agrupación.
- Réplicas de lectura que gestionan consultas no optimizadas de una aplicación.
- Consultas de informes dinámicas o bajo demanda con operaciones complejas, como GROUP BY y ORDER BY, que no siempre pueden utilizar los índices adecuados.
- Operaciones `CREATE INDEX` o `REINDEX` de ordenación.
- Otras cargas de trabajo que utilizan tablas temporales internas.

Supervisión de instancias de base de datos que utilizan lecturas optimizadas de Aurora

Puede supervisar las consultas que utilizan la caché por niveles habilitada para lecturas optimizadas con el comando EXPLAIN, como se muestra en el siguiente ejemplo:

```
Postgres=> EXPLAIN (ANALYZE, BUFFERS) SELECT c FROM sbtest15 WHERE id=100000000
```

```
QUERY PLAN
-------------------------------------------------------------------------------------
 Index Scan using sbtest15_pkey on sbtest15  (cost=0.57..8.59 rows=1 width=121) (actual
 time=0.287..0.288 rows=1 loops=1)
   Index Cond: (id = 100000000)
   Buffers: shared hit=3 read=2 aurora_orcache_hit=2
   I/O Timings: shared/local read=0.264
 Planning:
   Buffers: shared hit=33 read=6 aurora_orcache_hit=6
   I/O Timings: shared/local read=0.607
 Planning Time: 0.929 ms
 Execution Time: 0.303 ms
(9 rows)
Time: 2.028 ms
```

> **Note**
>
> Los campos `aurora_orcache_hit` y `aurora_storage_read` de la sección `Buffers` del plan explicativo solo se muestran cuando las lecturas optimizadas están activadas y sus valores son superiores a cero. El campo de lectura es el total de los campos `aurora_orcache_hit` y `aurora_storage_read`.

Puede supervisar las instancias de base de datos que utilizan lecturas optimizadas de Aurora con las siguientes métricas de CloudWatch:

- `AuroraOptimizedReadsCacheHitRatio`
- `FreeEphemeralStorage`
- `ReadIOPSEphemeralStorage`
- `ReadLatencyEphemeralStorage`
- `ReadThroughputEphemeralStorage`
- `WriteIOPSEphemeralStorage`
- `WriteLatencyEphemeralStorage`
- `WriteThroughputEphemeralStorage`

Estas métricas proporcionan datos sobre el almacén de instancias disponible, las IOPS y el rendimiento. Para obtener más información acerca de estas métricas, consulte Métricas de nivel de instancia para Amazon Aurora.

También puede utilizar la extensión `pg_proctab` para supervisar el almacenamiento NVMe.

```
postgres=>select * from pg_diskusage();

major | minor |       devname       | reads_completed | reads_merged | sectors_read |
 readtime | writes_completed | writes_merged | sectors_written | writetime | current_io
 | iotime  | totaliotime
------+-------+---------------------+-----------------+--------------+--------------
+----------+------------------+---------------+-----------------+-----------
+------------+---------+-------------
      |       | rdstemp             |           23264 |            0 |       191450 |
  11670 |          1750892 |             0 |        24540576 |    819350 |          0 |
 3847580 |      831020
      |       | rdsephemeralstorage |           23271 |            0 |       193098 |
   2620 |           114961 |             0 |        13845120 |    130770 |          0 |
  215010 |      133410
(2 rows)
```

Prácticas recomendadas para lecturas optimizadas de Aurora

Utilice estas prácticas recomendadas para las lecturas optimizadas de Aurora:

- Supervise el espacio de almacenamiento disponible en el almacén de instancias con la métrica de CloudWatch `FreeEphemeralStorage`. Si el almacén de instancias alcanza su límite debido a la carga de trabajo de la instancia de base de datos, ajuste la simultaneidad y las consultas que utilicen demasiados objetos temporales o modifíquelas para utilizar una clase de instancia de base de datos más grande.
- Supervise la métrica de CloudWatch para conocer la tasa de aciertos de caché de lecturas optimizadas. Operaciones como VACUUM modifican grandes cantidades de bloques con gran rapidez. Esto puede provocar una caída temporal en la tasa de aciertos. La extensión `pg_prewarm` se puede utilizar para cargar datos en la caché del búfer que permite a Aurora escribir algunos de estos bloques directamente en la caché de lecturas optimizadas.
- Puede habilitar la administración de la caché de clúster (CCM) para calentar la caché del búfer y la caché por niveles en un lector de nivel 0, que se utilizará como destino de conmutación por error. Si la CCM está activada, la caché del buffer se escanea periódicamente para escribir páginas que puedan eliminarse en la caché por niveles. Para obtener más información sobre CCM, consulte

Recuperación rápida después de una conmutación por error con la administración de caché del clúster para Aurora PostgreSQL.

Optimización de subconsultas correlacionadas en Aurora PostgreSQL

Una subconsulta correlacionada hace referencia a las columnas de la tabla desde la consulta externa. Se evalúa una vez por cada fila devuelta por la consulta externa. En el siguiente ejemplo, la subconsulta hace referencia a una columna de la tabla ot. Esta tabla no está incluida en la cláusula FROM de la subconsulta, pero se hace referencia a ella en la cláusula FROM de la consulta externa. Si la tabla ot tiene 1 millón de filas, la subconsulta debe evaluarse 1 millón de veces.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Note

- La transformación de subconsultas y la caché de subconsultas están disponibles en Aurora PostgreSQL a partir de la versión 16.8, mientras que Babelfish para Aurora PostgreSQL admite estas características desde la versión 4.2.0.
- A partir de las versiones 4.6.0 y 5.2.0 de Babelfish para Aurora PostgreSQL, estas características se controlan mediante los siguientes parámetros:
  - babelfishpg_tsql.apg_enable_correlated_scalar_transform
  - babelfishpg_tsql.apg_enable_subquery_cache

  De forma predeterminada, ambos parámetros están activados.

Mejora del rendimiento de las consultas de Aurora PostgreSQL mediante la transformación de subconsultas

Aurora PostgreSQL puede acelerar las subconsultas correlacionadas transformándolas en uniones externas equivalentes. Esta optimización se aplica a los dos tipos siguientes de subconsultas correlacionadas:

- Subconsultas que devuelven un único valor agregado y aparecen en la lista SELECT.

```
SELECT ot.a, ot.b, (SELECT AVG(it.b) FROM it WHERE it.a = ot.a) FROM ot;
```

- Subconsultas que devuelven un único valor agregado y aparecen en una cláusula WHERE.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Habilitar la transformación en la subconsulta

Para habilitar la transformación de subconsultas correlacionadas en uniones externas equivalentes, defina el parámetro `apg_enable_correlated_scalar_transform` en `ON`. El valor predeterminado de este parámetro es `OFF`.

Puede modificar el grupo de parámetros de instancia o clúster para establecer los parámetros. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.

De forma alternativa, puede configurar los ajustes solo para la sesión actual mediante el siguiente comando:

```
SET apg_enable_correlated_scalar_transform TO ON;
```

Verificación de transformaciones

Utilice el comando EXPLAIN para comprobar si la subconsulta correlacionada se ha transformado en una combinación externa en el plan de consultas.

Cuando la transformación esté habilitada, la parte de la subconsulta correlacionada aplicable se transformará en una combinación externa. Por ejemplo:

```
postgres=> CREATE TABLE ot (a INT, b INT);
CREATE TABLE
postgres=> CREATE TABLE it (a INT, b INT);
CREATE TABLE

postgres=> SET apg_enable_correlated_scalar_transform TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);

                        QUERY PLAN
------------------------------------------------------------
 Hash Join
   Hash Cond: (ot.a = apg_scalar_subquery.scalar_output)
```

```
   Join Filter: ((ot.b)::numeric < apg_scalar_subquery.avg)
   ->  Seq Scan on ot
   ->  Hash
         ->  Subquery Scan on apg_scalar_subquery
               ->  HashAggregate
                     Group Key: it.a
                     ->  Seq Scan on it
```

La misma consulta no se transforma cuando el parámetro GUC pasa a `OFF`. El plan no tendrá una combinación externa, sino un subplan.

```
postgres=> SET apg_enable_correlated_scalar_transform TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);
                QUERY PLAN
---------------------------------------
 Seq Scan on ot
   Filter: ((b)::numeric < (SubPlan 1))
   SubPlan 1
     ->  Aggregate
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Limitaciones

- La subconsulta debe estar en la lista SELECT o en una de las condiciones de la cláusula WHERE. De lo contrario, no se transformará.
- La subconsulta debe devolver una función agregada. Las funciones de agregado definidas por el usuario no se admiten para la transformación.
- Una subconsulta cuya expresión de devolución no sea una simple función agregada no se transformará.
- La condición correlacionada en las cláusulas WHERE de la subconsulta debe ser una referencia de columna simple. De lo contrario, no se transformará.
- La condición correlacionada en las cláusulas WHERE de la subconsulta debe ser un predicado de igualdad simple.
- La subconsulta no puede contener una cláusula HAVING o una cláusula GROUP BY.
- La cláusula WHERE de la subconsulta puede contener uno o más predicados combinados con AND.

> **Note**
>
> El impacto de la transformación en el rendimiento varía según el esquema, los datos y la carga de trabajo. La ejecución de subconsultas correlacionada con la transformación puede mejorar significativamente el rendimiento a medida que aumenta el número de filas generadas por la consulta externa. Le recomendamos encarecidamente que pruebe esta característica en un entorno que no sea de producción con su esquema, datos y carga de trabajo reales antes de habilitarla en un entorno de producción.

Uso de la caché de subconsultas para mejorar el rendimiento de las consultas de Aurora PostgreSQL

Aurora PostgreSQL admite la caché de subconsultas para almacenar los resultados de las subconsultas correlacionadas. Esta característica omite las ejecuciones repetidas de subconsultas correlacionadas cuando los resultados de las subconsultas ya están en la memoria caché.

La caché de subconsultas

El nodo Memoize de PostgreSQL es la parte clave de la caché de subconsultas. El nodo Memoize mantiene una tabla hash en la caché local para hacer asignaciones desde los valores de los parámetros de entrada a las filas de resultados de las consultas. El límite de memoria de la tabla hash es el producto de work_mem y hash_mem_multiplier. Para obtener más información, consulte Resource Consumption.

Durante la ejecución de la consulta, la caché de subconsultas utiliza la tasa de aciertos de caché (CHR) para estimar si la caché mejora el rendimiento de las consultas y para decidir, en el tiempo de ejecución de la consulta, si se debe seguir utilizando la caché. La CHR es la relación entre el número de visitas a caché y el número total de solicitudes. Por ejemplo, si una subconsulta correlacionada debe ejecutarse 100 veces y 70 de esos resultados de ejecución se pueden recuperar de la memoria caché, la CHR es 0,7.

Por cada número apg_subquery_cache_check_interval de errores de caché, se evalúan las ventajas de la caché de subconsultas comprobando si la CHR es mayor que apg_subquery_cache_hit_rate_threshold. De lo contrario, la caché se borrará de la memoria y la ejecución de la consulta volverá a la ejecución original (reejecución de subconsulta sin caché).

Parámetros que controlan el comportamiento de la caché de subconsultas

En la siguiente tabla se muestran los parámetros que controlan el comportamiento de la caché de subconsultas.

Parámetro	Descripción	Predeterminado	Permitido
apg_enable_subquery_cache	Permite el uso de la memoria caché para subconsultas escalares correlacionadas.	OFF	ON, OFF
apg_subquery_cache_check_interval	Establece la frecuencia, en número de errores de caché, para evaluar la tasa de aciertos de caché de subconsultas.	500	0–2147483647
apg_subquery_cache_hit_rate_threshold	Establece el umbral de la tasa de aciertos de caché de subconsultas.	0.3	0,0-1,0

Note

- Los valores más altos de `apg_subquery_cache_check_interval` pueden mejorar la precisión de la estimación de los beneficios de la memoria caché según la CHR, pero aumentarán la sobrecarga de la memoria caché, ya que la CHR no se evaluará hasta que la tabla caché tenga un número `apg_subquery_cache_check_interval` de filas.
- Los valores altos de `apg_subquery_cache_hit_rate_threshold` sugieren el abandono de la caché de subconsultas y la vuelta a la opción original (reejecución de subconsultas sin caché).

Puede modificar el grupo de parámetros de instancia o clúster para establecer los parámetros. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.

De forma alternativa, puede configurar los ajustes solo para la sesión actual mediante el siguiente comando:

```
SET apg_enable_subquery_cache TO ON;
```

Activación de la caché de subconsultas en Aurora PostgreSQL

Cuando la caché de subconsultas está habilitada, Aurora PostgreSQL aplica la caché para guardar los resultados de las subconsultas. El plan de consultas tendrá entonces un nodo Memoize en SubPlan.

Por ejemplo, la siguiente secuencia de comandos muestra el plan de ejecución de consultas estimado de una subconsulta correlacionada simple sin caché de subconsultas.

```
postgres=> SET apg_enable_subquery_cache TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Seq Scan on it
           Filter: (a = ot.a)
```

Tras activar `apg_enable_subquery_cache`, el plan de consultas incluirá un nodo Memoize bajo el nodo SubPlan, lo que indica que la subconsulta planea usar la memoria caché.

```
postgres=> SET apg_enable_subquery_cache TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
```

```
   SubPlan 1
     ->  Memoize
           Cache Key: ot.a
           Cache Mode: binary
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

El plan de ejecución de consultas actual contiene más detalles de la caché de subconsultas, incluidos los aciertos y los errores de caché. El siguiente resultado muestra el plan real de ejecución de la consulta del ejemplo anterior tras insertar algunos valores en las tablas.

```
postgres=> EXPLAIN (COSTS FALSE, TIMING FALSE, ANALYZE TRUE) SELECT ot.a, ot.b FROM ot
 WHERE ot.b < (SELECT it.b FROM it WHERE it.a = ot.a);
            QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on ot (actual rows=2 loops=1)
   Filter: (b < (SubPlan 1))
   Rows Removed by Filter: 8
   SubPlan 1
     ->  Memoize (actual rows=0 loops=10)
           Cache Key: ot.a
           Cache Mode: binary
           Hits: 4  Misses: 6  Evictions: 0  Overflows: 0  Memory Usage: 1kB
           ->  Seq Scan on it (actual rows=0 loops=6)
                 Filter: (a = ot.a)
                 Rows Removed by Filter: 4
```

El número total de aciertos de caché es 4 y el número total de errores de caché es 6. Si el número total de aciertos y errores es inferior al número de bucles del nodo Memoize, la evaluación de la CHR no se ha aprobado, y la caché se ha borrado y abandonado en algún momento. A continuación, la ejecución de la subconsulta ha vuelto a la reejecución original sin caché.

Limitaciones

La caché de subconsultas no admite ciertos patrones de subconsultas correlacionadas. Estos tipos de consultas se ejecutarán sin caché, aunque la caché de subconsultas esté activada:

- Subconsultas correlacionadas IN/EXISTS/ANY/ALL
- Subconsultas correlacionadas que contienen funciones no deterministas.
- Subconsultas correlacionadas que hacen referencia a columnas de tablas externas con tipos de datos que no admiten operaciones de hash o igualdad.

Mejora del rendimiento de las consultas mediante la unión adaptativa

Descripción general

La unión adaptativa es una característica de vista previa de Aurora PostgreSQL 17.4 que ayuda a mejorar el rendimiento de las consultas. Esta característica está desactivada de forma predeterminada, pero puede habilitarla mediante los parámetros de configuración global de usuario (GUC). Como se trata de una característica de vista previa, es posible que los valores de los parámetros predeterminados cambien. Cuando está habilitada, la unión adaptativa ayuda a optimizar el rendimiento de las consultas al cambiar dinámicamente de una unión de bucles anidados a una combinación hash en tiempo de ejecución. Este cambio se produce cuando el optimizador de PostgreSQL ha elegido incorrectamente una unión de bucles anidados debido a estimaciones de cardinalidad imprecisas.

Configuración de unión adaptativa

Puede controlar la unión adaptativa mediante estos tres parámetros de GUC:

Parámetros de configuración de unión adaptativa

Parámetro de GUC	Descripción	Opciones predeterminadas y de configuración
apg_adaptive_join_crossover_multiplier	Este multiplicador funciona con el punto de cruce de filas para determinar cuándo cambiar de un bucle anidado a una combinación hash. El punto de cruce de filas es donde el optimizador SQL estima que las operaciones de unión de bucle anidado y combinación hash tienen el mismo costo. Un valor multiplicador más alto reduce la probabilidad de que la unión adaptativa cambie a una combinación hash.	Controla si la unión adaptativa está habilitada • Valor predeterminado: -1 (desactivado) • Rango válido: de -1 a DBL_MAX • Para habilitar: establézcalo en >= 1

Parámetro de GUC	Descripción	Opciones predeterminadas y de configuración
apg_adaptive_join_cost_threshold	Este parámetro establece un umbral mínimo de costo de consulta. La unión adaptativa se desactiva automáticamente para las consultas por debajo de este umbral. Esto evita la sobrecarga de rendimien to en consultas sencillas, en las que el costo de planificar una unión adaptativa podría superar los beneficios de cambiar de un bucle anidado a una combinación hash.	Establece el umbral de costo mínimo para la consulta • Valor predeterminado: 100 • Rango válido: de 0 a DBL_MAX

Parámetro de GUC	Descripción	Opciones predeterminadas y de configuración

apg_enable_parameterized_adaptive_join	Este parámetro amplía la funcionalidad de unión adaptativa a las uniones de bucles anidados parametrizadas cuando está habilitado. De forma predeterminada, la unión adaptativa solo funciona con uniones de bucle anidadas sin parametrizar, ya que es más probable que se beneficien del cambio a la combinación hash. Las uniones de bucle anidadas parametrizadas suelen funcionar mejor, lo que hace que el cambio a una combinación hash sea menos crítico.	Controla el comportamiento adaptativo de las uniones de bucle anidadas
		• Valor predeterminado: false
		• Valores válidos: true/false
		  • Cuando es falso: solo funciona con uniones de bucle anidadas sin parametrizar
		  • Cuando es verdadero: funciona con uniones de bucle anidadas parametrizadas y no parametrizadas
		Note
		Requiere que `apg_adaptive_join_crossover_multiplier` esté habilitado primero

Trabajo con tablas no registradas en Aurora PostgreSQL

Amazon Aurora PostgreSQL admite tablas no registradas que son a prueba de errores y mantienen la integridad de los datos incluso después de errores o conmutaciones por error de las instancias del escritor. En PostgreSQL estándar, las tablas no registradas omiten el registro de escritura previa (WAL) durante las operaciones de escritura, lo que se traduce en velocidades de escritura más rápidas. Sin embargo, esto conlleva una menor durabilidad, ya que las tablas no registradas no son a prueba de errores y pueden perder datos tras un error del sistema o un apagado incorrecto. Estas tablas no registradas se truncan automáticamente tras un error o un cierre incorrecto. Sus contenidos e índices tampoco se replican en los servidores en espera.

Por el contrario, Aurora PostgreSQL gestiona las tablas no registradas de forma diferente debido a su arquitectura de almacenamiento distribuido. Esto se debe a que el sistema de almacenamiento de Aurora no depende del WAL de PostgreSQL tradicional para garantizar su durabilidad. Sin embargo, los beneficios de rendimiento que suelen asociarse a las tablas no registradas en PostgreSQL estándar pueden no ser tan significativos en Aurora. Esto se debe a la arquitectura de almacenamiento distribuido de Aurora, que puede generar una sobrecarga adicional en comparación con el almacenamiento local utilizado en PostgreSQL estándar.

Cuando se usen tablas no registradas en Aurora PostgreSQL, tenga en cuenta lo siguiente:

- Solo puede acceder a las tablas no registradas desde el nodo de escritura del clúster de base de datos de Aurora.
- Los nodos de lector solo pueden acceder a las tablas no registradas cuando se promocionan al estado de escritor.
- Al intentar acceder a tablas no registradas desde un nodo de lector, se producirá el siguiente error:

  ```
  cannot access temporary or unlogged relations during recovery.
  ```

Creación de tablas no registradas

Para crear una tabla no registrada en Aurora PostgreSQL, agregue la palabra clave UNLOGGED en la instrucción CREATE TABLE:

```
CREATE UNLOGGED TABLE staging_sales_data (
    transaction_id bigint,
    customer_id bigint,
```

```
    product_id bigint,
    transaction_date date,
    amount NUMERIC
);
```

Gestión de tablas no registradas durante la migración

Al prepararse para migrar datos a Aurora PostgreSQL, es importante identificar y gestionar adecuadamente las tablas no registradas. Las tablas no registradas no se registran en WAL y se excluyen del flujo de replicación, lo que significa que no se replicarán en una réplica de lectura de Aurora.

Convierta las tablas no registradas en tablas registradas o elimínelas si no se están utilizando antes de crear una réplica de lectura de Aurora.

Para verificar la presencia de tablas no registradas en cada base de datos de la instancia, utilice el siguiente comando:

```
SELECT oid, relfilenode, relname, relpersistence, relkind
FROM pg_class
WHERE relpersistence ='u';
```

Para convertir una tabla no registrada de nuevo en una tabla registrada, utilice el siguiente comando:

```
ALTER TABLE table_name SET LOGGED;
```

Esta operación reescribe toda la tabla y la bloquea de forma exclusiva hasta que se complete. En el caso de tablas grandes, esto puede provocar un tiempo de inactividad significativo.

Conversión de tablas no registradas en tablas registradas

Cuando necesite convertir una tabla no registrada de nuevo en una tabla registrada, puede utilizar el siguiente comando:

```
ALTER TABLE table_name SET LOGGED;
```

Esta operación reescribe toda la tabla y la bloquea de forma exclusiva hasta que se complete la operación. En el caso de tablas grandes, esto puede provocar un tiempo de inactividad significativo.

Tablas no registradas y replicación lógica

Por lo general, las tablas no registradas no se incluyen en la replicación lógica porque esta se basa en WAL para capturar y transferir los cambios. De forma predeterminada, los cambios en las tablas no registradas no se registran en WAL y se excluyen del flujo de replicación, lo que las hace inadecuadas para los casos de uso en los que se requiere la replicación lógica. Sin embargo, Aurora PostgreSQL proporciona un parámetro llamado `rds.logically_replicate_unlogged_tables` que le permite controlar este comportamiento:

- Cuando `rds.logically_replicate_unlogged_tables` se establece en 0 (desactivado), las tablas no registradas se excluyen de la replicación lógica.
- Cuando `rds.logically_replicate_unlogged_tables` se establece en 1 (activado), las tablas no registradas se incluyen en la replicación lógica.

> **Note**
>
> En Aurora PostgreSQL, el parámetro `rds.logically_replicate_unlogged_tables` se establece de forma predeterminada en 1 (activado) en las versiones 14 y anteriores, y en 0 (desactivado) en las versiones 15 y posteriores.

Uso de autovacuum de PostgreSQL en Amazon Aurora PostgreSQL

Le recomendamos que use la característica autovacuum para mantener en buen estado su instancia de base de datos de PostgreSQL. Autovacuum automatiza el comienzo de los comandos VACUUM y ANALYZE. Comprueba las tablas con una gran cantidad de tuplas insertadas, actualizadas o eliminadas. Después de esta verificación, recupera el almacenamiento mediante la eliminación de datos obsoletos o tuplas de la base de datos de PostgreSQL.

De forma predeterminada, autovacuum está activado para las instancias de base de datos de Aurora PostgreSQL que crea por medio de cualquiera de los grupos de parámetros de base

de datos de PostgreSQL predeterminados. Otros parámetros de configuración asociados con la característica autovacuum también se establecen de forma predeterminada. Debido a que estos valores predeterminados son algo genéricos, puede beneficiarse de ajustar algunos de los parámetros asociados con la característica autovacuum para su carga de trabajo específica.

A continuación, puede encontrar más información sobre autovacuum y cómo ajustar algunos de sus parámetros en la instancia de base de datos de Aurora PostgreSQL.

Temas

- Asignación de memoria para autovacuum
- Reducción de la probabilidad de reinicio del identificador de transacción
- Determinar si las tablas de una base de datos necesitan vacío
- Determinar qué tablas cumplen actualmente los requisitos de autovacuum
- Determinar si autovacuum se está ejecutando actualmente y durante cuánto tiempo
- Realización de una inmovilización de vacío manual
- Reindexar una tabla cuando autovacuum se está ejecutando
- Administración de autovacuum con índices de gran tamaño
- Otros parámetros que afectan a autovacuum
- Establecimiento de parámetros autovacuum de nivel de tabla
- Registro de actividades de autovacuum y vacuum
- Comportamiento de autovacuum con bases de datos no válidas
- Identificación y resolución de los bloqueadores de vaciado intensivo en Aurora PostgreSQL

Asignación de memoria para autovacuum

Uno de los parámetros más importantes que afectan al desempeño de autovacuum es el parámetro `autovacuum_work_mem`. En las versiones 14 y anteriores de Aurora PostgreSQL, el parámetro `autovacuum_work_mem` se establece en -1, lo que indica que en su lugar se utiliza la configuración de `maintenance_work_mem`. En todas las demás versiones, `autovacuum_work_mem` se determina mediante GREATEST({DBInstanceClassMemory/32768}, 65536).

Las operaciones de limpieza manual siempre utilizan la configuración de `maintenance_work_mem`, con la configuración predeterminada de GREATEST ({DBInstanceClassMemory/63963136*1024},

65536), y también se puede ajustar en la sesión mediante el comando SET para operaciones manuales de VACUUM más específicas.

autovacuum_work_mem determina que la memoria de autovacuum conserve los identificadores de tuplas inactivas (pg_stat_all_tables.n_dead_tup) para los índices de limpieza.

Cuando realice los cálculos para determinar el valor del parámetro autovacuum_work_mem, tenga en cuenta lo siguiente:

- Si define el parámetro en un valor demasiado bajo, el proceso de limpieza puede tener que examinar la tabla varias veces para completar su trabajo. Esta variedad de análisis puede tener un impacto negativo en el rendimiento. Para instancias mayores, configurar maintenance_work_mem o autovacuum_work_mem a 1 GB como mínimo puede mejorar el rendimiento de las tablas de limpieza con un número elevado de tuplas inactivas. Sin embargo, en las versiones 16 y anteriores de PostgreSQL, el uso de memoria de la operación de limpieza está limitado a 1 GB, que es suficiente para procesar aproximadamente 179 millones de tuplas inactivas de una sola pasada. Si una tabla tiene más tuplas inactivas que las indicadas, la operación de limpieza tendrá que realizar varias pasadas por los índices de la tabla, lo que aumentará considerablemente el tiempo necesario. A partir de la versión 17 de PostgreSQL, no hay un límite de 1 GB y autovacuum puede procesar más de 179 millones de tuplas mediante árboles radix.

  Un identificador de tupla tiene un tamaño de 6 bytes. Con el fin de calcular la memoria necesaria para limpiar un índice de una tabla, realice una consulta a pg_stat_all_tables.n_dead_tup para buscar el número de tuplas inactivas y, a continuación, multiplique este número por 6 para determinar la memoria necesaria para limpiar el índice de una sola pasada. Puede utilizar la siguiente consulta:

  ```
  SELECT
      relname AS table_name,
      n_dead_tup,
      pg_size_pretty(n_dead_tup * 6) AS estimated_memory
  FROM
      pg_stat_all_tables
  WHERE
      relname = 'name_of_the_table';
  ```

- El parámetro autovacuum_work_mem funciona en combinación con el parámetro autovacuum_max_workers. Cada empleado de autovacuum_max_workers puede utilizar la memoria que asigne. Si tiene demasiadas tablas pequeñas, asigne más autovacuum_max_workers y menos autovacuum_work_mem. Si tiene tablas grandes (de

100 GB o más), asigne más memoria y menos procesos de trabajo. Debe tener suficiente memoria asignada para que funcione en la tabla más grande. Por lo tanto, asegúrese de que la combinación de procesos de trabajo y memoria sea igual a la memoria total que desea asignar.

Reducción de la probabilidad de reinicio del identificador de transacción

En algunos casos, la configuración de grupos de parámetros relacionada con autovacuum puede no ser lo suficientemente agresiva como para evitar el reinicio del identificador de transacción. Para solucionar esto, Aurora PostgreSQL proporciona un mecanismo que adapta los valores de los parámetros de autovacuum automáticamente. Autovacuum adaptativo es una característica para Aurora PostgreSQL. Puede encontrar una explicación detallada sobre el reinicio del identificador de transacción en la documentación de PostgreSQL.

Autovacuum adaptativo está activado de forma predeterminada para las instancias de Aurora PostgreSQL con el parámetro dinámico `rds.adaptive_autovacuum` establecido en Activado. Le recomendamos encarecidamente que mantenga esta opción activada. Sin embargo, para apagar el ajuste de parámetros autovacuum adaptativo, establezca el parámetro `rds.adaptive_autovacuum` en 0 u OFF.

El reinicio de ID de transacción sigue siendo posible incluso cuando Aurora Amazon RDS ajusta los parámetros de autovacuum. Le animamos a implementar una alarma Amazon CloudWatch para el reinicio de identificador de transacción. Para obtener más información, consulte la publicación Implement an early warning system for transaction ID wraparound in RDS for PostgreSQL (Implementar un sistema de alerta temprana para el ajuste de ID de transacción en RDS for PostgreSQL) en el Blog de Base de datos de AWS.

Con el ajuste de parámetros de autovacuum adaptable activado, Amazon RDS comienza a ajustar los parámetros de autovacuum cuando la métrica de CloudWatch `MaximumUsedTransactionIDs` alcanza el valor del parámetro `autovacuum_freeze_max_age` o 500 000 000, el que sea mayor.

Amazon RDS continúa ajustando los parámetros para el autovacuum si una tabla continúa tendiendo hacia el ajuste de ID de transacción. Cada uno de estos ajustes dedica más recursos a autovacuum para evitar el reinicio. Amazon RDS actualiza los siguientes parámetros relacionados con autovacuum:

- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`

- autovacuum_naptime

RDS modifica estos parámetros solo si el nuevo valor hace que autovacuum sea más agresivo. Estos parámetros se modifican en la memoria en la instancia de base de datos. Los valores en el grupo de parámetros no han cambiado. Para ver la configuración en memoria actual, utilice el comando de SQL PostgreSQL SHOW de PostgreSQL.

Cuando Amazon RDS modifica alguno de estos parámetros de autovacuum, genera un evento para la instancia de base de datos afectada. Este evento se puede ver en la Consola de administración de AWS y a través de la API de Amazon RDS. Una vez que la métrica CloudWatch `MaximumUsedTransactionIDs` vuelve por debajo del límite, Amazon RDS restablece los parámetros relacionados con el autovacuum en la memoria a los valores especificados en el grupo de parámetros. Luego, genera otro evento correspondiente a este cambio.

Determinar si las tablas de una base de datos necesitan vacío

Puede utilizar la siguiente consulta para mostrar el número de transacciones descongeladas en una base de datos. La columna `datfrozenxid` de una fila `pg_database` de una base de datos es un límite inferior en los identificadores de transacción normales que aparecen en esa base de datos. Esta columna es el mínimo de los valores `relfrozenxid` por tabla dentro de la base de datos.

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

Por ejemplo, los resultados de ejecutar la consulta anterior podrían ser los siguientes.

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

Cuando la antigüedad de una base de datos llega a los dos mil millones de identificadores de transacción, se produce el reinicio de los TransactionID (XID) y la base de datos cambia al modo de solo lectura. Puede usar esta consulta para generar una métrica y ejecutarla varias veces al día. De manera predeterminada, autovacuum está configurado para mantener la antigüedad de las transacciones en un máximo de 200,000,000 (`autovacuum_freeze_max_age`).

Una estrategia de monitorización de muestra podría ser la siguiente:

- Establezca el valor `autovacuum_freeze_max_age` en 200 millones de transacciones.
- Si una tabla llega a 500 millones de transacciones descongeladas, se dispara una alarma de gravedad baja. No es un valor disparatado, pero podría indicar que autovacuum no puede mantener el ritmo.
- Si una tabla llega a mil millones, se debe interpretar como una alarma para adoptar medidas. En general, conviene mantener las antigüedades más cerca de `autovacuum_freeze_max_age` por motivos de rendimiento. Le recomendamos que investigue utilizando las recomendaciones que siguen.
- Si una tabla llega a 1500 millones de transacciones sin vaciar, se dispara una alarma de gravedad alta. En función de la velocidad con la que la base de datos use los identificadores de transacción, esta alarma puede indicar que el sistema está agotando el tiempo para ejecutar autovacuum. En ese caso, le recomendamos una solución inmediata.

Si una tabla supera constantemente estos límites, modifique aún más sus parámetros de autovacuum. De manera predeterminada, usar VACUUM manualmente (que tiene deshabilitados los retardos basados en el costo) es un procedimiento más agresivo que usar el autovacuum predeterminado, pero es también más intrusivo para el sistema en su conjunto.

Le recomendamos lo siguiente:

- Esté atento y active un mecanismo de supervisión para que esté al tanto de la antigüedad de sus transacciones.

  A fin de obtener información acerca de la creación de un proceso que advierta sobre el reinicio del ID de transacción, consulte la publicación de blog de la base de datos de AWS sobre la implementación de un sistema de advertencia temprana para el reinicio de un ID de transacción en Amazon RDS for PostgreSQL.
- Para las tablas con más actividad, lleve a cabo una inmovilización manual de vacío con regularidad durante una ventana de mantenimiento además de confiar en autovacuum. Para obtener información acerca de la ejecución de una inmovilización de vacío manual, consulte Realización de una inmovilización de vacío manual.

Determinar qué tablas cumplen actualmente los requisitos de autovacuum

A menudo, hay una o dos tablas que necesitan vacío. Autovacuum se dirige siempre a las tablas cuyo valor `relfrozenxid` sea superior al número de transacciones en `autovacuum_freeze_max_age`. De lo contrario, si el número de tuplas obsoletas desde el último VACUUM supera el límite de vacío, la tabla se vacía.

El umbral de autovacuum se define como:

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

donde el `vacuum base threshold` es `autovacuum_vacuum_threshold`, el `vacuum scale factor` es `autovacuum_vacuum_scale_factor` y el `number of tuples` es `pg_class.reltuples`.

Mientras está conectado a la base de datos, ejecute la siguiente consulta para ver una lista de tablas que autovacuum considera aptas para el vacío.

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
```

```
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Determinar si autovacuum se está ejecutando actualmente y durante cuánto tiempo

Si necesita aspirar manualmente una tabla, asegúrese de determinar si el autovacuum se está ejecutando actualmente. Si es así, es posible que deba ajustar los parámetros para que funcione de manera más eficiente o desactivar el autovacuum temporalmente para que pueda ejecutar manualmente VACUUM.

Use la siguiente consulta para determinar si se está ejecutando autovacuum, cuánto tiempo lleva en ejecución y si se encuentra en espera en otra sesión.

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

Después de ejecutar la consulta, debería ver un resultado similar al siguiente.

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-----------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | active |            | 00:00:00                |
 SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query+
```

```
        |          |       |        |            |                          | FROM
pg_stat_activity
          +
        |          |       |        |            |                          | WHERE
query like '%VACUUM%'
         +
        |          |       |        |            |                          | ORDER BY
xact_start;
      +
```

Existen varios problemas que pueden provocar una sesión de autovaccum de larga duración (es decir, que tarde varios días). El problema más común es que el valor del parámetro `maintenance_work_mem` sea demasiado bajo para el tamaño de la tabla o la velocidad de las actualizaciones.

Le recomendamos que utilice la siguiente fórmula para establecer el valor del parámetro `maintenance_work_mem`.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

Las sesiones de autovacuum con una duración corta también pueden indicar problemas:

- Pueden indicar que no hay un número de `autovacuum_max_workers` suficientemente alto para la carga de trabajo. En ese caso, tendrá que especificar el número de procesos de trabajo.
- Puede indicar que hay una corrupción de índice (autovacuum falla y se reinicia en la misma relación pero no avanza). En este caso, ejecute un manual `vacuum freeze verbose table` para ver la causa exacta.

Realización de una inmovilización de vacío manual

Puede ocurrir que desee realizar una operación de vacío manual en una tabla que ya tenga un proceso de vacío en ejecución. Esto resulta útil si se ha identificado una tabla con una antigüedad cercana a dos mil millones de transacciones (o por encima del umbral que esté monitorizando).

Los siguientes pasos son pautas, con varias variaciones en el proceso. Por ejemplo, durante las pruebas, suponga que descubre que el parámetro `maintenance_work_mem` se definió en un valor demasiado bajo y que tiene que adoptar medidas de forma inmediata en una tabla. Sin embargo, quizás no desea rebotar la instancia en ese momento. Con las consultas de las secciones

anteriores, puede determinar qué tabla está causando el problema y comprobar que hay una sesión de autovacuum que lleva mucho tiempo en ejecución. Sabe que tiene que cambiar el ajuste del parámetro `maintenance_work_mem`, pero también tiene que adoptar medidas de inmediato y aplicar el vacío en la tabla afectada. El siguiente procedimiento muestra qué hacer en esa situación.

Para realizar manualmente una inmovilización de vacío

1. Abra dos sesiones en la base de datos que contiene la tabla en la que desea ejecutar el vacío. Para la segunda sesión, use "screen" u otra utilidad que mantenga la sesión activa si se interrumpe la conexión.
2. En la sesión uno, obtenga el ID de proceso (PID) de la sesión de autovacuum que se ejecuta en la tabla.

   Ejecute la siguiente consulta para obtener el PID de la sesión de autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   xact_start;
   ```

3. En la sesión dos, calcule la cantidad de memoria que necesitará para esta operación. En este ejemplo, determinamos que podemos permitirnos usar un máximo de 2 GB de memoria para esta operación y, por tanto, definimos `maintenance_work_mem` en 2 GB para la sesión actual.

   ```
   SET maintenance_work_mem='2 GB';
   SET
   ```

4. En la sesión dos, ejecute el comando `vacuum freeze verbose` para la tabla. El ajuste de informe detallado resulta útil porque, aunque PostgreSQL no ofrece actualmente un informe de progreso para esto, se puede ver la actividad.

   ```
   \timing on
   Timing is on.
   vacuum freeze verbose pgbench_branches;
   ```

   ```
   INFO:  vacuuming "public.pgbench_branches"
   INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
   DETAIL:  0 index row versions were removed.
   0 index pages have been deleted, 0 are currently reusable.
   CPU 0.00s/0.00u sec elapsed 0.00 sec.
   ```

```
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. En la sesión uno, si la limpieza automática bloqueaba la sesión de vacío, `pg_stat_activity` muestra que la espera es T para la sesión de vacío. En este caso, finalice el proceso de limpieza automática de la siguiente manera.

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Algunas versiones anteriores de Amazon Aurora no pueden finalizar un proceso de limpieza automática mediante el comando anterior y producen el siguiente error: `ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227`. Para encontrar las versiones de PostgreSQL a las que se les han aplicado parches, busque el siguiente punto en las actualizaciones de Amazon Aurora PostgreSQL:
>
> ```
> Allow rds_superuser to terminate backends which are not explicitly
>  associated with a role
> ```

En este punto, comienza la sesión. La limpieza automática se reinicia inmediatamente, ya que esta tabla es probablemente la que ocupa una posición más alta en la lista de trabajo.

6. Inicie el comando `vacuum freeze verbose` en la sesión dos y luego finalice el proceso de autovacuum en la sesión uno.

Reindexar una tabla cuando autovacuum se está ejecutando

Si un índice se ha dañado, autovacuum seguirá procesando la tabla y generará errores. Si intenta realizar un vacío manual en esta situación, recibirá un mensaje de error como el siguiente.

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

Cuando el índice está dañado y autovacuum intenta ejecutarse en la tabla, se enfrenta a una sesión de autovacuum que ya se está ejecutando. Cuando ejecuta un comando REINDEX, elimina un bloqueo exclusivo en la tabla. Las operaciones de escritura están bloqueadas y también las operaciones de lectura que usan ese índice específico.

Para reindexar una tabla cuando autovacuum se está ejecutando en ella

1. Abra dos sesiones en la base de datos que contiene la tabla que desea vaciar. Para la segunda sesión, use "screen" u otra utilidad que mantenga la sesión activa si se interrumpe la conexión.
2. En la sesión uno, obtenga el PID de la sesión de autovacuum que se ejecuta en la tabla.

   Ejecute la siguiente consulta para obtener el PID de la sesión de autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```
3. En la sesión dos, ejecute el comando reindex.

   ```
   \timing on
   Timing is on.
   reindex index pgbench_branches_test_index;
   REINDEX
     Time: 9.966 ms
   ```
4. En la sesión uno, si autovacuum estaba bloqueando, verá en `pg_stat_activity` que la espera es “T” para su sesión de vacío. En este caso, terminará el proceso de autovacuum.

   ```
   SELECT pg_terminate_backend('the_pid');
   ```

En este punto, comienza la sesión. Es importante tener en cuenta que autovacuum se reiniciará inmediatamente, ya que esta tabla es probablemente la que ocupa una posición más alta en su lista de trabajo.

5. Inicie el comando en la sesión dos y termine a continuación el proceso de autovacuum de la sesión 1.

Administración de autovacuum con índices de gran tamaño

Como parte de su funcionamiento, autovacuum realiza varias fases de vaciado mientras se ejecuta en una tabla. Antes de limpiar la tabla, primero se vacían todos sus índices. Al eliminar varios índices de gran tamaño, esta fase consume una cantidad importante de tiempo y recursos. Por lo tanto, como práctica recomendada, asegúrese de controlar el número de índices de una tabla y eliminar los no utilizados.

Para este proceso, compruebe primero el tamaño general del índice. A continuación, determine si hay índices que es posible que no se utilicen y que se puedan eliminar, tal y como se muestra en los siguientes ejemplos.

Para comprobar el tamaño de la tabla y sus índices

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
pg_size_pretty
11 GB
(1 row)
```

En este ejemplo, el tamaño de los índices es mayor que el de la tabla. Esta diferencia puede provocar problemas de rendimiento, ya que los índices están sobrecargados o no se utilizan, lo que afecta a las operaciones de autovacuum y de inserción.

Para comprobar si hay índices no utilizados

En la vista pg_stat_user_indexes, puede comprobar la frecuencia con la que se utiliza un índice con la columna idx_scan. En el siguiente ejemplo, los índices no utilizados tienen el valor 0 en idx_scan.

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
-------+------------+------------+------------------+-----------------------+----------
+--------------+---------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
public      | pgbench_accounts | index_d               | 0
public      | pgbench_accounts | index_e               | 0
public      | pgbench_accounts | index_c               | 0
public      | pgbench_accounts | index_a               | 0
(7 rows)
```

> **Note**
>
> Estas estadísticas son incrementales desde el momento en que se restablecen las estadísticas. Supongamos que tiene un índice que solo se usa al final de un trimestre empresarial o solo para un informe específico. Es posible que este índice no se haya utilizado desde que se restablecieron las estadísticas. Para obtener más información, consulte Statistics Functions (Funciones de estadísticas). Los índices que se utilizan para garantizar la exclusividad no se analizan y no se deben identificar como índices no utilizados. Para identificar los índices no utilizados, debe tener un conocimiento profundo de la aplicación y sus consultas.

Para comprobar cuándo se restablecieron por última vez las estadísticas de una base de datos, utilice `pg_stat_database`.

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

Vaciado de una tabla lo más rápido posible

RDS para PostgreSQL 12 y versiones posteriores

Si tiene demasiados índices en una tabla grande, la instancia de base de datos podría estar a punto de reiniciar el identificador de transacción (XID), que es cuando el contador de XID vuelve a ponerse en cero. Si esta casilla no se marca, esta situación podría provocar la pérdida de datos. Sin embargo, puede vaciar rápidamente la tabla sin limpiar los índices. En RDS para PostgreSQL 12 y versiones posteriores, puede usar VACUUM con la cláusula `INDEX_CLEANUP`.

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
```

```
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

Si ya se está ejecutando una sesión de autovacuum, debe finalizarla para iniciar VACUUM manualmente. Para obtener información acerca de la ejecución de una inmovilización de vacío manual, consulte Realización de una inmovilización de vacío manual.

> **Note**
>
> Omitir la limpieza de índices con regularidad causa una sobrecarga de los índices, lo que degrada el rendimiento del análisis. El índice retiene las filas inactivas y la tabla retiene los punteros de línea inactivos. Como resultado, `pg_stat_all_tables.n_dead_tup` aumenta hasta que se ejecuta el vaciado automático o una operación VACUUM manual con limpieza de índices. Como práctica recomendada, use este procedimiento solo para impedir que el identificador se reinicie.

RDS para PostgreSQL 11 y versiones anteriores

Sin embargo, en RDS para PostgreSQL 11 y versiones anteriores, la única forma de hacer que el vacío se realice más rápidamente es reducir el número de índices de una tabla. La eliminación de un índice puede afectar a los planes de consulta. Le recomendamos que primero borre los índices no utilizados y, a continuación, los índices cuando el reinicio de XID sea inminente. Una vez finalizado el proceso de vaciado, puede volver a crear estos índices.

Otros parámetros que afectan a autovacuum

La siguiente consulta mostrará los valores de algunos de los parámetros que afectan directamente a autovacuum y a su comportamiento. Los parámetros de autovacuum se describen en detalle en la documentación de PostgreSQL.

```
SELECT name, setting, unit, short_desc
FROM pg_settings
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
```

```
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

Aunque todos estos parámetros afectan a autovacuum, estos son algunos de los más importantes:

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

Establecimiento de parámetros autovacuum de nivel de tabla

Los parámetros de almacenamiento relacionados con autovacuum se pueden definir en el nivel de tabla, algo que puede resultar mejor que alterar el comportamiento de toda la base de datos. Para las tablas grandes, podría ser necesario definir unos ajustes agresivos, y es posible que no sea deseable que autovacuum se comporte de esa forma para todas las tablas.

La siguiente consulta mostrará qué tablas tienen habilitadas actualmente las opciones de nivel de tabla.

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

Un ejemplo en el que esto puede resultar útil es el de las tablas que son mucho más grandes que el resto de las tablas. Supongamos que dispone de una tabla de 300 GB y otras 30 tablas inferior a 1 GB. En ese caso, podría definir algunos parámetros concretos para la tabla grande con el fin de evitar alterar el comportamiento de todo el sistema.

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

Al hacer esto, desactiva el retraso de autovacuum basado en costos para esta tabla a expensas de un mayor uso de recursos en su sistema. Normalmente, autovacuum hace una pausa por `autovacuum_vacuum_cost_delay` cada vez que se alcanza `autovacuum_cost_limit`. En la documentación de PostgreSQL, puede obtener información detallada relativa al vacío basado en el costo.

Registro de actividades de autovacuum y vacuum

La información sobre las actividades de autovacuum se envía a `postgresql.log` basado en el nivel especificado en el parámetro `rds.force_autovacuum_logging_level`. Los siguientes son los valores permitidos para este parámetro y las versiones de PostgreSQL para las que ese valor es la configuración predeterminada:

- `disabled` (PostgreSQL 10, PostgreSQL 9.6)
- `debug5`, `debug4`, `debug3`, `debug2`, `debug1`
- `info` (PostgreSQL 12, PostgreSQL 11)
- `notice`
- `warning` (PostgreSQL 13 y versiones posteriores)
- `error`, registro, `fatal`, `panic`

`rds.force_autovacuum_logging_level` funciona con el parámetro `log_autovacuum_min_duration`. El valor del parámetro `log_autovacuum_min_duration` es el límite (en milisegundos) por encima del cual se registran las acciones de autovacuum. Una configuración de `-1` no registra nada, mientras que una configuración de 0 registra todas las acciones. Al igual que con `rds.force_autovacuum_logging_level`, los valores predeterminados para `log_autovacuum_min_duration` dependen de la versión, como se indica a continuación:

- `10000 ms`: PostgreSQL 14, PostgreSQL 13, PostgreSQL 12 y PostgreSQL 11
- `(empty)`: no hay valor predeterminado para PostgreSQL 10 y PostgreSQL 9.6

Es recomendable que defina `rds.force_autovacuum_logging_level` como `WARNING`. También recomendamos configurar `log_autovacuum_min_duration` a un valor de 1000 a 5000.

Una configuración de 5000 registra la actividad que tarda más de 5000 milisegundos. Cualquier configuración que no sea -1 también registra mensajes si la acción de autovaccum se omite debido a un bloqueo en conflicto o relaciones eliminadas al mismo tiempo. Para más información, visite Automatic Vacuuming (Vacío automático) en la documentación de PostgreSQL.

Para solucionar problemas, puede cambiar el parámetro `rds.force_autovacuum_logging_level` a uno de los niveles de depuración, desde `debug1` hasta `debug5` para obtener la información más detallada. Le recomendamos que utilice la configuración de depuración durante periodos cortos y solo con el objetivo de solucionar problemas. Para más información, visite When to log (Cuándo registrarse) en la documentación de PostgreSQL.

> **Note**
>
> PostgreSQL permite a la cuenta `rds_superuser` consultar sesiones de autovacuum en `pg_stat_activity`. Por ejemplo, podrá identificar y finalizar una sesión de autovacuum que bloquea la ejecución de un comando o que hace que se ejecute más despacio que un comando de vacío emitido manualmente.

Comportamiento de autovacuum con bases de datos no válidas

Se introduce un nuevo valor `-2` en la columna `datconnlimit` del catálogo `pg_database` para indicar que las bases de datos interrumpidas en mitad de la operación DROP DATABASE no son válidas.

Este nuevo valor está disponible en las siguientes versiones de Aurora PostgreSQL:

- Versión 15.4 y todas las versiones posteriores
- Versión 14.9 y posteriores
- Versión 13.12 y posteriores
- Versión 12.16 y posteriores
- Versión 11.21 y posteriores

Las bases de datos no válidas no afectan a la capacidad de autovacuum de bloquear la funcionalidad de las bases de datos válidas. Autovacuum ignora las bases de datos no válidas. Por lo tanto, las operaciones vacuum habituales seguirán funcionando de forma adecuada y eficiente en todas las bases de datos válidas de su entorno de PostgreSQL.

Temas

- Supervisión del ID de transacción
- Ajustes en la consulta de supervisión
- Resolución de problemas relacionados con bases de datos no válidas

Supervisión del ID de transacción

La función `age(datfrozenxid)` se suele utilizar para supervisar la antigüedad del ID de transacción (XID) de las bases de datos a fin de evitar que este se reinicie.

Como las bases de datos no válidas se excluyen del autovacuum, su contador de ID de transacción (XID) puede alcanzar el valor máximo de `2 billion`, reiniciarse en `- 2 billion` y continuar este ciclo indefinidamente. Una consulta típica para supervisar el reinicio del ID de transacción podría ser así:

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

Sin embargo, al introducir el valor -2 para `datconnlimit`, las bases de datos no válidas pueden sesgar los resultados de esta consulta. Como estas bases de datos no son válidas y no deberían formar parte de las comprobaciones de mantenimiento periódicas, pueden provocar falsos positivos y dar la impresión de que `age(datfrozenxid)` es mayor de lo que realmente es.

Ajustes en la consulta de supervisión

Para garantizar una supervisión precisa, debe ajustar la consulta de supervisión a fin de excluir las bases de datos no válidas. Siga esta consulta recomendada:

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
WHERE
    datconnlimit <> -2;
```

Esta consulta garantiza que solo se tengan en cuenta las bases de datos válidas en el cálculo de `age(datfrozenxid)`, lo que aporta información fiable sobre la antigüedad del ID de transacción en todo el entorno de PostgreSQL.

Resolución de problemas relacionados con bases de datos no válidas

Al intentar conectarse a una base de datos no válida, es posible que vea un mensaje de error similar al siguiente:

```
postgres=> \c db1
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

Además, si el parámetro `log_min_messages` tiene un valor igual o superior a `DEBUG2`, es posible que vea que las siguientes entradas de registro muestran que el proceso de autovacuum omite la base de datos no válida:

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

Para resolver el problema, siga la `HINT` proporcionada durante el intento de conexión. Conéctese a cualquier base de datos válida mediante su cuenta maestra de RDS o una cuenta de base de datos con el rol `rds_superuser` y elimine las bases de datos no válidas.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

Identificación y resolución de los bloqueadores de vaciado intensivo en Aurora PostgreSQL

En PostgreSQL, la limpieza es fundamental para garantizar el buen estado de la base de datos, ya que recupera espacio de almacenamiento y evita problemas relacionados con los identificadores de transacciones. Sin embargo, hay situaciones que pueden impedir que el vaciado funcione como es debido, lo que puede mermar el rendimiento, provocar una sobrecarga de almacenamiento e incluso afectar a la disponibilidad de la instancia de base de datos debido a la superposición de ID de transacción. Por lo tanto, identificar y resolver estos problemas es esencial para lograr un rendimiento y una disponibilidad óptimos de la base de datos. Consulte Understanding autovacuum in Amazon RDS for PostgreSQL environments si desea conocer mejor la limpieza automática.

La función `postgres_get_av_diag()` ayuda a identificar los problemas que impiden o retrasan el avance de la limpieza agresiva. Se proporcionan sugerencias, entre otras, comandos para resolver el problema si es identificable o indicaciones para realizar diagnósticos adicionales cuando no se puede identificar el problema. Los bloqueadores de limpieza agresiva aparecen cuando su antigüedad supera el umbral de vacío automático adaptativo establecido por RDS, que es de 500 millones de identificadores de transacciones.

¿Qué antigüedad tiene el identificador de la transacción?

La función `age()` de identificadores de transacción calcula el número de transacciones que se han producido desde el identificador de transacción descongelado más antiguo de una base de datos (`pg_database.datfrozenxid`) o tabla (`pg_class.relfrozenxid`). Este valor indica la actividad de la base de datos desde la última operación de limpieza agresiva y resalta la carga de trabajo probable para los próximos procesos de LIMPIEZA.

¿Qué es una limpieza agresiva?

Una operación de LIMPIEZA agresiva lleva a cabo un escaneo exhaustivo de todas las páginas de una tabla, incluidas las que normalmente se omiten durante las limpiezas normales. Este análisis exhaustivo tiene como objetivo congelar los ID de transacción que se acercan a su antigüedad máxima, evitando de forma eficaz una situación conocida como superposición de identificadores de transacción.

Para que `postgres_get_av_diag()` pueda detectar bloqueadores, el bloqueador debe haber realizado al menos 500 millones de transacciones.

Temas

- Instalación de herramientas de supervisión y diagnóstico de autovacuum en Aurora PostgreSQL
- Funciones de postgres_get_av_diag() en Aurora PostgreSQL
- Resolución de bloqueadores de vaciado identificables en Aurora PostgreSQL
- Resolución de bloqueadores de vaciado no identificables en Aurora PostgreSQL
- Resolución de problemas de rendimiento de vaciado en Aurora PostgreSQL
- Explicación de los mensajes de tipo NOTICE en Aurora PostgreSQL

Instalación de herramientas de supervisión y diagnóstico de autovacuum en Aurora PostgreSQL

La función `postgres_get_av_diag()` está disponible actualmente en las siguientes versiones de Aurora PostgreSQL:

- Versión 17.4 y otras versiones 17 posteriores
- Versión 16.7 y otras versiones 16 superiores
- Versión 15.11 y otras versiones 15 superiores
- Versión 14.16 y otras versiones 14 superiores
- Versión 13.19 y otras versiones 13 superiores

Para utilizar `postgres_get_av_diag()`, cree la extensión `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

Compruebe que la extensión esté instalada.

```
postgres=> \dx rds_tools
             List of installed extensions
   Name    | Version |  Schema   |                    Description
 ----------+---------+-----------
+-------------------------------------------------------------
 rds_tools |   1.9   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

Compruebe que la función se haya creado.

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

Funciones de postgres_get_av_diag() en Aurora PostgreSQL

La función `postgres_get_av_diag()` recupera información de diagnóstico sobre los procesos de autovacuum que se bloquean o se retrasan en una base de datos de Aurora PostgreSQL. La consulta debe ejecutarse en la base de datos con el ID de transacción más antiguo para obtener resultados precisos. Para obtener más información sobre el uso de la base de datos con el ID de transacción más antiguo, consulte Not connected to the database with the age of oldest transaction ID

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
        rds_tools.postgres_get_av_diag ()
    ORDER BY
        autovacuum_lagging_by DESC) q;
```

La función `postgres_get_av_diag()` devuelve una tabla con la siguiente información:

blocker

Especifica la categoría de actividad de la base de datos que bloquea el vaciado.

- Instrucción activa
- Inactividad en la transacción
- Transacción preparada
- Ranura de replicación lógica
- Instancias de lector
- Tablas temporales

database

Especifica el nombre de la base de datos, si está disponible y es compatible. Esta es la base de datos en la que la actividad está en curso y bloquea o bloqueará el autovacuum. Esta es la base de datos a la que debe conectarse y sobre la que debe actuar.

blocker_identifier

Especifica el identificador de la actividad que bloquea o bloqueará el autovacuum. El identificador puede ser un ID de proceso junto con una instrucción SQL, una transacción preparada, una dirección IP de una réplica de lectura y el nombre de la ranura de replicación, ya sea lógica o física.

wait_event

Especifica el evento de espera de la sesión de bloqueo y se aplica a los siguientes bloqueadores:

- Instrucción activa
- Inactividad en la transacción

autovacum_lagging_by

Especifica el número de transacciones que tiene pendiente el autovacuum según sus trabajos por realizar y por categoría.

suggestion

Especifica sugerencias para resolver el bloqueo. Estas instrucciones incluyen el nombre de la base de datos en la que existe la actividad, cuando proceda, el ID de proceso (PID) de la sesión, cuando proceda, y la acción que se debe realizar.

suggested_action

Sugiere la acción que se debe llevar a cabo para resolver el bloqueo.

Resolución de bloqueadores de vaciado identificables en Aurora PostgreSQL

Autovacuum lleva a cabo vaciados de forma intensiva y reduce la antigüedad de los ID de transacción hasta situarlos por debajo del umbral especificado por el parámetro `autovacuum_freeze_max_age` de la instancia de RDS. Esta antigüedad se puede consultar mediante la métrica `MaximumUsedTransactionIDs` de Amazon CloudWatch.

Para encontrar la configuración de `autovacuum_freeze_max_age` (que tiene un valor predeterminado de 200 millones de ID de transacción) para una instancia de Amazon RDS, puede utilizar la siguiente consulta:

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
    name = 'autovacuum_freeze_max_age';
```

Tenga en cuenta que `postgres_get_av_diag()` solo comprueba si hay bloqueadores de vaciado intensivo cuando la antigüedad supera el umbral de autovacuum adaptativo de Amazon RDS de 500 millones de ID de transacción. Para que `postgres_get_av_diag()` detecte los bloqueadores, el bloqueador debe tener al menos 500 millones de transacciones de antigüedad.

La función `postgres_get_av_diag()` identifica los siguientes tipos de bloqueadores:

Temas

- Instrucción activa
- Inactividad en la transacción
- Transacción preparada
- Ranura de replicación lógica
- Instancias de lector
- Tablas temporales

Instrucción activa

En PostgreSQL, una instrucción activa es una instrucción SQL que la base de datos está ejecutando actualmente. Incluye consultas, transacciones o cualquier operación en curso. Al realizar la

supervisión mediante `pg_stat_activity`, la columna de estado indica que el proceso con el PID correspondiente está activo.

La función `postgres_get_av_diag()` muestra un resultado similar al siguiente cuando identifica una instrucción que resulta ser una instrucción activa.

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

Acción sugerida

Siguiendo las instrucciones de la columna `suggestion`, el usuario puede conectarse a la base de datos en la que se encuentra la instrucción activa y, tal como se especifica en la columna `suggested_action`, se recomienda revisar detenidamente la opción de finalizar la sesión. Si la finalización es segura, se puede utilizar la función `pg_terminate_backend()` para finalizar la sesión. Esta acción la puede realizar un administrador (como la cuenta maestra de RDS) o un usuario con el privilegio `pg_terminate_backend()` necesario.

> **Warning**
>
> Al finalizar la sesión, se desharán (`ROLLBACK`) los cambios que haya realizado. En función de sus requisitos, es posible que quiera volver a ejecutar la instrucción. Sin embargo, se recomienda hacerlo únicamente después de que el proceso de autovacuum haya finalizado su operación de vaciado intensivo.

Inactividad en la transacción

El concepto de inactividad en una instrucción de transacción se refiere a cualquier sesión en la que se haya abierto una transacción explícita (por ejemplo, emitiendo una instrucción `BEGIN`), se haya realizado algún trabajo y se esté esperando a que el cliente pase más trabajo o dé la señal de finalización de la transacción emitiendo una instrucción `COMMIT`, `ROLLBACK` o `END` (lo que daría como resultado un `COMMIT` implícitamente).

La función `postgres_get_av_diag()` muestra un resultado similar al siguiente cuando identifica una instrucción `idle in transaction` como bloqueador.

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

Acción sugerida

Como se indica en la columna `suggestion`, puede conectarse a la base de datos en la que se encuentra la sesión de inactividad en la transacción y finalizar la sesión mediante la función `pg_terminate_backend()`. El usuario puede ser su usuario administrador (cuenta maestra de RDS) o un usuario con el privilegio `pg_terminate_backend()`.

> **Warning**
>
> Al finalizar la sesión, se desharán (`ROLLBACK`) los cambios que haya realizado. En función de sus requisitos, es posible que quiera volver a ejecutar la instrucción. Sin embargo, se recomienda hacerlo únicamente después de que el proceso de autovacuum haya finalizado su operación de vaciado intensivo.

Transacción preparada

PostgreSQL permite realizar transacciones que forman parte de una estrategia de confirmación de dos fases denominada transacciones preparadas. Se habilitan al establecer el parámetro `max_prepared_transactions` en un valor distinto de cero. Las transacciones preparadas han sido diseñadas para garantizar que una transacción sea duradera y permanezca disponible incluso después de que la base de datos se bloquee, se reinicie o se desconecte del cliente. Al igual que las transacciones normales, se les asigna un identificador de transacción y pueden afectar al autovacuum. Si se deja en un estado preparado, el autovacuum no puede realizar la congelación y podría provocar un reinicio del ID de transacción.

Cuando las transacciones se dejan preparadas indefinidamente sin que las resuelva un administrador de transacciones, se convierten en transacciones preparadas huérfanas. La única forma de solucionar este problema es confirmar o revertir la transacción mediante los comandos `COMMIT PREPARED` o `ROLLBACK PREPARED` respectivamente.

> **Note**
>
> Tenga en cuenta que una copia de seguridad realizada durante una transacción preparada seguirá conteniendo esa transacción después de la restauración. Consulte la siguiente información sobre cómo localizar y cerrar dichas transacciones.

La función `postgres_get_av_diag()` muestra el siguiente resultado cuando identifica un bloqueador que es una transacción preparada.

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

Acción sugerida

Como se menciona en la columna de sugerencias, conéctese a la base de datos en la que se encuentre la transacción preparada. Sobre la base de la columna `suggested_action`, revise detenidamente si desea enviar una instrucción `COMMIT` o `ROLLBACK`, y realizar la acción correspondiente.

Para supervisar las transacciones preparadas en general, PostgreSQL ofrece una vista de catálogo llamada `pg_prepared_xacts`. Puede utilizar la siguiente consulta para buscar transacciones preparadas.

```
SELECT
    gid,
    prepared,
```

```
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

Ranura de replicación lógica

El propósito de una ranura de replicación es almacenar los cambios no consumidos hasta que se repliquen en un servidor de destino. Para obtener más información, consulte Logical replication de PostgreSQL.

Existen dos tipos de ranuras de replicación lógica.

Ranuras de replicación lógica inactivas

Cuando finaliza la replicación, los registros de transacciones no consumidas no se pueden eliminar y la ranura de replicación queda inactiva. Aunque un suscriptor no utilice actualmente una ranura de replicación lógica inactiva, esta permanece en el servidor, lo que provoca la retención de los archivos WAL y evita la eliminación de los registros de transacciones antiguos. Esto puede aumentar el uso del disco y, específicamente, impedir que autovacuum limpie las tablas del catálogo interno, ya que el sistema debe evitar que se sobrescriba la información de LSN. Si este problema no se soluciona, puede provocar una sobrecarga del catálogo, una degradación del rendimiento y un mayor riesgo de que se produzcan vaciados previos al reinicio, lo que podría causar tiempo de inactividad en las transacciones.

Ranuras de replicación lógica activas pero lentas

A veces, la eliminación de las tuplas inactivas del catálogo se retrasa debido a la degradación del rendimiento de la replicación lógica. Este retraso en la replicación ralentiza la actualización de `catalog_xmin` y puede provocar una sobrecarga del catálogo y un vaciado previo al reinicio.

La función `postgres_get_av_diag()` muestra un resultado similar al siguiente cuando encuentra una ranura de replicación lógica que funciona como bloqueador.

```
blocker              | Logical replication slot
database             | my_database
blocker_identifier   | slot1
```

```
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

Acción sugerida

Para resolver este problema, compruebe la configuración de la replicación para ver si hay problemas con el esquema o los datos de destino que puedan estar finalizando el proceso de aplicación. Los motivos más comunes son los siguientes:

- Columnas faltantes
- Tipos de datos incompatibles
- Discrepancia de datos
- Tabla faltante

Si el problema está relacionado con problemas de infraestructura:

- Problemas de red: ¿cómo resuelvo los problemas con una base de datos de Amazon RDS en un estado de red incompatible?
- La base de datos o la instancia de base de datos no están disponibles por una de las siguientes razones:
  - La instancia de réplica se ha quedado sin espacio de almacenamiento: consulte qué hacer cuando las instancias de base de datos de Amazon RDS se quedan sin almacenamiento para obtener información sobre cómo añadir almacenamiento.
  - Parámetros incompatibles: revise ¿Cómo puedo corregir una instancia de base de datos de Amazon RDS que está estancada en el estado parámetros incompatibles? para obtener más información acerca de cómo solucionar este problema.

Si la instancia está fuera de la red de AWS o en AWS EC2, consulte a su administrador sobre cómo resolver los problemas relacionados con la disponibilidad o la infraestructura.

Eliminación de la ranura inactiva

> **Warning**
>
> Precaución: Antes de eliminar una ranura de replicación, asegúrese exhaustivamente de que no tenga ninguna replicación en curso, de que esté inactiva y de que se encuentre en un estado irrecuperable. Si se elimina una ranura de forma prematura, se podría interrumpir la replicación o provocar la pérdida de datos.

Después de confirmar que la ranura de replicación ya no es necesaria, elimínela para permitir que el autovacuum continúe. La condición `active = 'f'` garantiza que solo se eliminará una ranura inactiva.

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

Instancias de lector

Cuando la configuración hot_standby_feedback está habilitada, evita que el autovacuum de la instancia de escritura elimine filas muertas que podrían seguir siendo necesarias para las consultas que se ejecutan en la instancia de lector. Este comportamiento es necesario porque las consultas que se ejecutan en la instancia de lector (también aplicable a las instancias de lector en la base de datos global de Aurora) requieren que esas filas permanezcan disponibles en la instancia de escritura, lo que evita conflictos y cancelaciones de consultas.

> **Note**
>
> `hot_standby_feedback` está habilitado de forma predeterminada y no se puede modificar en Aurora PostgreSQL.

La función `postgres_get_av_diag()` muestra un resultado similar al siguiente cuando encuentra una réplica de lectura con una ranura de replicación física como bloqueador.

```
blocker              | Oldest query running on aurora reader
database             | Not applicable
blocker_identifier   | my-aurora-reader-2
wait_event           | Not applicable
autovacuum_lagging_by | 540,122,859
```

```
suggestion            | Run the following query on the reader "my-aurora-reader-2" to
 find the long running query:

                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 523476310;

                      | Review carefully and you may consider terminating the query on
 reader using suggested_action.
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 523476310;","

                      | [OR]


                      | ","Delete the reader if not needed"}
```

Como se recomienda en la columna `suggested_action`, revise detenidamente estas opciones para desbloquear el autovacuum.

- Finalizar la consulta: de acuerdo con las instrucciones de la columna de sugerencias, puede conectarse a la réplica de lectura, tal y como se especifica en la columna suggested_action. Se recomienda revisar detenidamente la opción para finalizar la sesión. Si la finalización se considera segura, se puede utilizar la función `pg_terminate_backend()` para finalizar la sesión. Esta acción la puede realizar un administrador (como la cuenta maestra de RDS) o un usuario con el privilegio pg_terminate_backend() necesario.

  Puede ejecutar el siguiente comando SQL en la réplica de lectura para finalizar la consulta que impide que el proceso de vaciado en el principal pueda limpiar las filas antiguas. El valor de `backend_xmin` se indica en la salida de la función:

  ```
  SELECT
      pg_terminate_backend(pid)
  FROM
      pg_catalog.pg_stat_activity
  WHERE
      backend_xmin::text::bigint = backend_xmin;
  ```

- Eliminar las instancias de lectura si no son necesarias: si la instancia de lectura ya no es necesaria, puede eliminarla. Esto eliminará la sobrecarga de replicación asociada y permitirá que el servidor principal recicle los registros de transacciones sin que la instancia se lo obstaculice.

Tablas temporales

Las tablas temporales, que se crean con la palabra clave `TEMPORARY`, residen en el esquema temporal (por ejemplo, pg_temp_xxx) y solo la sesión que las haya creado puede acceder a ellas. Las tablas temporales se eliminan al finalizar la sesión. Sin embargo, estas tablas son invisibles para el proceso de autovacuum de PostgreSQL y la sesión que las haya creado debe vaciarlas manualmente. Intentar vaciar la tabla temporal desde otra sesión no tiene ningún efecto.

En circunstancias poco habituales, puede existir una tabla temporal sin que sea propiedad de una sesión activa. Si la sesión propietaria finaliza inesperadamente debido a un bloqueo grave, un problema de red o un suceso similar, es posible que la tabla temporal no se limpie y quede como una tabla “huérfana”. Cuando el proceso de autovacuum de PostgreSQL detecta una tabla temporal huérfana, registra el siguiente mensaje:

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

La función `postgres_get_av_diag()` muestra un resultado similar al siguiente cuando identifica una tabla temporal como bloqueador. Para que la función muestre correctamente el resultado relacionado con las tablas temporales, debe ejecutarse en la misma base de datos en la que se encuentren esas tablas.

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

Acción sugerida

Siga las instrucciones que aparecen en la columna `suggestion` del resultado para identificar y eliminar la tabla temporal que impide la ejecución del autovacuum. Use el siguiente comando para eliminar la tabla temporal notificada por `postgres_get_av_diag()`. Reemplace el nombre de la tabla en función del resultado proporcionado por la función `postgres_get_av_diag()`.

```
DROP TABLE my_temp_schema.my_temp_table;
```

La siguiente consulta se puede utilizar para identificar tablas temporales:

```
SELECT
    oid,
    relname,
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

Resolución de bloqueadores de vaciado no identificables en Aurora PostgreSQL

En esta sección se analizan otros motivos que pueden impedir que el progreso del vaciado. Actualmente, la función `postgres_get_av_diag()` no puede identificar directamente estos problemas.

Temas

- Incoherencia en los índices
- Tasa de transacciones excepcionalmente alta

Incoherencia en los índices

Un índice que no sea coherente desde el punto de vista lógico puede impedir que avance el autovacuum. Los siguientes errores u otros similares se registran durante la fase de vaciado del índice o cuando se accede al índice mediante instrucciones SQL.

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

Indicaciones

Reconstruya el índice u omita los índices utilizando `INDEX_CLEANUP` con un `VACUUM FREEZE` manual.

- Uso de la opción CONCURRENTLY: antes de la versión 12 de PostgreSQL, la reconstrucción de un índice requería un bloqueo de tabla exclusivo, lo que restringía el acceso a la misma. Con PostgreSQL versión 12 y versiones posteriores, la opción CONCURRENTLY permite el bloqueo por filas, lo que mejora significativamente la disponibilidad de la tabla. A continuación, se muestra el comando:

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  Si bien CONCURRENTLY resulta menos disruptivo, puede ser más lento en tablas de uso intensivo. Si es posible, considere la posibilidad de crear el índice durante los períodos de poco tráfico. Para obtener más información, consulte REINDEX en la documentación de PostgreSQL.
- Uso de la opción INDEX_CLEANUP FALSE: si los índices son grandes y se calcula que tardarán mucho en completarse, puede desbloquear el autovacuum ejecutando un VACUUM FREEZE manual y excluyendo los índices. Esta funcionalidad está disponible en la versión 12 y posteriores de PostgreSQL.

  Omitir los índices le permitirá saltarse el proceso de vaciado del índice incoherente y mitigar el problema del reinicio. Sin embargo, esto no resolverá el problema subyacente de la página no válida. Para solucionar por completo el problema de la página no válida y resolverlo, tendrá que volver a crear el índice.

Tasa de transacciones excepcionalmente alta

En PostgreSQL, las tasas de transacción altas pueden afectar significativamente al rendimiento de autovacuum, lo que implica una limpieza más lenta de las tuplas inactivas y a un aumento del riesgo de reiniciar los ID de transacción. Puede supervisar la tasa de transacciones midiendo la diferencia en `max(age(datfrozenxid))` entre dos períodos de tiempo, normalmente por segundo. Además, puede utilizar las siguientes métricas de contador de Información de rendimiento de RDS para medir la tasa de transacciones (la suma de xact_commit y xact_rollback), que es el número total de transacciones.

Contador	Tipo	Unidad	Métrica
xact_commit	Transacciones	Confirmaciones por segundo	db.Transactions.xact_commit

Contador	Tipo	Unidad	Métrica
xact_rollback	Transacciones	Restauraciones por segundo	db.Transactions.xact_rollback

Un aumento rápido indica una alta carga de transacciones, lo que puede ser excesivo para autovacuum y provocar sobrecargas, bloqueos y posibles problemas de rendimiento. Esto puede tener un impacto negativo en el proceso de autovacuum de dos maneras:

- Actividad de la tabla: la tabla específica que se está vaciando podría estar registrando un gran volumen de transacciones, lo que provocaría retrasos.
- Recursos del sistema: el sistema en general puede estar sobrecargado, lo que dificulta que autovacuum acceda a los recursos necesarios para funcionar de manera eficiente.

Plantéese las siguientes estrategias para permitir que autovacuum funcione de manera más eficaz y pueda seguir el ritmo de sus tareas:

1. Reduzca la tasa de transacciones si es posible. Plantéese la posibilidad de agrupar o agrupar transacciones similares cuando sea posible.
2. Utilice tablas que se actualicen con frecuencia mediante la operación `VACUUM FREEZE` manual cada noche, semana o quincena durante las horas de menor actividad.
3. Plantéese la posibilidad de escalar verticalmente su clase de instancia para asignar más recursos del sistema con el fin de administrar el volumen de transacciones elevado y el autovacuum.

Resolución de problemas de rendimiento de vaciado en Aurora PostgreSQL

En esta sección se analizan los factores que suelen contribuir a reducir el rendimiento del vaciado y cómo abordar estos problemas.

Temas

- Vaciado de índices grandes
- Demasiadas tablas o bases de datos que vaciar
- Se está ejecutando un vaciado intensivo (para evitar el reinicio)

Vaciado de índices grandes

VACUUM funciona a través de fases secuenciales: inicialización, escaneo del montón, vaciado de índices y montón, limpieza de índices, truncamiento del montón y limpieza final. Durante el escaneo del montón, el proceso elimina las páginas, las desfragmenta y las congela. Después de completar el escaneo del montón, VACUUM limpia los índices, se devuelven las páginas vacías al sistema operativo y se realizan tareas de limpieza final, como el vaciado del mapa del espacio libre y la actualización de las estadísticas.

Es posible que sea necesario realizar varias pasadas para el vaciado de índices cuando `maintenance_work_mem` (o `autovacuum_work_mem`) no es suficiente para procesar el índice. En la versión 16 y anteriores de PostgreSQL, un límite de memoria de 1 GB para almacenar los ID de tuplas inactivas a menudo forzaba varias pasadas en índices grandes. PostgreSQL 17 presenta `TidStore`, que asigna memoria de forma dinámica en lugar de utilizar una matriz de asignación única. Esto elimina la restricción de 1 GB, utiliza la memoria de manera más eficiente y reduce la necesidad de realizar varios análisis de índice por cada índice.

Es posible que los índices grandes aún requieran varias pasadas en PostgreSQL 17 si la memoria disponible no puede acomodar todo el procesamiento del índice de una vez. Por lo general, los índices mayores contienen más tuplas inactivas que requieren varias pasadas.

Detección de operaciones de limpieza lentas

La función `postgres_get_av_diag()` puede detectar cuando las operaciones de limpieza se ejecutan lentamente debido a memoria insuficiente. Para obtener más información sobre esta función, consulte Instalación de herramientas de supervisión y diagnóstico de autovacuum en Aurora PostgreSQL.

La función `postgres_get_av_diag()` emite los siguientes avisos cuando la memoria disponible no es suficiente para completar la limpieza del índice en una sola pasada.

`rds_tools` 1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
```

```
       memory. The suggested value is an estimate based on the current number of dead
tuples for the table being vacuumed, which might not fully reflect the latest state.
Additionally, review the possibility of manual
       vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
TRUE) table_name;). For more information, see

       Working with PostgreSQL autovacuum in the Amazon Aurora User Guide.
```

> **Note**
>
> La función `postgres_get_av_diag()` se basa en `pg_stat_all_tables.n_dead_tup` para estimar la cantidad de memoria necesaria para el vaciado de índices.

Cuando la función `postgres_get_av_diag()` identifique una operación de limpieza lenta que requiera múltiples análisis de índice debido a que no hay suficiente `autovacuum_work_mem`, generará el siguiente mensaje:

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
       For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

Indicaciones

Puede aplicar las siguientes soluciones alternativas utilizando manualmente `VACUUM FREEZE` para acelerar la congelación de la tabla.

Aumentar la memoria de vaciado

Como sugiere la función `postgres_get_av_diag()`, se recomienda aumentar el parámetro `autovacuum_work_mem` para abordar las posibles restricciones de memoria en cada instancia. Aunque `autovacuum_work_mem` es un parámetro dinámico, es importante tener en cuenta que, para que la nueva configuración de memoria surta efecto, el daemon de autovacuum debe reiniciar sus procesos de trabajo. Para lograrlo:

1. Confirme que la nueva configuración esté establecida.
2. Finalice los procesos que actualmente estén ejecutando el autovacuum.

Este enfoque garantiza que la asignación de memoria ajustada se aplique a las nuevas operaciones de autovacuum.

Para obtener resultados más inmediatos, considere la posibilidad de realizar manualmente una operación `VACUUM FREEZE` con una configuración de `maintenance_work_mem` mayor durante la sesión:

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Si utiliza Amazon RDS y descubre que necesita memoria adicional para poder utilizar valores más altos para `maintenance_work_mem` o `autovacuum_work_mem`, plantéese la posibilidad de actualizar a una clase de instancia con más memoria. Esto puede proporcionarle los recursos necesarios para mejorar las operaciones de vaciado manuales y automáticas, lo que se traduce en una mejora del rendimiento general de vaciado y del de las bases de datos.

Desactivar INDEX_CLEANUP

El `VACUUM` manual de la versión 12 y posteriores de PostgreSQL permite omitir la fase de limpieza de índices, mientras que el autovacuum de emergencia en la versión 14 y posteriores de PostgreSQL lo hace automáticamente en función del parámetro `vacuum_failsafe_age`.

> **Warning**
>
> Omitir la limpieza de índices puede provocar una sobrecarga de índices y perjudicar el rendimiento de las consultas. Para mitigar esta situación, considere la posibilidad de volver a indexar o vaciar los índices afectados durante un período de mantenimiento.

Para obtener más información sobre cómo gestionar índices grandes, consulte la documentación en Administración de autovacuum con índices de gran tamaño .

Vaciado de índices en paralelo

A partir de PostgreSQL 13, los índices se pueden vaciar y limpiar en paralelo de forma predeterminada utilizando `VACUUM` de forma manual, con un proceso de trabajo de vaciado asignado a cada índice. Sin embargo, para que PostgreSQL determine si una operación de vaciado es apta para su ejecución en paralelo, se deben cumplir criterios específicos:

- Debe haber al menos dos índices.
- El parámetro `max_parallel_maintenance_workers` debe estar establecido al menos en 2.
- El tamaño del índice debe superar el límite `min_parallel_index_scan_size`, que de forma predeterminada es de 512 KB.

Puede ajustar la configuración `max_parallel_maintenance_workers` en función de la cantidad de vCPU disponibles en su instancia de Amazon RDS y la cantidad de índices de la tabla para optimizar el tiempo de respuesta del vaciado.

Para obtener más información, consulte Parallel vacuuming in Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL.

Demasiadas tablas o bases de datos que vaciar

Como se menciona en la documentación de PostgreSQL sobre el daemon autovacuum, este funciona mediante múltiples procesos. Esto incluye un lanzador de autovacuum persistente responsable de iniciar los procesos de trabajo de autovacuum para cada base de datos del sistema. El lanzador programa estos procesos de trabajo para que se inicien aproximadamente cada `autovacuum_naptime` segundos por cada base de datos.

Con “N” bases de datos, un nuevo proceso de trabajo comienza aproximadamente cada [`autovacuum_naptime`/N segundos]. Sin embargo, el número total de procesos de trabajo simultáneos está limitado por la configuración `autovacuum_max_workers`. Si el número de bases de datos o tablas que requieren vaciado supera este límite, la siguiente base de datos o tabla se procesará en cuanto haya un proceso de trabajo disponible.

Cuando muchas tablas o bases de datos grandes requieren un vaciado al mismo tiempo, todos los procesos de trabajo de autovacuum disponibles pueden permanecer ocupados durante un período prolongado, lo que retrasa el mantenimiento de otras tablas y bases de datos. En entornos con altas tasas de transacciones, este cuello de botella puede agravarse rápidamente y provocar posibles problemas de vaciado en su instancia de Amazon RDS.

Cuando `postgres_get_av_diag()` detecta un número elevado de tablas o bases de datos, proporciona la siguiente recomendación:

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

Indicaciones

Aumentar autovacuum_max_workers

Para agilizar el vaciado, recomendamos ajustar el parámetro `autovacuum_max_workers` para permitir que haya más procesos de trabajo de autovacuum simultáneos. Si persisten los cuellos de botella en el rendimiento, plantéese la posibilidad de escalar verticalmente su instancia de Amazon RDS a una clase con más vCPU, lo que puede mejorar aún más las capacidades de procesamiento en paralelo.

Se está ejecutando un vaciado intensivo (para evitar el reinicio)

La antigüedad de la base de datos (MaximumUsedTransactionIDs) en PostgreSQL solo disminuye cuando se completa correctamente un vaciado intensivo (para evitar el reinicio). Hasta que finalice este vaciado, la antigüedad seguirá aumentando en función de la velocidad de transacciones.

La función `postgres_get_av_diag()` genera el `NOTICE` siguiente cuando detecta un vaciado intensivo. Sin embargo, solo activa este resultado después de que el vaciado haya estado activo durante al menos dos minutos.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Para obtener más información sobre el vaciado intensivo, consulte When an aggressive vacuum is already running.

Puede comprobar si se está realizando un vaciado intensivo con la siguiente consulta:

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
```

```
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
    pg_stat_activity a
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

Para determinar si se trata de un vaciado intensivo (para evitar el reinicio), compruebe la columna de consulta del resultado. La expresión “para evitar el reinicio” indica que se trata de un vaciado intensivo.

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

Por ejemplo, supongamos que hay un bloqueador en el valor de antigüedad de transacciones de 1000 millones y una tabla que requiere un vaciado intensivo para evitar el reinicio a esa misma antigüedad de transacciones. Además, hay otro bloqueador en el valor de antigüedad de transacciones de 750 millones. Tras superar el bloqueador en el valor de antigüedad de transacciones de 1000 millones, la antigüedad no se reducirá inmediatamente a 750 millones. Seguirá siendo alta hasta que se complete la tabla que necesita el vaciado intensivo o cualquier transacción con una antigüedad superior a los 750 millones. Durante este período, la antigüedad de las transacciones de su clúster de PostgreSQL seguirá aumentando. Una vez que se complete el proceso de vaciado, la antigüedad de las transacciones se reducirá a 750 millones, pero volverá a aumentar de nuevo hasta que se finalice todo el vaciado. Este ciclo continuará mientras

se mantengan estas condiciones, hasta que la antigüedad de las transacciones finalmente se reduzca hasta el nivel configurado para su instancia de Amazon RDS, especificado por `autovacuum_freeze_max_age`.

Explicación de los mensajes de tipo NOTICE en Aurora PostgreSQL

La función `postgres_get_av_diag()` proporciona los siguientes mensajes de tipo NOTICE:

Cuando la antigüedad aún no ha alcanzado aún el umbral de supervisión

El umbral de supervisión para que `postgres_get_av_diag()` identifique los bloqueadores es de 500 millones de transacciones por defecto. Si `postgres_get_av_diag()` genera el siguiente mensaje NOTICE, indica que la antigüedad de la transacción aún no ha alcanzado este umbral.

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

No está conectado a la base de datos que tenga la antigüedad del ID de transacción más antiguo

La función `postgres_get_av_diag()` proporciona el resultado más preciso cuando se conecta a la base de datos con el ID de transacción más antiguo. En su caso, la base de datos con el ID de transacción más antiguo notificada por `postgres_get_av_diag()` será diferente a “my_database”. Si no se ha conectado a la base de datos correcta, se generará el siguiente mensaje tipo NOTICE:

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

Conectarse a la base de datos con la antigüedad de transacción más antigua es importante por las siguientes razones:

- Identificar los bloqueadores de tablas temporales: dado que los metadatos de las tablas temporales son específicos de cada base de datos, normalmente se encuentran en la base de datos en la que se crearon. Sin embargo, si una tabla temporal resulta ser la que más bloquea y reside en la base de datos con la transacción más antigua, esta información podría resultar engañosa. La conexión a la base de datos correcta garantiza la identificación precisa del bloqueador de tablas temporal.

- Diagnóstico de vaciados lentos: los metadatos del índice y la información sobre el recuento de tablas son específicos de la base de datos y son necesarios para diagnosticar los problemas de vaciado lento.

La base de datos con la transacción más antigua se encuentra en una base de datos rdsadmin o template0

En algunos casos, las bases de datos `rdsadmin` o `template0` pueden identificarse como la base de datos con el ID de transacción más antiguo. Si esto ocurre, `postgres_get_av_diag()` emitirá el siguiente mensaje de tipo NOTICE:

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

Compruebe que el bloqueador de la lista no se haya originado en ninguna de estas dos bases de datos. Si se notifica que el bloqueador está presente en `rdsadmin` o `template0` de ellas, póngase en contacto con el servicio de asistencia, ya que estas bases de datos no son accesibles para el usuario y requieren intervención.

Es muy poco probable que las bases de datos `rdsadmin` o `template0` contengan un bloqueador principal.

Cuando ya está en curso un vaciado intensivo

La función `postgres_get_av_diag()` está diseñada para notificar si se está ejecutando un proceso de vaciado intensivo, pero solo activa esta salida después de que el vaciado haya estado activo durante al menos 1 minuto. Este retraso intencionado ayuda a reducir las probabilidades de que se produzcan falsos positivos. Mediante esta espera, la función garantiza que solo se registren los vaciados efectivos y significativos, lo que permite una supervisión más precisa y fiable de la actividad de vaciado.

La función `postgres_get_av_diag()` genera el siguiente mensaje de tipo NOTICE cuando detecta que se están realizando uno o varios vaciados intensivos.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Como se indica en el mensaje NOTICE, siga supervisando el rendimiento del vaciado. Para obtener más información acerca del vaciado intensivo, consulte Se está ejecutando un vaciado intensivo (para evitar el reinicio)

Cuando el vaciado intensivo está apagado

La función `postgres_get_av_diag()` genera el siguiente mensaje NOTICE si el autovacuum está deshabilitado en la instancia de la base de datos:

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

Autovacuum es una característica fundamental de la instancia de base de datos de Aurora PostgreSQL que garantiza un funcionamiento fluido de la base de datos. Elimina automáticamente las versiones de filas antiguas, recupera espacio de almacenamiento y evita que las tablas se sobrecarguen, lo que ayuda a mantener la eficiencia de las tablas y los índices para lograr un rendimiento óptimo. Además, protege contra el reinicio de los identificadores de transacciones, lo que puede detener las transacciones en su instancia de Amazon RDS. La desactivación del autovacuum puede provocar una disminución a largo plazo del rendimiento y la estabilidad de la base de datos. Le sugerimos que lo mantenga activado todo el tiempo. Para obtener más información, consulte Descripción de autovacuum en entornos de Aurora PostgreSQL.

> **Note**
>
> La desactivación de autovacuum no detiene los vaciados intensivos. Seguirán produciéndose una vez que las tablas alcancen el umbral de `autovacuum_freeze_max_age`.

El número de transacciones pendientes es críticamente bajo

La función `postgres_get_av_diag()` genera el siguiente mensaje de tipo NOTICE cuando un vaciado previo al reinicio es inminente. Este mensaje NOTICE se emite cuando su instancia de Amazon RDS está a 100 millones de transacciones de la posibilidad de rechazar nuevas transacciones.

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

Es necesario realizar acciones inmediatas para evitar el tiempo de inactividad de la base de datos. Debe supervisar de cerca sus operaciones de vaciado y considerar la posibilidad de

iniciar manualmente un `VACUUM FREEZE` en la base de datos afectada para evitar errores en las transacciones.

Administración de la contención de TOAST OID en Amazon Aurora PostgreSQL

TOAST (la técnica de almacenamiento de atributos sobredimensionados) es una característica de PostgreSQL diseñada para gestionar valores de datos de gran tamaño que superan el tamaño típico de un bloque de base de datos de 8 KB. PostgreSQL no permite que las filas físicas abarquen varios bloques. El tamaño del bloque actúa como límite superior del tamaño de las filas. TOAST supera esta restricción al dividir los valores de campo grandes en fragmentos más pequeños. Los almacena de forma independiente en una tabla TOAST específica vinculada a la tabla principal. Para obtener más información, consulte la documentación de implementación y mecanismo de almacenamiento de PostgreSQL TOAST.

Temas

- Descripción de las operaciones de TOAST
- Identificación de los desafíos de rendimiento
- Recomendaciones
- Monitorización

Descripción de las operaciones de TOAST

TOAST realiza la compresión y almacena valores de campo grandes fuera de línea. TOAST asigna un OID (identificador de objeto) único a cada fragmento de datos sobredimensionados almacenados en la tabla TOAST. La tabla principal almacena el ID del valor TOAST y el ID de relación en la página para hacer referencia a la fila correspondiente de la tabla TOAST. Esto permite a PostgreSQL localizar y administrar de forma eficiente estos fragmentos de TOAST. Sin embargo, a medida que crece la tabla de TOAST, el sistema corre el riesgo de agotar los OID disponibles, lo que provoca una degradación del rendimiento y un posible tiempo de inactividad debido al agotamiento de los OID.

Identificadores de objetos en TOAST

Un identificador de objeto (OID) es un identificador único de todo el sistema que PostgreSQL utiliza para hacer referencia a objetos de bases de datos, como tablas, índices y funciones. Estos

identificadores desempeñan un papel fundamental en las operaciones internas de PostgreSQL, ya que permiten a la base de datos localizar y administrar los objetos de forma eficiente.

Para las tablas con conjuntos de datos elegibles para acciones de toast, PostgreSQL asigna los OID para identificar de forma única cada fragmento de datos sobredimensionados almacenado en la tabla TOAST asociada. El sistema asocia cada fragmento con un `chunk_id`, lo que ayuda a PostgreSQL a organizar y ubicar estos fragmentos de manera eficiente dentro de la tabla TOAST.

Identificación de los desafíos de rendimiento

La administración del OID de PostgreSQL se basa en un contador global de 32 bits para poder agrupar después de generar 4 mil millones de valores únicos. Aunque el clúster de bases de datos comparte este contador, la asignación del OID implica dos pasos durante las operaciones TOAST:

- Contador global de asignación: el contador global asigna un nuevo OID a todo el clúster.
- Búsqueda local de conflictos: la tabla TOAST garantiza que el nuevo OID no entre en conflicto con los OID existentes que ya se utilizan en esa tabla específica.

La degradación del rendimiento puede producirse cuando:

- La tabla TOAST tiene una alta fragmentación o un uso denso del OID, lo que provoca retrasos en la asignación del OID.
- Con frecuencia, el sistema asigna y reutiliza los OID en entornos con fragmentos de datos altos o tablas amplias que utilizan TOAST de forma extensiva.

Para obtener más información, consulte los límites de tamaño de las tablas TOAST de PostgreSQL y la documentación de asignación de OID:

Un contador global genera los OID y los agrupa cada 4000 millones de valores, de modo que, de vez en cuando, el sistema vuelve a generar un valor ya utilizado. PostgreSQL lo detecta y lo vuelve a intentar con el siguiente OID. Se puede producir INSERT lento si hay una serie muy larga de valores OID usados sin espacios en blanco en la tabla TOAST. Estos desafíos se hacen más pronunciados a medida que se va llenando el espacio del OID, lo que hace que las inserciones y actualizaciones sean más lentas.

Identificación del problema

- Las instrucciones `INSERT` simples tardan mucho más de lo habitual de forma incoherente y aleatoria.
- Solo se producen retrasos en las instrucciones `INSERT` y `UPDATE` que implican operaciones de TOAST.
- Las siguientes entradas de registro aparecen en los registros de PostgreSQL cuando el sistema tiene dificultades para encontrar los OID disponibles en las tablas TOAST:

```
LOG: still searching for an unused OID in relation "pg_toast_20815"
DETAIL: OID candidates have been checked 1000000 times, but no unused OID has been
 found yet.
```

- La información de rendimiento indica un número elevado de sesiones activas de media (AAS) asociadas a eventos de espera `LWLock:buffer_io` y `LWLock:OidGenLock`.

  Puede ejecutar la siguiente consulta SQL para identificar las transacciones INSERT de larga duración con eventos de espera:

```
SELECT
    datname AS database_name,
    usename AS database_user,
    pid,
    now() - pg_stat_activity.xact_start AS transaction_duration,
    concat(wait_event_type, ':', wait_event) AS wait_event,
    substr(query, 1, 30) AS TRANSACTION,
    state
FROM
    pg_stat_activity
WHERE (now() - pg_stat_activity.xact_start) > INTERVAL '60 seconds'
    AND state IN ('active', 'idle in transaction', 'idle in transaction (aborted)',
 'fastpath function call', 'disabled')
    AND pid <> pg_backend_pid()
AND lower(query) LIKE '%insert%'
ORDER BY
    transaction_duration DESC;
```

  Ejemplos de resultados de consulta que muestran operaciones INSERT con tiempos de espera prolongados:

```
 database_name |  database_user  |  pid  | transaction_duration |     wait_event
 |          transaction           | state
---------------+-----------------+-------+----------------------
+---------------------+--------------------------------+--------
 postgres       | db_admin_user| 70965 | 00:10:19.484061      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 69878 | 00:06:14.976037      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 68937 | 00:05:13.942847      | :                   |
 INSERT INTO "products" (......... | active
```

Aislamiento del problema

- Pruebe una inserción pequeña: inserte un registro más pequeño que el umbral `toast_tuple_target`. Recuerde que la compresión se aplica antes del almacenamiento de TOAST. Si esto funciona sin problemas de rendimiento, el problema está relacionado con las operaciones de TOAST.
- Pruebe la tabla nueva: cree una tabla nueva con la misma estructura e ingrese un registro más grande que `toast_tuple_target`. Si esto funciona sin problemas, el problema se localiza en la asignación OID de la tabla original.

Recomendaciones

Los siguientes enfoques pueden ayudar a resolver los problemas de contención de TOAST OID.

- Limpieza y archivado de datos: revise y elimine cualquier dato obsoleto o innecesario para liberar los OID para su uso futuro o archive los datos. Tenga en cuenta las siguientes restricciones:
  - Escalabilidad limitada, ya que es posible que no siempre sea posible realizar limpiezas en el futuro.
  - Posible operación VACUUM de larga duración para eliminar las tuplas inactivas resultantes.
- Escriba en una tabla nueva: cree una tabla nueva para futuras inserciones y utilice una vista `UNION ALL` para combinar datos antiguos y nuevos para las consultas. Esta vista presenta los datos combinados de las tablas antiguas y nuevas, lo que permite a las consultas acceder a ellos como una sola tabla. Tenga en cuenta las siguientes restricciones:
  - Es posible que las actualizaciones de la tabla antigua sigan agotando los OID.

- Partición o fragmento: particione los datos de la tabla o del fragmento para mejorar la escalabilidad y el rendimiento. Tenga en cuenta las siguientes restricciones:
  - El aumento de la complejidad de la lógica y el mantenimiento de las consultas, y la posible necesidad de realizar cambios en las aplicaciones para gestionar correctamente los datos particionados.

Monitorización

Uso de tablas de sistemas

Puede utilizar las tablas del sistema de PostgreSQL para supervisar el aumento del uso de los OID.

> **Warning**
>
> En función del número de OID de la tabla de TOAST, puede tardar un tiempo en completarse. Le recomendamos que programe la supervisión fuera del horario laboral para minimizar el impacto.

El siguiente bloque anónimo cuenta el número de OID distintos que se utilizan en cada tabla de TOAST y muestra la información de la tabla principal:

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the number of distinct used OIDs (chunk IDs) from the TOAST table
```

```
            EXECUTE 'SELECT COUNT(DISTINCT chunk_id) FROM ' || r.toast_table INTO o;
            -- If there are used OIDs, find the associated parent table and its schema
            IF o <> 0 THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table: %
 | Number of used OIDs: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Ejemplo de salida que muestra las estadísticas de uso de los OID por tabla de TOAST:

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Number of used OIDs: 45,623,317
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Number of used OIDs: 10,000
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Number of used OIDs: 1,000,000
DO
```

El siguiente bloque anónimo recupera el OID máximo asignado para cada tabla de TOAST que no esté vacía:

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
```

```
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the max(chunk_id) from the TOAST table
            EXECUTE 'SELECT max(chunk_id) FROM ' || r.toast_table INTO o;
            -- If there's at least one TOASTed chunk, find the associated parent table
 and its schema
            IF o IS NOT NULL THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table:
 % | Max chunk_id: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Ejemplo de salida que muestra el número máximo de ID de fragmentos para las tablas de TOAST:

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Max chunk_id: 45,639,907
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Max chunk_id: 45,649,929
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Max chunk_id: 46,649,935
DO
```

Uso de información sobre rendimiento

Los eventos de espera `LWLock:buffer_io` y `LWLock:OidGenLock` aparecen en la información de rendimiento durante las operaciones que requieren la asignación de nuevos identificadores de objeto (OID). El promedio alto de sesiones activas (AAS) de estos eventos suele indicar que hay problemas durante la asignación de los OID y la administración de los recursos. Esto es particularmente común en entornos con una alta cantidad de datos, un uso extensivo de datos de gran tamaño o una creación frecuente de objetos.

LWLock:buffer_io

`LWLock:buffer_io` es un evento de espera que se produce cuando una sesión de PostgreSQL espera a que se completen las operaciones de E/S en un búfer compartido. Esto suele ocurrir cuando la base de datos lee datos del disco a la memoria o escribe páginas modificadas de la memoria al disco. El evento de espera de `BufferIO` garantiza la coherencia al impedir que varios procesos accedan o modifiquen el mismo búfer mientras las operaciones de E/S están en curso. La alta incidencia de este evento de espera puede indicar cuellos de botella en el disco o una actividad de E/S excesiva en la carga de trabajo de la base de datos.

Durante las operaciones de TOAST:

- PostgreSQL asigna los OID a los objetos grandes y garantiza su exclusividad escaneando el índice de la tabla de TOAST.
- Los índices de TOAST de gran tamaño pueden requerir el acceso a varias páginas para verificar la exclusividad de los OID. Esto se traduce en un aumento de la E/S del disco, especialmente cuando el grupo de búferes no puede almacenar en caché todas las páginas requeridas.

El tamaño del índice afecta directamente a la cantidad de páginas del búfer a las que se debe acceder durante estas operaciones. Incluso si el índice no está sobrecargado, su gran tamaño puede aumentar la E/S del búfer, especialmente en entornos de alta concurrencia o alta rotación. Para obtener más información, consulte la LWLock:BufferIO wait event troubleshooting guide.

LWLock:OidGenLock

`OidGenLock` es un evento de espera que se produce cuando una sesión de PostgreSQL está esperando para asignar un nuevo identificador de objetos (OID). Este bloqueo garantiza que los OID se generen de forma secuencial y segura, lo que permite que solo un proceso genere los OID a la vez.

Durante las operaciones de TOAST:

- Asignación de OID para los fragmentos de la tabla de TOAST: PostgreSQL asigna los OID a los fragmentos de las tablas de TOAST al administrar registros de datos de gran tamaño. Cada OID debe ser único para evitar conflictos en el catálogo del sistema.
- Alta simultaneidad: dado que el acceso al generador de OID es secuencial, cuando varias sesiones crean simultáneamente objetos que requieren OID, se puede producir contención para `OidGenLock`. Esto aumenta la probabilidad de que las sesiones esperen a que se complete la asignación de los OID.
- Dependencia del acceso al catálogo del sistema: la asignación de los OID requiere actualizaciones en las tablas del catálogo del sistema compartido, como `pg_class` y `pg_type`. Si estas tablas tienen mucha actividad (debido a las frecuentes operaciones de DDL), puede aumentar la contención de bloqueos para `OidGenLock`.
- Alta demanda de asignación del OID: las cargas de trabajo pesadas de TOAST con registros de datos de gran tamaño requieren una asignación coherente del OID, lo que aumenta la contención.

Factores adicionales que aumentan la contención del OID:

- Creación frecuente de objetos: las cargas de trabajo que crean y descartan objetos con frecuencia, como tablas temporales, aumentan la contención en el contador de OID global.
- Bloqueo del contador global: se accede en serie al contador de OID global para garantizar su exclusividad, lo que crea un único punto de contención en entornos de alta concurrencia.

Uso de Babelfish para Aurora PostgreSQL

Babelfish para Aurora PostgreSQL amplía su clúster de base de datos Aurora PostgreSQL con la capacidad de aceptar conexiones de bases de datos de clientes de SQL Server. Con Babelfish, las aplicaciones que se crearon originalmente para SQL Server pueden funcionar directamente con Aurora PostgreSQL con pocos cambios de código en comparación con una migración tradicional y sin cambiar los controladores de base de datos. Para obtener más información sobre cómo migrar, consulte Migración de una base de datos SQL Server a Babelfish para Aurora PostgreSQL.

Babelfish proporciona un punto de conexión adicional para un clúster de bases de datos de Aurora PostgreSQL que le permite comprender el protocolo de nivel de conexión de SQL Server y las instrucciones de SQL Server de uso común. Las aplicaciones cliente que utilizan el protocolo de conexión de flujo de datos tabulares (TDS) pueden conectarse de forma nativa al puerto del agente de escucha de TDS en Aurora PostgreSQL. Para obtener más información sobre TDS, consulte[MS-TDS]: Tabular Data Stream Protocol ([MS-TDS]: Protocolo de flujo de datos tabulares) en el sitio web de Microsoft.

> **Note**
>
> Babelfish para Aurora PostgreSQL es compatible con las versiones 7.1 a 7.4 de TDS.

Babelfish también proporciona acceso a los datos mediante la conexión de PostgreSQL. De forma predeterminada, los dos dialectos SQL que admite Babelfish están disponibles a través de sus protocolos de conexión nativos en los siguientes puertos:

- Dialecto de SQL Server (T-SQL), los clientes se conectan al puerto 1433.
- Dialecto de PostgreSQL (PL/pgSQL), los clientes se conectan al puerto 5432.

Babelfish utiliza el lenguaje Transact-SQL (T-SQL) con algunas diferencias. Para obtener más información, consulte Diferencias entre Babelfish for Aurora PostgreSQL y SQL Server.

En las siguientes secciones, puede encontrar información sobre la configuración y el uso de un clúster de base de datos de Babelfish para Aurora PostgreSQL.

Temas

- Limitaciones de Babelfish
- Descripción de la arquitectura y configuración de Babelfish

- Creación de un clúster de base de datos de Babelfish para Aurora PostgreSQL
- Migración de una base de datos SQL Server a Babelfish para Aurora PostgreSQL
- Autenticación de bases de datos con Babelfish para Aurora PostgreSQL
- Conexión a un clúster de base de datos de Babelfish
- Uso de Babelfish
- Solución de problemas de Babelfish
- Desactivación de Babelfish
- Administración de las actualizaciones de versiones de Babelfish para Aurora PostgreSQL
- Referencia de Babelfish para Aurora PostgreSQL

Limitaciones de Babelfish

Las limitaciones siguientes se aplican actualmente a Babelfish para Aurora PostgreSQL:

- Babelfish no admite las siguientes características de Aurora:
  - AWS Identity and Access Management
  - Secuencias de actividades de la base de datos (DAS)
  - API de datos de RDS con Aurora PostgreSQL sin servidor v2 y aprovisionada
  - RDS Proxy con RDS para SQL Server
  - SCRAM (mecanismo de autenticación mediante desafío-respuesta discontinuo)
  - Editor de consultas
- Babelfish no ofrece el siguiente soporte de API de controlador de cliente:
  - Las solicitudes de API con los atributos de conexión relacionados con el Coordinador de transacciones distribuidas de Microsoft (MSDTC) no se admiten. Estas incluyen las llamadas XA realizadas por la clase SQLServerXAResource en el controlador JDBC del servidor SQL.
- Actualmente, Babelfish no admite las siguientes extensiones de Aurora PostgreSQL:
  - `bloom`
  - `btree_gin`
  - `btree_gist`
  - `citext`
  - `cube`
  - `hstore`
  - `hypopg`
  - Replicación lógica mediante `pglogical`
  - `ltree`
  - `pgcrypto`
  - Administración de planes de consultas con `apg_plan_mgmt`

  Para obtener más información sobre las extensiones de PostgreSQL, consulte Uso de extensiones y contenedores de datos externos.
- No se admite el controlador JTDS de código abierto que está diseñado como una alternativa al controlador JDBC de Microsoft.

Descripción de la arquitectura y configuración de Babelfish

El clúster de base de datos de Aurora PostgreSQL-Compatible Edition que ejecuta Babelfish se administra de la misma manera que cualquier clúster de base de datos de Aurora. Es decir, se beneficia de la escalabilidad, la alta disponibilidad con compatibilidad con la conmutación por error y la replicación integrada que proporciona un clúster de base de datos de Aurora. Para obtener más información acerca de estas capacidades, consulte Administración del rendimiento y el escalado para clústeres de base de datos Aurora, Alta disponibilidad para Amazon Aurora yReplicación con Amazon Aurora. También tiene acceso a otras muchas herramientas y utilidades de AWS, como las siguientes:

- Amazon CloudWatch es un servicio de supervisión y observabilidad que le proporciona datos e información procesable. Para obtener más información, consulte Supervisión de métricas de Amazon Aurora con Amazon CloudWatch.
- Información sobre rendimiento es una característica de ajuste y supervisión del rendimiento de la base de datos que ayuda a evaluar rápidamente la carga de la base de datos. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.
- Las bases de datos globales de Aurora abarcan múltiples Regiones de AWS, lo que permite lecturas globales de baja latencia y proporcionan una recuperación rápida de cualquier interrupción que pueda afectar a toda una Región de AWS. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.
- La aplicación automática de revisiones de software mantiene la base de datos actualizada con las revisiones de características y seguridad más recientes cuando estén disponibles.
- Los eventos de Amazon RDS le notifican por correo electrónico o mensaje SMS los eventos importantes de la base de datos, como una conmutación por error automatizada. Para obtener más información, consulte Supervisión de eventos de Amazon Aurora.

A continuación, podrá conocer la arquitectura de Babelfish y cómo las bases de datos de SQL Server que se migran las gestiona Babelfish. Cuando cree el clúster de base de datos de Babelfish, tendrá que tomar algunas decisiones por adelantado sobre si se trata de una o varias bases de datos, intercalaciones y otros detalles.

Temas

- Arquitectura de Babelfish
- Configuración del grupo de parámetros del clúster de base de datos para Babelfish

- Comprensión de las intercalaciones de Babelfish para Aurora PostgreSQL
- Administración de la gestión de errores de Babelfish con escotillas de escape

Arquitectura de Babelfish

Al crear un clúster de Aurora PostgreSQL con Babelfish activado, Aurora aprovisiona el clúster con una base de datos PostgreSQL denominada `babelfish_db`. Esta base de datos es donde residen todos los objetos y estructuras de SQL Server migrados.

> **Note**
>
> En un clúster Aurora PostgreSQL, el nombre de la base de datos `babelfish_db` está reservado para Babelfish. La creación de su propia base de datos "babelfish_db" en un clúster de base de datos de Babelfish impide que Aurora aprovisione correctamente Babelfish.

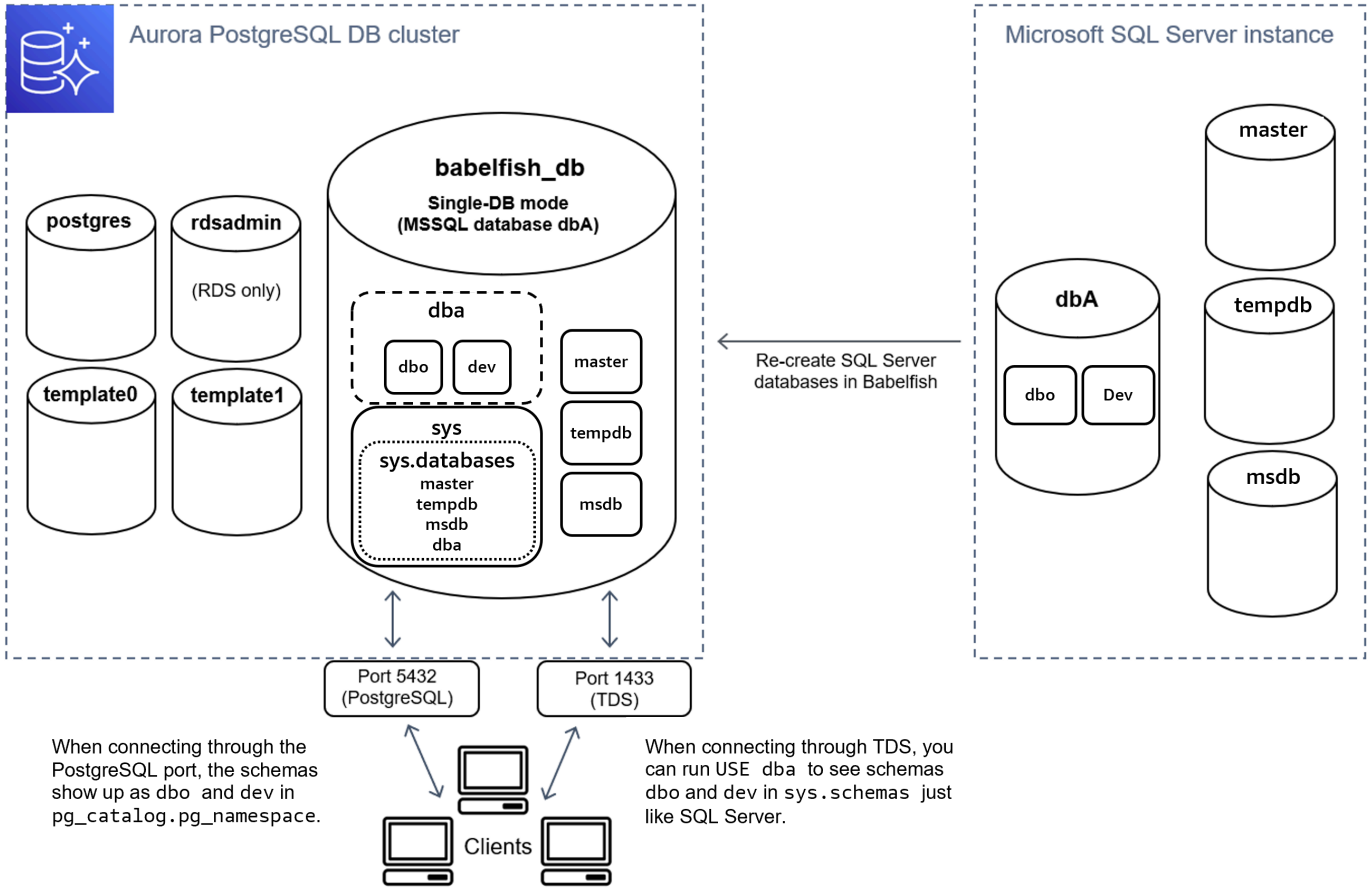
Al conectarse al puerto TDS, la sesión se coloca en la base de datos `babelfish_db`. En T-SQL, la estructura tiene un aspecto similar a estar conectado a una instancia de SQL Server. Puede ver las bases de datos `master`, `msdb` y `tempdb`, y el catálogo `sys.databases`. Puede crear bases de datos de usuario adicionales y cambiar entre bases de datos con la instrucción USE. Cuando se crea una base de datos de usuario de SQL Server, se aplana en la base de datos `babelfish_db` de PostgreSQL. La base de datos retiene la sintaxis y la semántica entre bases de datos iguales o similares a las proporcionadas por SQL Server.

Uso de Babelfish con una base de datos única o varias bases de datos

Al crear un clúster de Aurora PostgreSQL para usarlo con Babelfish, elija entre utilizar una sola base de datos de SQL Server por sí sola o varias bases de datos de SQL Server juntas. Su elección afecta a la manera en la que aparecen los nombres de los esquemas de SQL Server dentro de la base de datos `babelfish_db` en Aurora PostgreSQL. El modo de migración se almacena en el parámetro `migration_mode`. No debe cambiar este parámetro después de crear el clúster, ya que podría perder el acceso a todos los objetos SQL creados anteriormente.

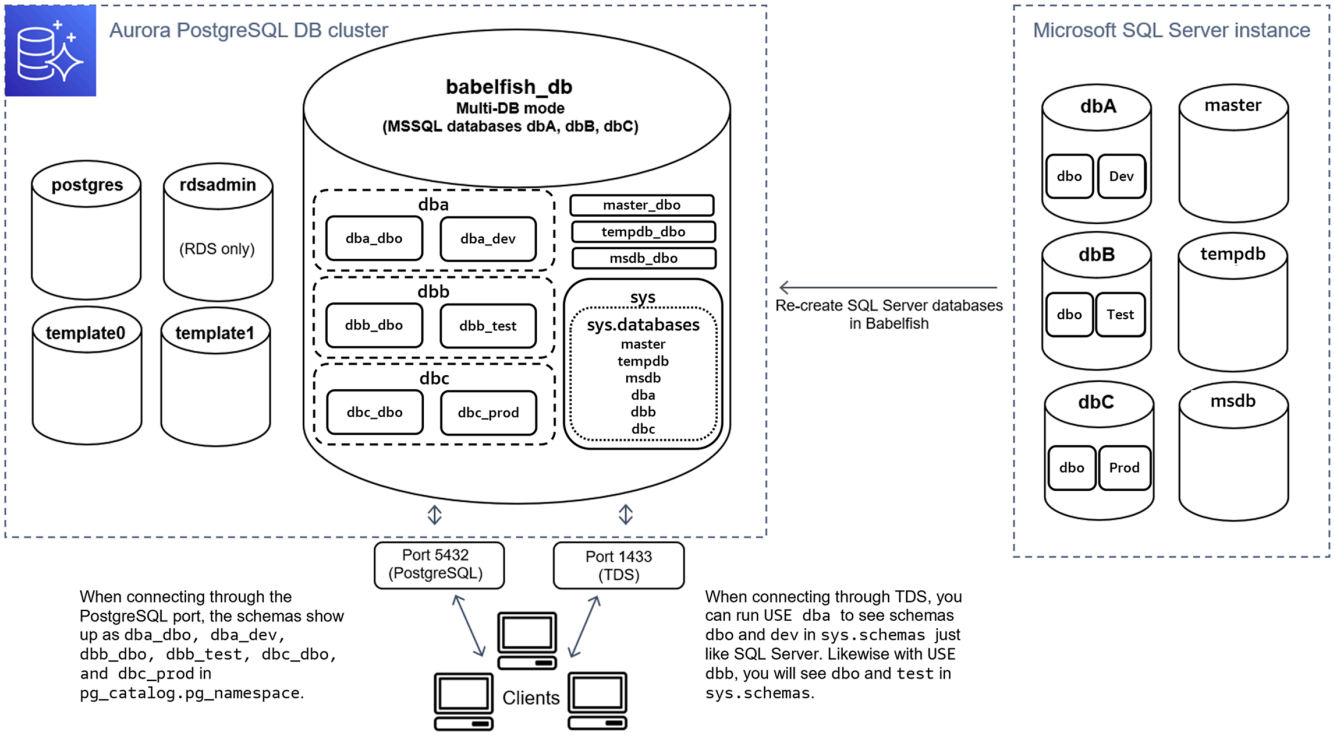
En el modo de una sola base de datos, los nombres de los esquemas de la base de datos de SQL Server siguen siendo los mismos en la base de datos `babelfish_db` de PostgreSQL. Si decide migrar solo una base de datos, se puede hacer referencia a los nombres de esquema de la base de

datos de usuario migrada en PostgreSQL con los mismos nombres que se utilizan en SQL Server. Por ejemplo, los esquemas dbo y smith residen dentro de la base de datos dbA.



Al conectarse a través de TDS, puede ejecutar USE dba para ver los esquemas dbo y dev de T-SQL, como lo haría en SQL Server. Los nombres de esquema sin cambios también están visibles en PostgreSQL.

En el modo de varias bases de datos, los nombres de esquema de las bases de datos de usuario se convierten en dbname_schemaname cuando se accede a ellas desde PostgreSQL. Los nombres de esquema siguen siendo los mismos cuando se accede a ellos desde T-SQL.



Como se muestra en la imagen, el modo de base de datos múltiple y el modo de base de datos única son iguales a los de SQL Server cuando se conectan a través del puerto TDS y utilizan T-SQL. Por ejemplo, `USE dbA` enumera los esquemas `dbo` y `dev` tal como se hace en SQL Server. Los nombres de esquema asignados, tales como `dba_dbo` y `dba_dev`, están visibles en PostgreSQL.

Cada base de datos sigue conteniendo sus esquemas. El nombre de cada base de datos se antepone al nombre del esquema de SQL Server, y se utiliza un guion bajo como delimitador, por ejemplo:

- `dba` contiene `dba_dbo` y `dba_dev`.
- `dbb` contiene `dbb_dbo` y `dbb_test`.
- `dbc` contiene `dbc_dbo` y `dbc_prod`.

Dentro de la base de datos `babelfish_db`, el usuario de T-SQL aún necesita ejecutar `USE dbname` para cambiar el contexto de la base de datos, de modo que la apariencia se mantenga similar a SQL Server.

Elección de un modo de migración

Cada modo de migración tiene ventajas y desventajas. Elija el modo de migración en función del número de bases de datos de usuario que tenga y de sus planes de migración. Después de crear

un clúster para usarlo con Babelfish, no se puede cambiar el modo de migración, ya que se puede perder el acceso a todos los objetos de SQL que se han creado anteriormente. Al elegir un modo de migración, tenga en cuenta los requisitos de las bases de datos de usuario y los clientes.

Al crear un clúster para usarlo con Babelfish, Aurora PostgreSQL crea las bases de datos del sistema, `master` y `tempdb`. Si ha creado o modificado objetos en las bases de datos del sistema (`master` o `tempdb`), asegúrese de volver a crear esos objetos en el nuevo clúster. A diferencia de SQL Server, Babelfish no reinicializa `tempdb` después de reiniciar el clúster.

Utilice el modo de migración de base de datos única en los siguientes casos:

- Si va a migrar una sola base de datos de SQL Server. En el modo de base de datos única, los nombres de esquema migrados a los que se accede desde PostgreSQL son idénticos a los nombres de esquema de SQL Server originales. Esto reduce los cambios de código en las consultas SQL existentes si desea optimizarlas para que se ejecuten con una conexión de PostgreSQL.
- Si su objetivo final es una migración completa de forma nativa a Aurora PostgreSQL. Antes de migrar, consolide los esquemas en un solo esquema (dbo) y, a continuación, migre a un solo clúster para reducir los cambios necesarios.

Utilice el modo de migración de varias bases de datos en los siguientes casos:

- Si desea disfrutar de la experiencia predeterminada de SQL Server con varias bases de datos de usuarios en la misma instancia.
- Si es necesario migrar varias bases de datos de usuarios juntas.

Configuración del grupo de parámetros del clúster de base de datos para Babelfish

Cuando crea un clúster de base de datos Aurora PostgreSQL y elige Turn on Babelfish (Activar Babelfish), se crea automáticamente un grupo de parámetros de clúster de base de datos si elige Create new (Crear nuevo). Este grupo de parámetros del clúster de base de datos se basa en el grupo de parámetros del clúster de base de datos de Aurora PostgreSQL para la versión de Aurora PostgreSQL elegida para la instalación, por ejemplo, la versión 14 de Aurora PostgreSQL. Se le asigna un nombre mediante el siguiente patrón general:

```
custom-aurora-postgresql14-babelfish-compat-3
```

Puede cambiar la siguiente configuración durante el proceso de creación del clúster, pero algunas opciones no pueden cambiarse una vez que se han almacenado en el grupo de parámetros personalizados, así que elija con cuidado:

- Base de datos única o bases de datos múltiples
- Configuración regional de intercalación predeterminada
- Nombre de intercalación
- Grupo de parámetros de base de datos

Para utilizar un grupo de parámetros existente del clúster de base de datos de Aurora PostgreSQL versión 13 o superior, edite el grupo y establezca el parámetro `babelfish_status` como on. Especifique cualquier opción de Babelfish antes de crear el clúster de Aurora PostgreSQL. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.

Los siguientes parámetros controlan las preferencias de Babelfish. A menos que se indique lo contrario en la descripción, los parámetros se pueden modificar. El valor predeterminado se incluye en la descripción. Para ver los valores permitidos de cualquier parámetro, haga lo siguiente:

> **Note**
>
> Al asociar un nuevo grupo de parámetros de base de datos con una instancia de base de datos, los parámetros estáticos y dinámicos modificados se aplican solo después de reiniciar la instancia de base de datos. Sin embargo, si modifica los parámetros dinámicos en el grupo de parámetros de base de datos después de asociarlos a la instancia de base de datos, dichos cambios se aplican inmediatamente sin reiniciar.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Parameter groups (Grupos de parámetros) en el panel de navegación.
3. Elija el grupo de parámetros del clúster de base de datos `default.aurora-postgresql14` en la lista.
4. Ingrese el nombre de un parámetro en el campo de búsqueda. Por ejemplo, escriba `babelfishpg_tsql.default_locale` en el campo de búsqueda para mostrar este parámetro, además de su valor predeterminado y su configuración permitida.

> **Note**
>
> Las bases de datos globales de Babelfish para Aurora PostgreSQL solo funcionan en regiones secundarias si los siguientes parámetros están activados en esas regiones.

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tsql.apg_enable_correlated_scalar_transform	Permite al planificador transformar una subconsulta escalar correlacionada en Babelfish. (Valor predeterminado: on) (Valor permitido: on, off)	dynamic	true
babelfishpg_tsql.apg_enable_subquery_cache	Permite el uso de la memoria caché para subconsultas escalares correlacionadas en Babelfish. (Valor predeterminado: on) (Valor permitido: on, off)	dynamic	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tds.tds_default_numeric_scale	Establece la escala predeterminada de tipo numérico que se va a enviar en los metadatos de la columna TDS si el motor no lo especifica. (Valor predeterminado: 8; permitido: 0 a 38).	dynamic	true
babelfishpg_tds.tds_default_numeric_precision	Un entero que establece la precisión predeterminada del tipo numérico que se va a enviar en los metadatos de la columna TDS si el motor no lo especifica. (Valor predeterminado: 38; permitido: 1 a 38).	dynamic	true
babelfishpg_tds.tds_default_packet_size	Un entero que establece el tamaño de paquete predeterminado para conectar clientes de SQL Server. (Valor predeterminado: 4096; permitido: 512 a 32767).	dynamic	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tds.tds_default_protocol_version	Un entero que establece una versión de protocolo de TDS predeterminada para conectar clientes. (Valor predeterminado: DEFAULT; permitido: TDSv7.0, TDSv7.1, TDSv7.1.1, TDSv7.2, TDSv7.3A, TDSv7.3B, TDSv7.4, DEFAULT).	dynamic	true
babelfishpg_tds.default_server_name	Una cadena que identifica el nombre predeterminado del servidor de Babelfish. (Valor predeterminado: Microsoft SQL Server; permitido: nulo).	dynamic	true
babelfishpg_tds.tds_debug_log_level	Un entero que establece el nivel de registro de TDS; 0 desactiva el registro. (Valor predeterminado: 1; permitido: 0, 1, 2, 3).	dynamic	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tds.listen_addresses	Una cadena que establece el nombre de host o las direcciones IP en las que se escuchará TDS. Este parámetro no se puede modificar una vez creado el clúster de base de datos de Babelfish. (Valor predeterminado: *; permitido: nulo).	–	false
babelfishpg_tds.port	Un entero que especifica el puerto TCP utilizado para las solicitudes en la sintaxis de SQL Server. (Valor predeterminado: 1433; permitido: 1 a 65535).	estática	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tds.tds_ssl_encrypt	Un booleano que activa el cifrado (0) o lo desactiva (1) para los datos que recorren el puerto del agente de escucha de TDS. Para obtener información detallada sobre el uso de SSL para las conexiones de clientes, consulte Configuración SSL de Babelfish y conexiones de cliente. (Valor predeterminado: 0; permitido: 0, 1).	dynamic	true
babelfishpg_tds.tds_ssl_max_protocol_version	Una cadena que especifica la versión máxima del protocolo SSL/TLS que se utilizará para la sesión TDS. (Valor predeterminado: "TLSv1.2"; permitido: "TLSv1", "TLSv1.1", "TLSv1.2")	dynamic	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tds.tds_ssl_min_protocol_version	Una cadena que especifica la versión mínima del protocolo SSL/TLS que se utilizará para la sesión TDS. (Predeterminado: “TLSv1.2” de Aurora PostgreSQL versión 16, “TLSv1” para versiones anteriores a Aurora PostgreSQL versión 16) (Permisible: “TLSv1”, “TLSv1.1”, “TLSv1.2”)	dynamic	true
babelfishpg_tds.unix_socket_directories	Una cadena que identifica el directorio socket Unix del servidor TDS. Este parámetro no se puede modificar una vez creado el clúster de base de datos de Babelfish. (Valor predeterminado: /tmp; permitido: nulo).	–	false

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tds.unix_socket_group	Una cadena que identifica el grupo socket Unix del servidor TDS. Este parámetro no se puede modificar una vez creado el clúster de base de datos de Babelfish. (Valor predeterminado: rdsdb; permitido: nulo).	–	false

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tsql.default_locale	Una cadena que especifica la configuración regional predeterminada que se utiliza para las intercalaciones de Babelfish. La configuración regional predeterminada es solo la configuración regional y no incluye ningún calificador. Establezca este parámetro cuando aprovisione un clúster de bases de datos de Babelfish. Después de aprovisionar el clúster de base de datos, se ignoran los cambios de este parámetro. (Valor predeterminado: en_US; permitido: consulte las tablas).	estática	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tsql.migration_mode	Una lista no modificable que especifica la compatibilidad con bases de datos de uno o varios usuarios. Establezca este parámetro cuando aprovisione un clúster de bases de datos de Babelfish. Después de aprovisionar el clúster de base de datos, no puede modificar el valor de este parámetro. (Valor predeterminado: multi-db desde Aurora PostgreSQL versión 16, single-db para versiones anteriores a la versión 16 de Aurora PostgreSQL) (Permitido: single-db, multi-db, null).	estática	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
babelfishpg_tsql.server_collation_name	Una cadena que especifica el nombre de la intercalación utilizada para acciones de nivel de servidor. Establezca este parámetro cuando aprovisione un clúster de bases de datos de Babelfish. Después de aprovisionar el clúster de bases de datos, no modifique el valor de este parámetro. (Valor predeterminado: bbf_unicode_general_ci_as; permitido: consulte las tablas).	estática	true
babelfishpg_tsql.version	Una cadena que establece la salida de la variable @@VERSION. No modifique este valor para clústeres de base de datos de Aurora PostgreSQL. (Valor predeterminado: nulo; permitido: default).	dynamic	true

Parámetro	Descripción	Tipo de aplicación	Es modificable
rds.babelfish_status	Una cadena que establece el estado de la funcionalidad de Babelfish. Cuando este parámetro se establece en `datatypesonly` , Babelfish se desactiva , pero los tipos de datos de SQL Server siguen disponibles. (Valor predeterm inado: off; permitido : on, off, datatypes only).	estática	true
unix_socket_permis sions	Un entero que establece los permisos de socket Unix del servidor de TDS. Este parámetro no se puede modificar una vez creado el clúster de base de datos de Babelfish . (Valor predeterm inado: 0700; permitido : 0 a 511).	–	false

Configuración SSL de Babelfish y conexiones de cliente

Para exigir conexiones SSL/TLS al clúster de base de datos de Babelfish para Aurora PostgreSQL, use el parámetro `rds.force_ssl`.

- Para requerir conexiones SSL/TLS, establezca el valor del parámetro `rds.force_ssl` en 1 (activado).
- Para desactivar las conexiones SSL/TLS requeridas, establezca el valor del parámetro `rds.force_ssl` en 0 (desactivado).

El valor predeterminado de este parámetro depende de la versión de Aurora PostgreSQL:

- Para las versiones 17 y posteriores de Aurora PostgreSQL: el valor predeterminado es 1 (activado).
- Para las versiones 16 y más antiguas de Aurora PostgreSQL: el valor predeterminado es 0 (desactivado).

> **Note**
>
> Cuando realiza una actualización de versión principal de Aurora PostgreSQL versión 16 o anterior a la versión 17 o posterior, el valor predeterminado del parámetro cambia de 0 (desactivado) a 1 (activado). Este cambio puede provocar errores de conectividad en las aplicaciones que no estén configuradas para SSL. Puede volver al comportamiento predeterminado anterior estableciendo este parámetro en 0 (desactivado).

Para obtener información específica sobre el controlador, consulte Conexión a un clúster de base de datos de Babelfish.

Cuando un cliente se conecta al puerto TDS (predeterminado: `1433`), Babelfish compara la configuración de la capa de sockets seguros (SSL) enviada durante el protocolo de enlace del cliente con la configuración de los parámetros SSL de Babelfish (`tds_ssl_encrypt`). A continuación, Babelfish determina si se permite una conexión. Si se permite una conexión, el comportamiento de cifrado se aplica o no, según la configuración de los parámetros y la compatibilidad con el cifrado ofrecido por el cliente.

En la tabla siguiente se muestra cómo se comporta Babelfish para cada combinación.

Configuración de SSL del cliente	Configuración de SSL de Babelfish	rds.force_ssl	¿Se permite la conexión?	Valor devuelto al cliente
ENCRYPT_ON	Cualquiera	Cualquiera	Permitido, toda la conexión está cifrada	ENCRYPT_ON
ENCRYPT_OFF	tds_ssl_encrypt=1	Cualquiera	Permitido, toda la conexión está cifrada	ENCRYPT_REQ
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=0	Permitido, el paquete de inicio de sesión está cifrado	ENCRYPT_OFF
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=1	No, conexión cerrada	ENCRYPT_OFF
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=0	Sí	ENCRYPT_NOT_SUP
ENCRYPT_NOT_SUP	tds_ssl_encrypt=1	Cualquiera	No, conexión cerrada	ENCRYPT_REQ
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=1	No, conexión cerrada	ENCRYPT_NOT_SUP
ENCRYPT_CLIENT_CERT	Cualquiera	Cualquiera	No, conexión cerrada	No se admite

Comprensión de las intercalaciones de Babelfish para Aurora PostgreSQL

Cuando crea un clúster de base de datos de Aurora PostgreSQL con Babelfish, elige una intercalación para sus datos. Una intercalación especifica el orden de clasificación y los patrones de bits que producen el texto o los caracteres en un determinado lenguaje humano escrito. Una intercalación incluye reglas que comparan los datos para un conjunto determinado de patrones de bits. La intercalación está relacionada con la localización. Las diferentes configuraciones regionales afectan a la asignación de caracteres, el orden de clasificación y otros aspectos similares. Los atributos de intercalación se reflejan en los nombres de varias intercalaciones. Para obtener información sobre los atributos, consulte la Babelfish collation attributes table.

Babelfish asigna las intercalaciones de SQL Server a intercalaciones comparables proporcionadas por Babelfish. Babelfish predefine las intercalaciones Unicode con comparaciones de cadenas y órdenes de clasificación sensibles a la cultura. Babelfish también proporciona una manera de traduce las intercalaciones de la base de datos de SQL Server a la intercalación de Babelfish más cercana. Se proporcionan intercalaciones específicas para cada configuración regional en diferentes idiomas y regiones.

Algunas intercalaciones especifican una página de códigos que corresponde a una codificación del lado del cliente. Babelfish traduce automáticamente de la codificación del servidor a la codificación del cliente en función de la intercalación de cada columna de salida.

Babelfish admite las intercalaciones indicadas en la Babelfish supported collations table. Babelfish asigna las intercalaciones de SQL Server a intercalaciones comparables proporcionadas por Babelfish.

Babelfish utiliza la versión 153.80 de la biblioteca de intercalación de Componentes internacionales para Unicode (ICU). Para obtener más información sobre las intercalaciones de ICU, consulte Collation (Intercalación) en la documentación de ICU. Para obtener más información sobre PostgreSQL y la intercalación, consulte Collation Support (Compatibilidad con intercalación) en la documentación de PostgreSQL.

Temas

- Parámetros de clúster de base de datos que controlan la intercalación y la configuración regional
- Intercalaciones deterministas y no deterministas en Babelfish
- Intercalaciones admitidas por Babelfish en el nivel de base de datos
- Intercalaciones de servidores y objetos en Babelfish
- Comportamiento predeterminado de las intercalaciones en Babelfish

- Administración de intercalaciones
- Limitaciones de intercalación y diferencias de comportamiento

Parámetros de clúster de base de datos que controlan la intercalación y la configuración regional

Los siguientes parámetros afectan al comportamiento de intercalación.

babelfishpg_tsql.default_locale

Este parámetro especifica la configuración regional predeterminada que usa la intercalación. Este parámetro se utiliza en combinación con los atributos enumerados en la Babelfish collation attributes table para personalizar las intercalaciones de un idioma y una región específicos. El valor predeterminado para este parámetro es `en-US`.

La configuración regional predeterminada se aplica a todos los nombres de intercalación de Babelfish que empiezan por "BBF" y a todas las intercalaciones de SQL Server asignadas a intercalaciones de Babelfish. El cambio de la configuración de este parámetro en un clúster de base de datos de Babelfish existente no afecta a la configuración regional de las intercalaciones existentes. Para ver la lista de intercalaciones, consulte la Babelfish supported collations table.

babelfishpg_tsql.server_collation_name

Este parámetro especifica la intercalación predeterminada para el servidor (instancia de clúster de base de datos de Aurora PostgreSQL) y la base de datos. El valor predeterminado es `sql_latin1_general_cp1_ci_as`. `server_collation_name` tiene que ser una intercalación `CI_AS` porque en T-SQL, la intercalación del servidor determina cómo se comparan los identificadores.

Al crear su clúster de base de datos Babelfish, puede elegir el valor de Collation name (Nombre de la intercalación) en la lista seleccionable. Se incluyen las intercalaciones enumeradas en la Babelfish supported collations table. No modifique `server_collation_name` después de crear la base de datos de Babelfish.

La configuración que elija al crear el clúster de la base de datos de Babelfish para Aurora PostgreSQL se almacena en el grupo de parámetros del clúster de base de datos asociado con el clúster para estos parámetros y establece su comportamiento de intercalación.

Intercalaciones deterministas y no deterministas en Babelfish

Babelfish admite las combinaciones deterministas y no deterministas:

- Una intercalación determinista evalúa como iguales los caracteres que tienen secuencias de bytes idénticas. Eso significa que `x` y `X` no son iguales en una intercalación determinista. Las intercalaciones deterministas pueden distinguir mayúsculas de minúsculas (CS) y acentos (AS).
- Una intercalación no determinista no necesita una coincidencia idéntica. Una intercalación no determinista evalúa `x` y `X` como iguales. Las intercalaciones no deterministas no distinguen mayúsculas de minúsculas (IC) ni acentos (IA).

En la siguiente tabla puede encontrar algunas diferencias de comportamiento entre Babelfish y PostgreSQL cuando se utilizan intercalaciones no deterministas.

Babelfish	PostgreSQL
Admite la cláusula LIKE para las intercala ciones CI_AS.	No admite la cláusula LIKE en las intercala ciones no deterministas.
Solo admite la cláusula LIKE en las siguiente s intercalaciones de IA y a partir de la versión 4.2.0 de Babelfish: • bbf_unicode_cp1250_ci_ai • bbf_unicode_cp1250_cs_ai • bbf_unicode_cp1257_ci_ai • bbf_unicode_cp1257_cs_ai • bbf_unicode_cp1_ci_ai • bbf_unicode_cp1_cs_ai • estonian_ci_ai • finnish_swedish_ci_ai • french_ci_ai • latin1_general_ci_ai • latin1_general_cs_ai • modern_spanish_ci_ai • polish_ci_ai • sql_latin1_general_cp1_ci_ai • sql_latin1_general_cp1_cs_ai	No admite la cláusula LIKE en las intercala ciones no deterministas.

Babelfish	PostgreSQL
• traditional_spanish_ci_ai	

Para obtener una lista de otras limitaciones y diferencias de comportamiento de Babelfish en comparación con SQL Server y PostgreSQL, consulte Limitaciones de intercalación y diferencias de comportamiento.

Babelfish y SQL Server siguen una convención de nomenclatura para las intercalaciones que describen los atributos de intercalación, como se muestra en la tabla siguiente.

Atributo	Descripción
IA	No distingue acentos.
AS	Distingue acentos.
BIN2	BIN2 solicita que los datos se clasifiquen en orden de puntos de código. El orden de puntos de código Unicode es el mismo orden de caracteres para las codificaciones UTF-8, UTF-16 y UCS-2. El orden de los puntos de código es una intercalación determinista rápida.
CI	No distingue mayúsculas de minúsculas.
CS	Distingue mayúsculas de minúsculas.
PREF	Para ordenar las letras mayúsculas antes que las minúsculas, utilice una intercalación de PREF. Si la comparación no distingue mayúsculas de minúsculas, la versión en mayúscula de una letra se ordena antes que la versión en minúscula, si no hay otra distinción. La biblioteca de ICU admite preferencias en mayúsculas con `colCaseFirst=upper` , pero no para las intercalaciones CI_AS. PREF solo se puede aplicar a las intercalaciones `CS_AS` deterministas.

Intercalaciones admitidas por Babelfish en el nivel de base de datos

Babelfish admite las siguientes intercalaciones en el nivel de base de datos:

- bbf_unicode_bin2
- bbf_unicode_cp1_ci_ai
- bbf_unicode_cp1_ci_as
- bbf_unicode_cp1250_ci_ai
- bbf_unicode_cp1250_ci_as
- bbf_unicode_cp1257_ci_ai
- bbf_unicode_cp1257_ci_as
- estonian_ci_ai
- estonian_ci_as
- finnish_swedish_ci_ai
- finnish_swedish_ci_as
- french_ci_ai
- french_ci_as
- latin1_general_bin2
- latin1_general_ci_ai
- latin1_general_ci_as
- latin1_general_90_bin2
- latin1_general_100_bin2
- latin1_general_140_bin2
- modern_spanish_ci_ai
- modern_spanish_ci_as
- polish_ci_ai
- polish_ci_as
- sql_latin1_general_cp1_ci_ai
- sql_latin1_general_cp1_ci_as
- sql_latin1_general_cp1250_ci_as
- sql_latin1_general_cp1251_ci_as
- sql_latin1_general_cp1257_ci_as
- traditional_spanish_ci_ai
- traditional_spanish_ci_as

> **Note**
>
> Para usar una intercalación diferente en el nivel de base de datos, asegúrese de que coincida con la intercalación del nivel de servidor. Para obtener más información, consulte Intercalaciones de servidores y objetos en Babelfish

Intercalaciones de servidores y objetos en Babelfish

Utilice las siguientes intercalaciones como intercalación de servidores o intercalación de objetos.

ID de intercalación	Notas
bbf_unicode_general_ci_as	Admite la comparación sin distinción de mayúsculas y minúsculas y el operador LIKE.
bbf_unicode_cp1_ci_as	Intercalación no determinista también conocido como CP1252.
bbf_unicode_CP1250_ci_as	Intercalación no determinista utilizada para representar textos en idiomas de Europa Central y Europa del Este que utilizan escritura latina.
bbf_unicode_CP1251_ci_as	Intercalación no determinista para los idiomas que utilizan el alfabeto cirílico.
bbf_unicode_cp1253_ci_as	Intercalación no determinista utilizada para representar el griego moderno.
bbf_unicode_cp1254_ci_as	Intercalación no determinista que admite el turco.
bbf_unicode_cp1255_ci_as	Intercalación no determinista que admite el hebreo.
bbf_unicode_cp1256_ci_as	Intercalación no determinista utilizada para escribir idiomas que utilizan escritura árabe.

ID de intercalación	Notas
bbf_unicode_cp1257_ci_as	Intercalación no determinista utilizada para admitir los idiomas estonio, letón y lituano.
bbf_unicode_cp1258_ci_as	Intercalación no determinista utilizada para escribir caracteres vietnamitas.
bbf_unicode_cp874_ci_as	Intercalación no determinista utilizada para escribir caracteres tailandeses.
sql_latin1_general_cp1250_ci_as	Codificación de caracteres de un solo byte no determinista utilizada para representar caracteres latinos.
sql_latin1_general_cp1251_ci_as	Intercalación no determinista que admite caracteres latinos.
sql_latin1_general_cp1_ci_as	Intercalación no determinista que admite caracteres latinos.
sql_latin1_general_cp1253_ci_as	Intercalación no determinista que admite caracteres latinos.
sql_latin1_general_cp1254_ci_as	Intercalación no determinista que admite caracteres latinos.
sql_latin1_general_cp1255_ci_as	Intercalación no determinista que admite caracteres latinos.
sql_latin1_general_cp1256_ci_as	Intercalación no determinista que admite caracteres latinos.

ID de intercalación	Notas
sql_latin1_general_cp1257_ci_as	Intercalación no determinista que admite caracteres latinos.
sql_latin1_general_cp1258_ci_as	Intercalación no determinista que admite caracteres latinos.
chinese_prc_ci_as	Intercalación no determinista que admite chino (RPC).
cyrillic_general_ci_as	Intercalación no determinista que admite el cirílico.
finnish_swedish_ci_as	Intercalación no determinista que admite el finlandés.
french_ci_as	Intercalación no determinista que admite el francés.
Japanese_CI_AS	Intercalación no determinista que admite japonés. Se admite en Babelfish 2.1.0 y versiones posteriores.
korean_wansung_ci_as	Intercalación no determinista que admite el coreano (con clasificación de diccionario).
latin1_general_ci_as	Intercalación no determinista que admite caracteres latinos.
modern_spanish_ci_as	Intercalación no determinista que admite el español moderno.
polish_ci_as	Intercalación no determinista que admite el polaco.
thai_ci_as	Intercalación no determinista que admite el tailandés.
traditional_spanish_ci_as	Intercalación no determinista que admite el español (clasificación tradicional).

ID de intercalación	Notas
turkish_ci_as	Intercalación no determinista que admite el turco.
ukrainian_ci_as	Intercalación no determinista que admite el ucraniano.
vietnamese_ci_as	Intercalación no determinista que admite el vietnamita.

Puede utilizar las siguientes intercalaciones como intercalaciones de objetos.

Dialecto	Opciones deterministas	Opciones no deterministas
Árabe	Arabic_CS_AS	Arabic_CI_AS Arabic_CI_AI
Alfabeto árabe	BBF_Unicode_CP1256_CS_AS BBF_Unicode_Pref_CP1256_CS_AS	BBF_Unicode_CP1256_CI_AI BBF_Unicode_CP1256_CS_AI
Binario	latin1_general_bin2 BBF_Unicode_BIN2	–
Idiomas de Europa Central y Europa del Este que utilizan escritura latina	BBF_Unicode_CP1250_CS_AS BBF_Unicode_Pref_CP1250_CS_AS	BBF_Unicode_CP1250_CI_AI BBF_Unicode_CP1250_CS_AI
Chino	Chinese_PRC_CS_AS	Chinese_PRC_CI_AS Chinese_PRC_CI_AI

Dialecto	Opciones deterministas	Opciones no deterministas
Cyrillic_General	Cyrillic_General_CS_AS	Cyrillic_General_CI_AS Cyrillic_General_CI_AI
Alfabeto cirílico	BBF_Unicode_CP1251_CS_AS BBF_Unicode_Pref_CP1251_CS_AS	BBF_Unicode_CP1251_CI_AI BBF_Unicode_CP1251_CS_AI
Estonio	Estonian_CS_AS	Estonian_CI_AS Estonian_CI_AI
Estonio, letón y lituano	BBF_Unicode_CP1257_CS_AS BBF_Unicode_Pref_CP1257_CS_AS	BBF_Unicode_CP1257_CI_AI BBF_Unicode_CP1257_CS_AI
Finnish_Swedish	Finnish_Swedish_CS_AS	Finnish_Swedish_CI_AS Finnish_Swedish_CI_AI
Francés	French_CS_AS	French_CI_AS French_CI_AI
Griego	Greek_CS_AS	Greek_CI_AS Greek_CI_AI
Hebreo	BBF_Unicode_CP1255_CS_AS BBF_Unicode_Pref_CP1255_CS_AS Hebrew_CS_AS	BBF_Unicode_CP1255_CI_AI BBF_Unicode_CP1255_CS_AI Hebrew_CI_AS Hebrew_CI_AI

Dialecto	Opciones deterministas	Opciones no deterministas
Japonés (Babelfish 2.1.0 y posterior)	Japanese_CS_AS	Japanese_CI_AI Japanese_CI_AS
Korean_Wamsung	Korean_Wamsung_CS_AS	Korean_Wamsung_CI_AS Korean_Wamsung_CI_AI
Caracteres latinos para la página de códigos CP1252	latin1_general_cs_as BBF_Unicode_General_CS_AS BBF_Unicode_General_Pref_CS_AS BBF_Unicode_Pref_CP1_CS_AS BBF_Unicode_CP1_CS_AS	latin1_general_ci_as latin1_general_ci_ai latin1_general_cs_ai BBF_Unicode_General_CI_AI BBF_Unicode_General_CS_AI BBF_Unicode_CP1_CI_AI BBF_Unicode_CP1_CS_AI
Griego moderno	BBF_Unicode_CP1253_CS_AS BBF_Unicode_Pref_CP1253_CS_AS	BBF_Unicode_CP1253_CI_AI BBF_Unicode_CP1253_CS_AI
Modern_Spanish	Modern_Spanish_CS_AS	Modern_Spanish_CI_AS Modern_Spanish_CI_AI
Mongol	Mongolian_cs_as	Mongolian_CI_AS Mongolian_CI_AI

Dialecto	Opciones deterministas	Opciones no deterministas
Polaco	Polish_CS_AS	Polish_CI_AS Polish_CI_AI
Tailandés	BBF_Unicode_CP874_CS_AS BBF_Unicode_Pref_CP874_CS_AS Thai_cs_as	BBF_Unicode_CP874_CI_AI BBF_Unicode_CP874_CS_AI Thai_ci_as, Thai_ci_ai
Traditional_Spanish	Traditional_Spanish_CS_AS	Traditional_Spanish_CI_AS Traditional_Spanish_CI_AI
Turco	BBF_Unicode_CP1254_CS_AS BBF_Unicode_Pref_CP1254_CS_AS Turkish_CS_AS	BBF_Unicode_CP1254_CI_AI BBF_Unicode_CP1254_CS_AI Turkish_ci_as, turco_ci_ai
Ucraniano	Ukranian_CS_AS	Ukranian_CI_AS Ukranian_CI_AI
Vietnamita	BBF_Unicode_CP1258_CS_AS BBF_Unicode_Pref_CP1258_CS_AS Vietnamese_CS_AS	BBF_Unicode_CP1258_CI_AI BBF_Unicode_CP1258_CS_AI Vietnamese_CI_AS Vietnamese_CI_AI

Comportamiento predeterminado de las intercalaciones en Babelfish

Anteriormente, la intercalación predeterminada de los tipos de datos recopilables era `pg_catalog.default`. Los tipos de datos y los objetos que dependen de estos tipos de datos siguen una intercalación que distingue entre mayúsculas y minúsculas. Esta condición puede afectar a los objetos T-SQL del conjunto de datos, ya que la intercalación no distingue entre mayúsculas y minúsculas. A partir de Babelfish 2.3.0, la intercalación predeterminada para los tipos de datos recopilables (excepto TEXT y NTEXT) es la misma que la intercalación en el parámetro `babelfishpg_tsql.server_collation_name`. Al actualizar a Babelfish 2.3.0, la intercalación predeterminada se selecciona automáticamente en el momento de la creación del clúster de base de datos, lo que no tiene ningún impacto visible.

Administración de intercalaciones

La biblioteca de ICU proporciona seguimiento de versiones de intercalación para garantizar que los índices que dependen de las intercalaciones se puedan volver a indexar cuando esté disponible una nueva versión de ICU. Para comprobar si su base de datos actual tiene intercalaciones que se deban actualizar, puede utilizar la siguiente consulta después de conectarse mediante `psql` o `pgAdmin`:

```
SELECT pg_describe_object(refclassid, refobjid,
    refobjsubid) AS "Collation",
    pg_describe_object(classid, objid, objsubid) AS "Object"
    FROM pg_depend d JOIN pg_collation c ON refclassid = 'pg_collation'::regclass
    AND refobjid = c.oid WHERE c.collversion <> pg_collation_actual_version(c.oid)
    ORDER BY 1, 2;
```

Esta consulta devuelve resultados como los siguientes:

```
 Collation | Object
-----------+--------
(0 rows)
```

En este ejemplo, no es necesario actualizar ninguna intercalación.

Para obtener un listado de las intercalaciones predefinidas en su base de datos Babelfish, puede utilizar `psql` o `pgAdmin` con la siguiente consulta:

```
SELECT * FROM pg_collation;
```

Las intercalaciones predefinidas se almacenan en la tabla `sys.fn_helpcollations`. Puede utilizar el siguiente comando para mostrar información sobre una intercalación (como sus indicadores lcid, estilo y marcas de collate). Para obtener un listado de todas las intercalaciones mediante `sqlcmd`, conéctese al puerto T-SQL (1433, de forma predeterminada) y ejecute la siguiente consulta:

```
1> :setvar SQLCMDMAXVARTYPEWIDTH 40
2> :setvar SQLCMDMAXFIXEDTYPEWIDTH 40
3> SELECT * FROM fn_helpcollations()
4> GO
name                                  description
------------------------------------- ---------------------------------------
arabic_cs_as                          Arabic, case-sensitive, accent-sensitive
arabic_ci_ai                          Arabic, case-insensitive, accent-insensi
arabic_ci_as                          Arabic, case-insensitive, accent-sensiti
bbf_unicode_bin2                      Unicode-General, case-sensitive, accent-
bbf_unicode_cp1250_ci_ai              Default locale, code page 1250, case-ins
bbf_unicode_cp1250_ci_as              Default locale, code page 1250, case-ins
bbf_unicode_cp1250_cs_ai              Default locale, code page 1250, case-sen
bbf_unicode_cp1250_cs_as              Default locale, code page 1250, case-sen
bbf_unicode_pref_cp1250_cs_as         Default locale, code page 1250, case-sen
bbf_unicode_cp1251_ci_ai              Default locale, code page 1251, case-ins
bbf_unicode_cp1251_ci_as              Default locale, code page 1251, case-ins
bbf_unicode_cp1254_ci_ai              Default locale, code page 1254, case-ins
...
(124 rows affected)
```

Las líneas 1 y 2 que aparecen en el ejemplo reducen la salida solo a efectos de legibilidad de la documentación.

```
1> SELECT SERVERPROPERTY('COLLATION')
2> GO
serverproperty
------------------------------------------------------------------------
sql_latin1_general_cp1_ci_as

(1 rows affected)
1>
```

Limitaciones de intercalación y diferencias de comportamiento

Babelfish utiliza la biblioteca de ICU para admitir la intercalación. PostgreSQL se creó con una versión específica de ICU y puede coincidir como máximo con una versión de una intercalación.

Las variaciones entre versiones son inevitables, al igual que las variaciones menores en el tiempo a medida que evolucionan los lenguajes. En la siguiente lista puede encontrar las limitaciones conocidas y las variaciones de comportamiento de las intercalaciones de Babelfish:

- Índices y dependencia del tipo de intercalación: un índice de un tipo definido por el usuario que depende de la biblioteca de intercalación de Componentes internacionales para Unicode (ICU), la biblioteca utilizada por Babelfish, no se invalida cuando se cambia la versión de la biblioteca.
- Función COLLATIONPROPERTY: las propiedades de intercalación solo se implementan para las intercalaciones de BBF admitidas de Babelfish. Para obtener más información, consulte Babelfish supported collations table.
- Diferencias de las reglas de orden de Unicode: las intercalaciones de SQL Server ordenan los datos codificados en Unicode (`nchar` y `nvarchar`) de una manera distinta a los datos que no están codificados en Unicode (`char` y `varchar`). Las bases de datos de Babelfish siempre están codificadas en UTF-8 y siempre aplican reglas de orden Unicode de manera coherente, independientemente del tipo de datos, por lo que el orden de clasificación para `char` o `varchar` es el mismo que para `nchar` o `nvarchar`.
- Intercalaciones secundarias iguales y comportamiento de orden: la intercalación secundaria de igualdad Unicode de ICU predeterminada (`CI_AS`) ordena los signos de puntuación y otros caracteres no alfanuméricos antes que los caracteres numéricos y los caracteres numéricos antes que los caracteres alfabéticos. Sin embargo, el orden de puntuación y otros caracteres especiales son diferentes.
- Intercalaciones terciarias, solución para ORDER BY: las intercalaciones de SQL, tales como `SQL_Latin1_General_Pref_CP1_CI_AS`, admiten la función `TERTIARY_WEIGHTS` y la capacidad de ordenar cadenas que se comparan por igual en una intercalación `CI_AS` que debe ordenarse en mayúsculas primero: `ABC`, `ABc`, `AbC`, `Abc`, `aBC`, `aBc`, `abC` y, por último, `abc`. Por lo tanto, la función de análisis `DENSE_RANK OVER (ORDER BY column)` evalúa estas cadenas como del mismo rango, pero las ordena en mayúscula primero en una partición.

  Puede obtener un resultado similar con Babelfish mediante la adición de una cláusula `COLLATE` a la cláusula `ORDER BY` que especifica una intercalación terciaria `CS_AS` que especifica `@colCaseFirst=upper`. Sin embargo, el modificador `colCaseFirst` se aplica solo a cadenas terciarias iguales (en lugar de secundarias iguales, como sucede con una intercalación `CI_AS`). Por lo tanto, no se pueden emular las intercalaciones de SQL terciarias mediante una única intercalación de ICU.

  Como solución alternativa, le recomendamos que modifique las aplicaciones que utilizan la intercalación `SQL_Latin1_General_Pref_CP1_CI_AS` para utilizar la intercalación

`BBF_SQL_Latin1_General_CP1_CI_AS` en primer lugar. A continuación, agregue `COLLATE BBF_SQL_Latin1_General_Pref_CP1_CS_AS` a cualquier cláusula `ORDER BY` de esta columna.

- Expansión de caracteres: una expansión de caracteres trata a un solo carácter igual que a una secuencia de caracteres del nivel principal. La intercalación `CI_AS` predeterminada de SQL Server admite la expansión de caracteres. Las intercalaciones de UCI admiten la expansión de caracteres solo en las intercalaciones que no distinguen los acentos.

  Cuando sea necesaria la expansión de caracteres, utilice una intercalación `AI` para obtener comparaciones. Sin embargo, el operador LIKE no admite actualmente estas intercalaciones.

- Codificación char y varchar: cuando se utilizan intercalaciones SQL para los tipos de datos `char` o `varchar`, el orden de clasificación de los caracteres anteriores a ASCII 127 viene determinado por la página de códigos específica para esa intercalación de SQL. Para las intercalaciones SQL, las cadenas declaradas como `char` o `varchar` pueden ordenarse de forma diferente a las cadenas declaradas como `nchar` o `nvarchar`.

  PostgreSQL codifica todas las cadenas con la codificación de la base de datos, de modo que se convierten todos los caracteres a UTF-8 y se ordenan mediante reglas Unicode.

  Dado que las intercalaciones de SQL ordenan los tipos de datos nchar y nvarchar mediante reglas Unicode, Babelfish codifica todas las cadenas del servidor mediante UTF-8. Babelfish ordena las cadenas nchar y nvarchar de la misma manera que ordena las cadenas char y varchar mediante las reglas Unicode.

- Carácter suplementario: las funciones `NCHAR`, `UNICODE` y `LEN` de SQL Server admiten caracteres para puntos de código fuera del Plano Básico Multilingüe (BMP) de Unicode. Por el contrario, las intercalaciones que no son SC utilizan caracteres pares sustitutos para gestionar caracteres complementarios. Para los tipos de datos Unicode, SQL Server puede representar hasta 65 535 caracteres mediante UCS-2 o el rango Unicode completo (1,114,114 caracteres) si se utilizan caracteres complementarios.

- Intercalaciones con distinción de los tipos de kana (KS): una intercalación con distinción de los tipos de kana (KS) trata los caracteres kana japoneses `Hiragana` y `Katakana` de forma diferente. ICU admite el estándar de intercalación japonés `JIS X 4061`. El ahora obsoleto modificador `colhiraganaQ [on | off]` de configuración regional podría proporcionar la misma funcionalidad que las intercalaciones de KS. Sin embargo, Babelfish no admite actualmente las intercalaciones de KS del mismo nombre que SQL Server.

- Intercalaciones de distinción de ancho (WS): cuando un carácter de un byte (ancho medio) y el mismo carácter representado como un carácter de doble byte (ancho completo) se tratan de forma diferente, la intercalación se denomina de distinción de ancho (WS). Sin embargo, Babelfish no admite actualmente las intercalaciones de WS del mismo nombre que SQL Server.
- Intercalaciones de distinción de selector de variaciones (VSS): las intercalaciones de distinción de selector de variaciones (VSS) distinguen entre los selectores de variaciones ideográficas en las intercalaciones en japonés `Japanese_Bushu_Kakusu_140` y `Japanese_XJIS_140`. Una secuencia de variaciones se compone de un carácter base más un selector de variaciones adicional. Si no selecciona la opción `_VSS`, el selector de variaciones no se tiene en cuenta en la comparación.

  Babelfish no admite las intercalaciones de VSS.
- Intercalaciones BIN y BIN2: una intercalación BIN2 ordena los caracteres según el orden de los puntos de código. El orden binario byte a byte de UTF-8 conserva el orden de puntos de código Unicode, por lo que también es probable que esta sea la intercalación de mejor rendimiento. Si el orden de puntos de código Unicode funciona para una aplicación, considere utilizar una intercalación BIN2. Sin embargo, el uso de una intercalación BIN2 puede provocar que los datos se muestren en el cliente en un orden inesperado culturalmente. Las nuevas asignaciones a caracteres en minúscula se agregan a Unicode a medida que avanza el tiempo, por lo que la función `LOWER` puede funcionar de manera diferente en distintas versiones de ICU. Este es un caso especial del problema de control de versiones de intercalación más general, en lugar de como algo específico de la intercalación BIN2.

  Babelfish proporciona la intercalación de `BBF_Latin1_General_BIN2` con la distribución de Babelfish para intercalar en el orden de puntos de código Unicode. En una intercalación BIN, solo el primer carácter se ordena como wchar. Los caracteres restantes se ordenan byte a byte, efectivamente en orden de puntos de código según su codificación. Este enfoque no sigue las reglas de intercalación Unicode y Babelfish no lo admite.
- Intercalaciones no deterministas y limitación de CHARINDEX: en las versiones de Babelfish anteriores a la 2.1.0, no puede utilizar CHARINDEX con intercalaciones no deterministas. De forma predeterminada, Babelfish utiliza una intercalación sin distinción de mayúsculas y minúsculas (no determinista). El uso de CHARINDEX para las versiones más antiguas de Babelfish genera el siguiente error de ejecución:

```
nondeterministic collations are not supported for substring searches
```

> **Note**
>
> Esta limitación y esta solución alternativa se aplican solo a la versión 1.x de Babelfish (versiones Aurora PostgreSQL 13.x). Las versiones 2.1.0 y posteriores de Babelfish no tienen este problema.

Puede solucionar este problema de una de las siguientes maneras:

- Convierta explícitamente la expresión en una intercalación distinta entre mayúsculas y minúsculas y doble ambos argumentos aplicando LOWER o UPPER. Por ejemplo, `SELECT charindex('x', a) FROM t1` se convertiría en lo siguiente:

```
SELECT charindex(LOWER('x'), LOWER(a COLLATE sql_latin1_general_cp1_cs_as)) FROM t1
```

- Cree una función SQL f_charindex y reemplace las llamadas de CHARINDEX por llamadas a la siguiente función:

```
CREATE function f_charindex(@s1 varchar(max), @s2 varchar(max)) RETURNS int
AS
BEGIN
declare @i int = 1
WHILE len(@s2) >= len(@s1)
BEGIN
  if LOWER(@s1) = LOWER(substring(@s2,1,len(@s1))) return @i
  set @i += 1
  set @s2 = substring(@s2,2,999999999)
END
return 0
END
go
```

Administración de la gestión de errores de Babelfish con escotillas de escape

Babelfish imita el comportamiento de SQL en el flujo de control y el estado de transacción siempre que sea posible. Cuando Babelfish encuentra un error, devuelve un código de error similar al código de error de SQL Server. Si Babelfish no puede asignar el error a un código de SQL Server, devuelve un código de error fijo (`33557097`) y realiza acciones específicas en función del tipo de error, como se indica a continuación:

- En el caso de los errores de tiempo de compilación, Babelfish revierte la transacción.
- En el caso de los errores de tiempo de ejecución, Babelfish finaliza el lote y revierte la transacción.
- En el caso de los errores de protocolo entre el cliente y el servidor, la transacción no se revierte.

Si un código de error no se puede asignar a un código equivalente y el código de un error similar está disponible, el código de error se asigna al código alternativo. Por ejemplo, los comportamientos que provocan los códigos `8143` y `8144` de SQL Server se asignan a `8143`.

Los errores que no se pueden asignar no respetan una construcción de `TRY... CATCH`.

Puede usar `@@ERROR` para devolver un código de error de SQL Server o la función `@@PGERROR` para devolver un código de error de PostgreSQL. También puede utilizar la función `fn_mapped_system_error_list` para devolver una lista de códigos de error asignados. Para obtener información sobre los códigos de error de PostgreSQL, consulte el sitio web de PostgreSQL.

Modificación de la configuración de escotilla de escape Babelfish

Para gestionar mejor las instrucciones que podrían producir errores, Babelfish define ciertas opciones llamadas “escotillas de escape”. Una escotilla de escape es una opción que especifica el comportamiento de Babelfish cuando encuentra una característica o sintaxis no admitidas.

Puede utilizar el procedimiento almacenado `sp_babelfish_configure` para controlar la configuración de una escotilla de escape. Utilice el script para establecer la escotilla de escape en `ignore` o `strict`. Si está configurado en `strict`, Babelfish devuelve un error que debe corregir antes de continuar.

Incluya la palabra clave `server` para aplicar los cambios a la sesión actual y a nivel de clúster.

El uso es como se indica a continuación:

- Para enumerar todas las escotillas de escape y su estado, además de la información de uso, ejecute `sp_babelfish_configure`.

- Para enumerar las escotillas de escape con nombre y sus valores, en la sesión actual o en todo el clúster, ejecute el comando `sp_babelfish_configure 'hatch_name'` en el que *`hatch_name`* es el identificador de una o más escotillas de escape. *`hatch_name`*puede utilizar comodines de SQL, como '%'.
- Para establecer uno o más escotillas de escape en el valor especificado, ejecute `sp_babelfish_configure ['hatch_name' [, 'strict'|'ignore' [, 'server']]]`. Para que la configuración sea permanente a nivel de todo el clúster, incluya la palabra clave `server`, como se muestra a continuación:

  ```
  EXECUTE sp_babelfish_configure 'escape_hatch_unique_constraint', 'ignore', 'server'
  ```

  Para configurarlos solo para la sesión actual, no utilice `server`.
- Para restablecer todas las escotillas de escape a sus valores predeterminados, ejecute`sp_babelfish_configure 'default'` (Babelfish 1.2.0 y posteriores).

La cadena que identifica la escotilla (o escotillas) podría contener comodines de SQL. Por ejemplo, a continuación, se establecen todas las escotillas de escape de sintaxis en `ignore` para el clúster de Aurora PostgreSQL.

```
EXECUTE sp_babelfish_configure '%', 'ignore', 'server'
```

En la tabla siguiente encontrará descripciones y valores predeterminados para las escotillas de escape predefinidas Babelfish.

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_checkpoint`	Permite el uso de la instrucción CHECKPOINT en el código de procedimiento, pero la instrucción CHECKPOINT no está implementada actualmente.	ignore
`escape_hatch_constraint_name_for_default`	Controla el comportamiento de Babelfish relacionado con	ignore

Escotilla de escape	Descripción	Predeterminado
	los nombres de restricciones predeterminados.	
`escape_hatch_database_misc_options`	Controla el comportamiento de Babelfish relacionado con las siguientes opciones de CREATE DATABASE: CONTAINMENT, DB_CHAINING, TRUSTWORTHY, PERSISTENT_LOG_BUFFER.	ignore
`escape_hatch_for_replication`	Controla el comportamiento de Babelfish relacionado con la cláusula [NOT] FOR REPLICATION cuando se crea o modifica una tabla.	strict
`escape_hatch_fulltext`	Controla el comportamiento de Babelfish relacionado con las características de FULLTEXT, como DEFAULT_FULLTEXT_LANGUAGE en CREATE/ALTER DATABASE, CREATE FULLTEXT INDEX o sp_fulltext_database.	ignore

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_ignore_dup_key`	Controla el comportamiento de Babelfish relacionado con CREATE/ALTER TABLE y CREATE INDEX. Cuando IGNORE_DUP_KEY=ON genera un error cuando se establece en `strict` (el valor predeterminado) o ignora el error cuando se establece en `ignore` (versión de Babelfish 1.2.0 y posteriores).	strict
`escape_hatch_index_clustering`	Controla el comportamiento de Babelfish relacionado con las palabras clave clústerED o NONclústerED para índices y restricciones PRIMARY KEY o UNIQUE. Cuando se ignora clústerED, el índice o la restricción se siguen creando como si se hubiera especificado NONclústerED.	ignore
`escape_hatch_index_columnstore`	Controla el comportamiento de Babelfish relacionado con la cláusula COLUMNSTORE. Si se especifica `ignore`, Babelfish crea un índice de árbol B. regular.	strict

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_join_hints`	Controla el comportamiento de las palabras clave en un operador JOIN: LOOP, HASH, MERGE, REMOTE, REDUCE, REDISTRIBUTE, REPLICATE.	ignore
`escape_hatch_language_non_english`	Controla el comportamiento de Babelfish relacionado con idiomas distintos del inglés para los mensajes en pantalla. Babelfish actualmente solo admite `us_english` para mensajes en pantalla. SET LANGUAGE puede utilizar una variable que contiene el nombre del idioma, por lo que el idioma que se esté configurando solo se puede detectar en tiempo de ejecución.	strict
`escape_hatch_login_hashed_password`	Cuando se ignora, suprime el error de la palabra clave `HASHED` para `CREATE LOGIN` y `ALTER LOGIN`.	strict
`escape_hatch_login_misc_options`	Cuando se ignora, suprime el error de otras palabras clave además de `HASHED`, `MUST_CHANGE` , `OLD_PASSWORD` y `UNLOCK` para `CREATE LOGIN` y `ALTER LOGIN`.	strict

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_login_old_password`	Cuando se ignora, suprime el error de la palabra clave `OLD_PASSWORD` para `CREATE LOGIN` y `ALTER LOGIN`.	strict
`escape_hatch_login_password_must_change`	Cuando se ignora, suprime el error de la palabra clave `MUST_CHANGE` para `CREATE LOGIN` y `ALTER LOGIN`.	strict
`escape_hatch_login_password_unlock`	Cuando se ignora, suprime el error de la palabra clave `UNLOCK` para `CREATE LOGIN` y `ALTER LOGIN`.	strict
`escape_hatch_nocheck_add_constraint`	Controla el comportamiento de Babelfish relacionado con la cláusula WITH CHECK o NOCHECK para las restricciones.	strict
`escape_hatch_nocheck_existing_constraint`	Controla el comportamiento de Babelfish relacionado con las restricciones FOREIGN KEY o CHECK.	strict

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_query_hints`	Controla el comportamiento de Babelfish relacionado con sugerencias de consulta. Cuando esta opción está configurada para ignorar, el servidor ignora las sugerencias que utilizan la cláusula OPTION (...) para especificar los aspectos del procesamiento de consultas . Algunos ejemplos incluyen SELECT FROM ... OPTION(MERGE JOIN HASH, MAXRECURSION 10)).	ignore
`escape_hatch_rowversion`	Controla el comportamiento de los tipos de datos ROWVERSION y TIMESTAMP. Para obtener más información, consulte Uso de las características de Babelfish con implementación limitada.	strict
`escape_hatch_schemabinding_function`	Controla el comportamiento de Babelfish relacionado con la cláusula WITH SCHEMABINDING. De forma predeterminada, la cláusula WITH SCHEMABINDING se ignora cuando se especifica con los comandos CREATE o ALTER FUNCTION.	ignore

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_schemabinding_procedure`	Controla el comportamiento de Babelfish relacionado con la cláusula WITH SCHEMABINDING. De forma predeterminada, la cláusula WITH SCHEMABINDING se ignora cuando se especifica con los comandos CREATE o ALTER PROCEDURE.	ignore
`escape_hatch_rowguidcol_column`	Controla el comportamiento de Babelfish relacionado con la cláusula ROWGUIDCOL cuando se crea o modifica una tabla.	strict
`escape_hatch_schemabinding_trigger`	Controla el comportamiento de Babelfish relacionado con la cláusula WITH SCHEMABINDING. De forma predeterminada, la cláusula WITH SCHEMABINDING se ignora cuando se especifica con los comandos CREATE o ALTER TRIGGER.	ignore
`escape_hatch_schemabinding_view`	Controla el comportamiento de Babelfish relacionado con la cláusula WITH SCHEMABINDING. De forma predeterminada, la cláusula WITH SCHEMABINDING se ignora cuando se especifica con los comandos CREATE o ALTER VIEW.	ignore

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_session_settings`	Controla el comportamiento de Babelfish en relación con las instrucciones SET de nivel de sesión no admitidas.	ignore
`escape_hatch_showplan_all`	Controla el comportamiento de Babelfish relacionado con SET SHOWPLAN_ALL y SET STATISTICS PROFILE. Cuando se configuran en ignore, se comportan como SET BABELFISH_SHOWPLAN_ALL y SET BABELFISH_STATISTICS PROFILE; cuando se configuran en strict, se ignoran silenciosamente.	strict
`escape_hatch_storage_on_partition`	Controla el comportamiento de Babelfish relacionado con la cláusula `ON partition_scheme column` cuando se define la partición. Babelfish actualmente no implementa particiones.	strict

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_storage_options`	Escotilla de escape en cualquier opción de almacenamiento utilizada en CREATE, ALTER DATABASE, TABLE, INDEX. Esto incluye las cláusulas (LOG) ON, TEXTIMAGE_ON, FILESTREAM_ON que definen las ubicaciones de almacenamiento (particiones, grupos de archivos) para tablas, índices y restricciones, así como para una base de datos. Esta configuración de escotilla de escape se aplica a todas estas cláusulas (incluidas ON [PRIMARY] y ON "DEFAULT"). La excepción se produce cuando se especifica una partición para una tabla o índice con ON partition_scheme (columna).	ignore
`escape_hatch_table_hints`	Controla el comportamiento de las sugerencias de tabla especificadas mediante la cláusula WITH (...).	ignore

Escotilla de escape	Descripción	Predeterminado
`escape_hatch_unique_constraint`	Cuando se establece en un nivel estricto, una diferencia semántica desconocida entre SQL Server y PostgreSQL en el manejo de valores NULL en columnas indexadas puede generar errores. La diferencia semántica solo surge en casos de uso poco realistas, por lo que puede configurar esta escotilla de escape para que se «ignore» y evitar ver el error. Obsoleto a partir de las siguientes versiones: 3.6.0 y versiones superiores, 4.2.0 y versiones superiores	strict

Creación de un clúster de base de datos de Babelfish para Aurora PostgreSQL

Babelfish para Aurora PostgreSQL es compatible con la versión 13.4 de Aurora PostgreSQL y versiones posteriores.

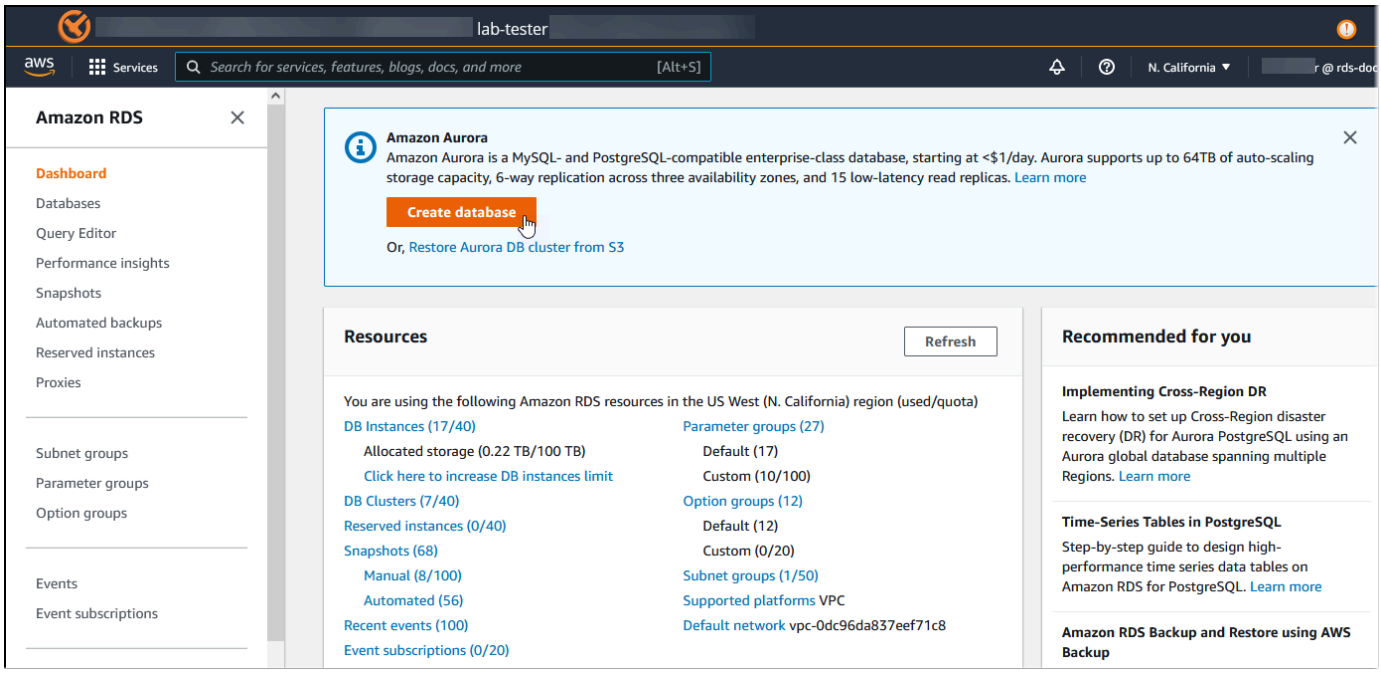
Puede utilizar Consola de administración de AWS o AWS CLI para crear un clúster de Aurora PostgreSQL con Babelfish.

> **Note**
>
> En un clúster de Aurora PostgreSQL, el nombre de la base de datos `babelfish_db` está reservado para Babelfish. La creación de su propia base de datos “babelfish_db” en un Babelfish for Aurora PostgreSQL impide que Aurora aprovisione Babelfish correctamente.

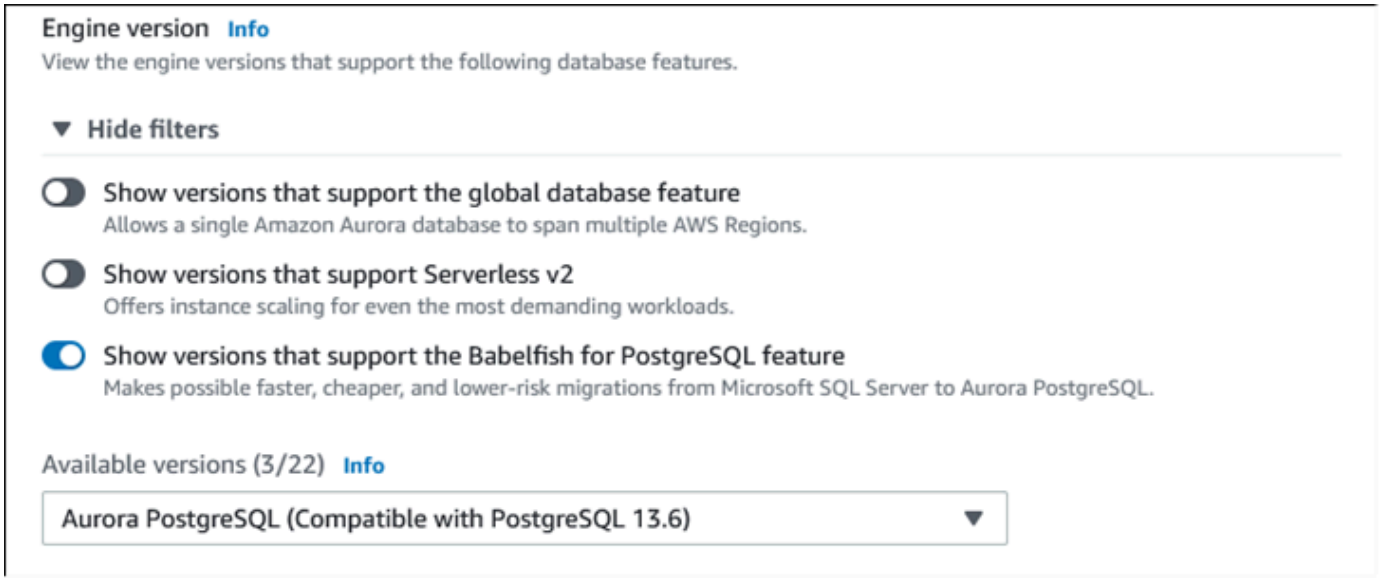
Consola

Para crear un clúster con Babelfish en ejecución con Consola de administración de AWS

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/, y elija Create database (Crear base de datos).



2. Para Choose a database creation method (Elegir un método de creación de base de datos), haga una de las siguientes operaciones:
   - Para especificar opciones de motor detalladas, elija Standard create (Creación estándar).
   - Para utilizar opciones preconfiguradas que admiten prácticas recomendadas para un clúster de Aurora, elija Easy create (Creación sencilla).
3. En Tipo de motor, elija Aurora (compatible con PostgreSQL).
4. Elija Show filters (Mostrar filtros) y, después, Show versions that support the Babelfish for PostgreSQL feature (Mostrar versiones que admiten la característica Babelfish for PostgreSQL) para enumerar los tipos de motores compatibles con Babelfish. Babelfish es compatible actualmente con Aurora PostgreSQL 13.4 y versiones posteriores.
5. En Available versions (Versiones disponibles), elija una versión de Aurora PostgreSQL. Para obtener las últimas características de Babelfish, elija la versión principal de Aurora PostgreSQL más alta.

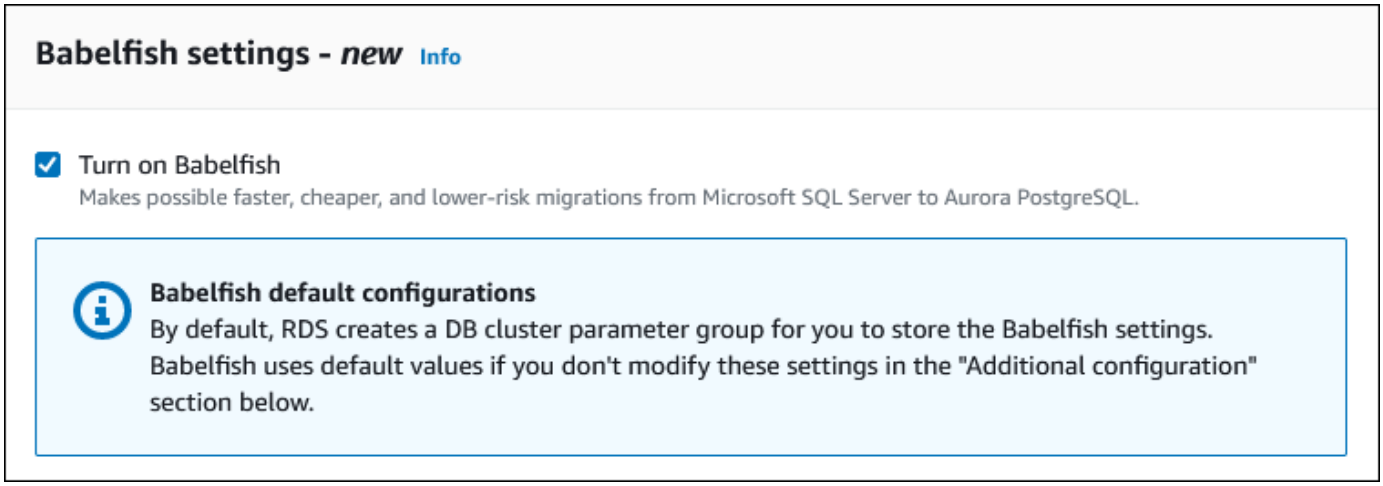


6. En Templates (Plantillas), elija la plantilla que coincida con su caso de uso.
7. En DB cluster identifier (Identificador de clúster de bases de datos), ingrese un nombre que pueda encontrar fácilmente más tarde en la lista de clústeres de base de datos.
8. En Master username (Nombre de usuario maestro), ingrese un nombre de usuario de administrador. El valor predeterminado de Aurora PostgreSQL es `postgres`. Puede aceptar el predeterminado o elegir otro nombre. Por ejemplo, para seguir la convención de nomenclatura utilizada en sus bases de datos de SQL Server, puede ingresar `sa` (administrador del sistema) para el nombre de usuario maestro.

Si no crea un usuario denominado `sa` en ese momento, puede crear uno más tarde con el cliente que elija. Después de crear el usuario, utilice el comando `ALTER SERVER ROLE` para agregarlo al grupo `sysadmin` (rol) del clúster.

> **Warning**
>
> El nombre de usuario maestro siempre debe tener caracteres en minúscula; de lo contrario, el clúster de base de datos no podrá conectarse a Babelfish a través del puerto TDS.

9. En Master password (Contraseña maestra), cree una contraseña sólida y confírmela.
10. En las opciones que siguen, hasta la sección Babelfish settings (Configuración de Babelfish), especifique la configuración del clúster de bases de datos. Para obtener más información acerca de cada configuración, consulte Configuración de clústeres de bases de datos de Aurora.
11. Para que la funcionalidad de Babelfish esté disponible, seleccione la casilla Turn on Babelfish (Activar Babelfish).



12. En DB cluster parameter group (Grupo de parámetros de clúster de bases de datos), haga una de las siguientes operaciones:

    - Elija Create new (Crear nuevo) para crear un nuevo grupo de parámetros con Babelfish activado.
    - Elija Choose existing (Elegir existente) para utilizar un grupo de parámetros existente. Si utiliza un grupo existente, asegúrese de modificar el grupo antes de crear el clúster y agregar valores para los parámetros de Babelfish. Para obtener información acerca de los parámetros de

Babelfish, consulte Configuración del grupo de parámetros del clúster de base de datos para Babelfish.

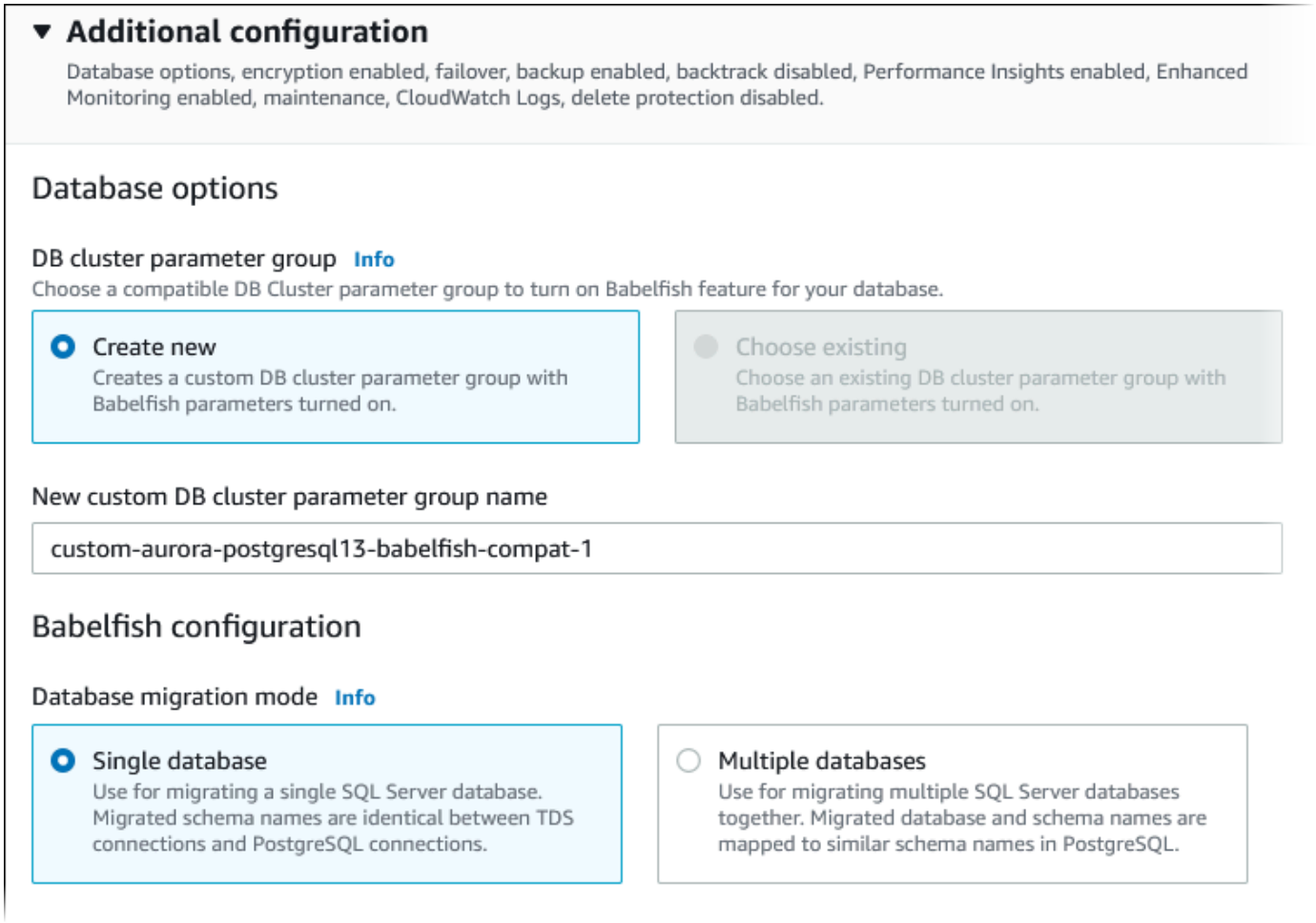
Si utiliza un grupo existente, proporcione el nombre del grupo en el cuadro siguiente.

13. En Database migration mode (Modo de migración de base de datos), elija una de las siguientes opciones:

    - Single database (Base de datos única) para migrar una sola base de datos de SQL Server.

      En algunos casos, puede migrar varias bases de datos de usuarios juntas, lo que tiene como objetivo final una migración completa a Aurora PostgreSQL de forma nativa sin Babelfish. Si las aplicaciones finales requieren esquemas consolidados (un solo esquema dbo), asegúrese de consolidar primero las bases de datos de SQL Server en una sola base de datos de SQL Server. A continuación, migre a Babelfish con el modo Single database (Base de datos única).
    - Multiple databases (Varias bases de datos) para migrar varias bases de datos de SQL Server (originadas en una sola instalación de SQL Server). El modo de varias bases de datos no consolida varias bases de datos que no se originan en una sola instalación de SQL Server. Para obtener información sobre la migración de varias bases de datos, consulte Uso de Babelfish con una base de datos única o varias bases de datos.

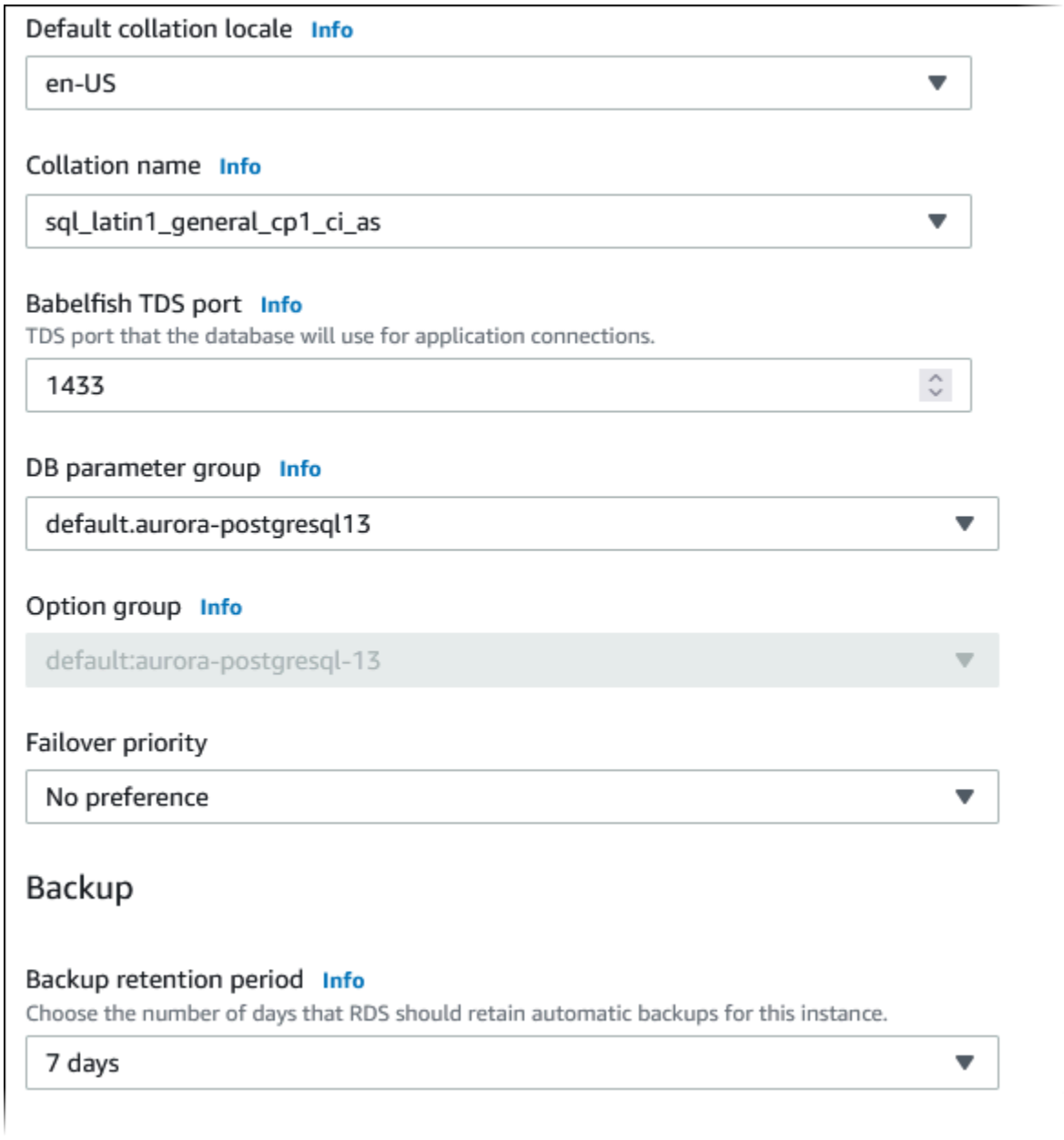
> **Note**
>
> A partir de la versión 16 de Aurora PostgreSQL, se eligen Varias bases de datos de forma predeterminada como modo de migración de bases de datos.



14. En Default collation locale (Configuración regional de intercalación predeterminada), ingrese la configuración regional de su servidor. El valor predeterminado es `en-US`. Para obtener información detallada acerca de las intercalaciones, consulte Comprensión de las intercalaciones de Babelfish para Aurora PostgreSQL.
15. En Collation name (Nombre de la intercalación), ingrese la intercalación predeterminada. El valor predeterminado es `sql_latin1_general_cp1_ci_as`. Para obtener información detallada, consulta Comprensión de las intercalaciones de Babelfish para Aurora PostgreSQL.
16. En Puerto TDS de Babelfish, introduzca el puerto predeterminado `1433`. Actualmente, Babelfish solo admite el puerto `1433` para su clúster de base de datos.
17. En DB parameter group (Grupo de parámetros de base de datos), elija un grupo de parámetros o haga que Aurora cree un grupo nuevo para usted con la configuración predeterminada.
18. En Failover priority (Prioridad de conmutación por error), elija una para la instancia. Si no elige un valor, el valor predeterminado es `tier-1`. Esta prioridad determina el orden en que se

promueven las réplicas de cuando el sistema se recupera de un error en la instancia principal. Para obtener más información, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

19. En Backup retention period (Periodo de retención de copia de seguridad), elija el tiempo (entre 1 y 35 días) durante el que Aurora retendrá las copias de seguridad de la base de datos. Puede utilizar los backups para las restauraciones a un momento dado (PITR) de la base de datos con una precisión de segundos. El periodo de retención predeterminado es de siete de días.



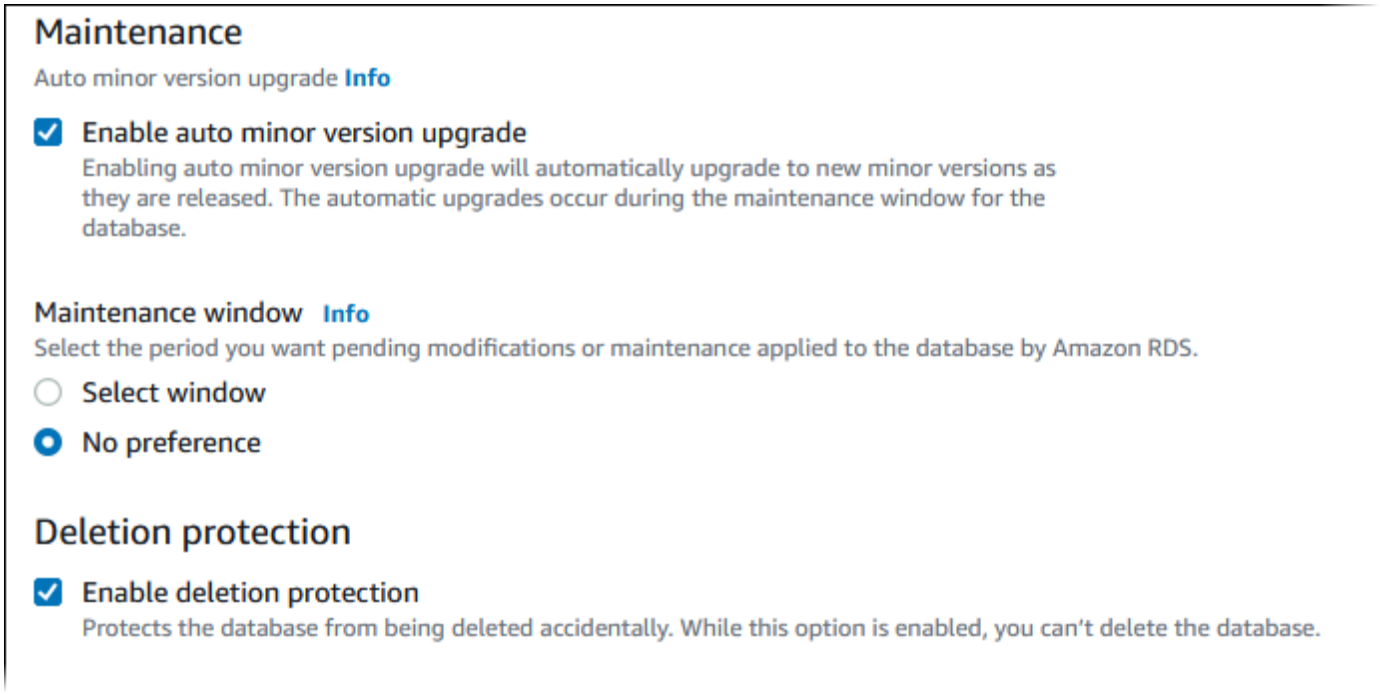
20. Elija Copy tags to snapshot (Copiar etiquetas en instantáneas) para copiar las etiquetas de las instancias de base de datos en una instantánea de base de datos al crear una instantánea.

> **Note**
>
> Al restaurar un clúster de base de datos a partir de una instantánea, no se restaura como un clúster de base de datos de Babelfish para Aurora PostgreSQL. Debe activar los parámetros que controlan las preferencias de Babelfish en el grupo de parámetros del clúster de base de datos para volver a habilitar Babelfish. Para obtener más información sobre estos parámetros, de Babelfish, consulte Configuración del grupo de parámetros del clúster de base de datos para Babelfish.

21. Elija Enable encryption (Habilitar cifrado) para activar el cifrado en reposo (cifrado de almacenamiento de Aurora) para este clúster de bases de datos.
22. Elija Enable Performance Insights (Habilitar Información sobre rendimiento) para activar Amazon RDS Performance Insights.
23. Elija Enable enhanced monitoring (Habilitar monitoreo mejorado) a fin de iniciar la recopilación de métricas en tiempo real para el sistema operativo en el que se ejecuta el clúster de bases de datos.
24. Elija PostgreSQL log (Registro de PostgreSQL) para publicar los archivos de registro en Amazon CloudWatch Logs.
25. Elija Enable auto minor version upgrade (Habilitar actualización automática de versiones secundarias) para actualizar automáticamente el clúster de bases de datos Aurora cuando haya disponible una actualización de versión secundaria.
26. En Maintenance window (Periodo de mantenimiento), haga lo siguiente:

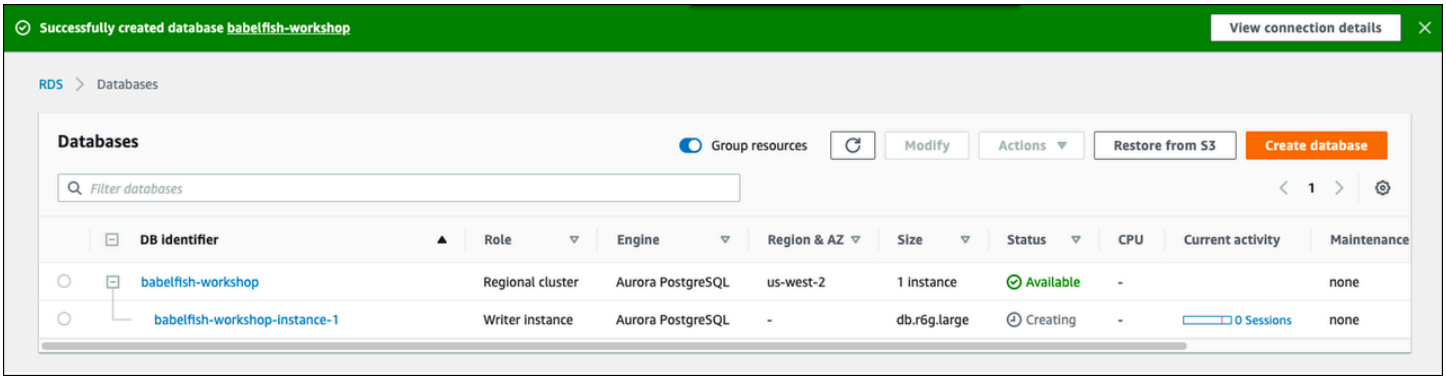
    - Para elegir una hora para que Amazon RDS haga modificaciones o haga el mantenimiento, elija Select window (Seleccionar ventana).
    - Para hacer el mantenimiento de Amazon RDS a una hora no programada, elija No preference (Sin preferencias).
27. Seleccione la casilla Enable deletion protection (Habilitar protección contra la eliminación), para proteger la base de datos de que no se elimine por accidente.

    Si activa esta característica, no podrá eliminar directamente la base de datos. En su lugar, debe modificar el clúster de bases de datos y desactivar esta característica antes de eliminar la base de datos.



28. Elija Creación de base de datos.

Puede encontrar la nueva base de datos configurada para Babelfish en el listado de Databases (Bases de datos). En la columna Status (Estado) se muestra Available (Disponible) cuando la implementación finaliza.



AWS CLI

Cuando cree un Babelfish para Aurora PostgreSQL; mediante la AWS CLI, necesita pasar al comando el nombre del grupo de parámetros de base de datos que se usará para el clúster. Para obtener más información, consulte Requisitos previos de clúster de base de datos.

Antes de poder utilizar AWS CLI para crear un clúster de Aurora PostgreSQL con Babelfish, haga lo siguiente:

- Elija la URL del punto de conexión de la lista de servicios de Amazon Aurora endpoints and quotas (Puntos de conexión y cuotas de Amazon Aurora).
- Cree un grupo de parámetros para el clúster. Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.
- Modifique el grupo de parámetros y agregue el parámetro que activa Babelfish.

Para crear un clúster de bases de datos de Aurora PostgreSQL mediante AWS CLI

Los siguientes ejemplos utilizan el nombre de usuario maestro predeterminado, `postgres`. Reemplácelo según sea necesario por el nombre de usuario que haya creado para su clúster de datos (por ejemplo, `sa`) o por el nombre de usuario que haya elegido si no ha aceptado el predeterminado.

1. Cree un grupo de parámetros.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --db-parameter-group-family aurora-postgresql14 \
   --description "description"
   ```

   Para Windows:

   ```
   aws rds create-db-cluster-parameter-group ^
   --endpoint-url endpoint-URL ^
   --db-cluster-parameter-group-name parameter-group ^
   --db-parameter-group-family aurora-postgresql14 ^
   --description "description"
   ```

2. Modifique el grupo de parámetros para activar Babelfish.

   Para Linux, macOS o Unix:

   ```
   aws rds modify-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   ```

```
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
--endpoint-url endpoint-url ^
--db-cluster-parameter-group-name paramater-group ^
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

3. Identifique el grupo de la subred de base de datos y el ID del grupo de seguridad de la nube virtual privada (VPC) para el nuevo clúster de bases de datos y, a continuación, llame al comando create-db-cluster.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-cluster \
   --db-cluster-identifier  cluster-name\
   --master-username postgres \
   --manage-master-user-password \
   --engine aurora-postgresql \
   --engine-version 14.3            \
   --vpc-security-group-ids security-group \
   --db-subnet-group-name subnet-group-name \
   --db-cluster-parameter-group-name parameter-group
   ```

   Para Windows:

   ```
   aws rds create-db-cluster ^
   --db-cluster-identifier cluster-name ^
   --master-username postgres ^
   --manage-master-user-password ^
   --engine aurora-postgresql ^
   --engine-version 14.3 ^
   --vpc-security-group-ids security-group ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-parameter-group-name parameter-group
   ```

   En este ejemplo se especifica la opción `--manage-master-user-password` para generar la contraseña del usuario maestro y administrarla en Secrets Manager. Para obtener más

información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager. También puede utilizar la opción `--master-password` para especificar y administrar la contraseña usted mismo.

4. Cree explícitamente la instancia principal para el clúster de base de datos. Utilice el nombre del clúster que creó en el paso 3 para el argumento `--db-cluster-identifier` cuando llame al comando create-db-instance, como se muestra a continuación.

   Para Linux, macOS o Unix:

   ```
   aws rds create-db-instance \
   --db-instance-identifier instance-name \
   --db-instance-class db.r6g \
   --db-subnet-group-name subnet-group \
   --db-cluster-identifier cluster-name \
   --engine aurora-postgresql
   ```

   Para Windows:

   ```
   aws rds create-db-instance ^
   --db-instance-identifier instance-name ^
   --db-instance-class db.r6g ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-identifier cluster-name ^
   --engine aurora-postgresql
   ```

Migración de una base de datos SQL Server a Babelfish para Aurora PostgreSQL

Puede utilizar Babelfish para Aurora PostgreSQL a fin de migrar una base de datos de SQL Server a un clúster de base de datos de Amazon Aurora PostgreSQL. Antes de migrar, revise Uso de Babelfish con una base de datos única o varias bases de datos.

Temas

- Información general sobre el proceso de migración
- Evaluación y gestión de diferencias entre SQL Server y Babelfish
- Herramientas de importación y exportación para migrar de SQL Server a Babelfish

Información general sobre el proceso de migración

En el siguiente resumen, se enumeran los pasos necesarios para migrar correctamente su aplicación de SQL Server y hacerla funcionar con Babelfish. Para obtener información sobre las herramientas que puede utilizar para los procesos de exportación e importación y para obtener más detalles, consulte Herramientas de importación y exportación para migrar de SQL Server a Babelfish. Para cargar los datos, le recomendamos que utilice AWS DMS con un clúster de base de datos de Aurora PostgreSQL como punto de conexión de destino.

1. Cree un nuevo clúster de base de datos Aurora PostgreSQL con Babelfish activado. Para aprender a hacerlo, consulte Creación de un clúster de base de datos de Babelfish para Aurora PostgreSQL.

   Para importar los diversos artefactos SQL exportados desde su base de datos SQL Server, conéctese al clúster de Babelfish mediante una herramienta de SQL Server como sqlcmd. Para obtener más información, consulte Uso de un cliente de SQL Server para conectarse al clúster de su base de datos.
2. En la base de datos SQL Server que desea migrar, exporte el lenguaje de definición de datos (DDL). El DDL es un código de SQL que describe los objetos de base de datos que contienen datos de usuario (como tablas, índices y vistas) y código de base de datos escrito por el usuario (como procedimientos almacenados, funciones definidas por el usuario y activadores).

   Para obtener más información, consulte Uso de SQL Server Management Studio (SSMS) para migrar a Babelfish.

3. Ejecute una herramienta de evaluación para valorar el alcance de cualquier cambio que deba realizar para que Babelfish pueda admitir eficazmente la aplicación que se ejecuta en SQL Server. Para obtener más información, consulte Evaluación y gestión de diferencias entre SQL Server y Babelfish.
4. Revise las limitaciones del punto de conexión de destino de AWS DMS y actualice el script DDL según sea necesario. Para obtener más información, consulte Limitations to using a PostgreSQL target endpoint with Babelfish tables en Using Babelfish for Aurora PostgreSQL as a target.
5. En su nuevo clúster de base de datos de Babelfish, ejecute el DDL con la base de datos T-SQL especificada para crear solo los esquemas, los tipos de datos definidos por el usuario y las tablas con sus restricciones de clave principal.
6. Utilice AWS DMS para migrar los datos de SQL Server a las tablas de Babelfish. Para la replicación continua mediante la captura de datos de cambios de SQL Server o la replicación SQL, utilice Aurora PostgreSQL en lugar de Babelfish como punto de conexión. Para ello, consulte Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service (Uso de Babelfish para Aurora PostgreSQL como destino para Database Migration Service).
7. Cuando se complete la carga de datos, cree todos los objetos T-SQL restantes que admita la aplicación en su clúster de Babelfish.
8. Vuelva a configurar la aplicación cliente para conectarse al punto de conexión de Babelfish en lugar de a la base de datos de SQL Server. Para obtener más información, consulte Conexión a un clúster de base de datos de Babelfish.
9. Modifique la aplicación según sea necesario y vuelva a probarla. Para obtener más información, consulte Diferencias entre Babelfish for Aurora PostgreSQL y SQL Server.

Todavía tiene que evaluar sus consultas SQL del cliente. Los esquemas generados a partir de la instancia de SQL Server convierten solo el código SQL en el servidor. Le recomendamos que siga los siguientes pasos:

- Capture las consultas del cliente mediante SQL Server Profiler con la plantilla predefinida TSQL_Replay. Esta plantilla captura la información de las instrucciones T-SQL que se pueden reproducir para el ajuste y las pruebas iterativas. Puede iniciar el generador de perfiles en SQL Server Management Studio, desde el menú Tools (Herramientas). Elija SQL Server Profiler para abrir el generador de perfiles y elija la plantilla TSQL_Replay.

  Para utilizarlo en su migración a Babelfish, inicie un seguimiento y, después, ejecute la aplicación mediante las pruebas funcionales. El generador de perfiles captura las instrucciones T-SQL. Cuando termine la prueba, detenga el seguimiento. Guarde el resultado en un archivo XML con

sus consultas del cliente (File > Save as > Trace XML File for Replay [Archivo > Guardar como > Archivo XML de seguimiento para la reproducción]).

Para obtener más información, consulte SQL Server Profiler en la documentación de Microsoft. Para obtener más información sobre la plantilla TSQL_Replay, consulte SQL Server Profiler Templates (Plantillas de SQL Server Profiler).

- Para las aplicaciones con consultas SQL complejas del cliente, le recomendamos que use Babelfish Compass para determinar la compatibilidad con Babelfish de estas consultas. Si el análisis indica que las instrucciones SQL del cliente contienen características de SQL no compatibles, revise los aspectos de SQL de la aplicación cliente y realice modificaciones según sea necesario.
- También puede capturar las consultas SQL como eventos extendidos (formato .xel). Paar ello, utilice SSMS XEvent Profiler. Tras generar el archivo .xel, extraiga las instrucciones SQL en archivos .xml que Compass pueda procesar. Para obtener más información, consulte Use the SSMS XEvent Profiler (Uso de SSMS XEvent Profiler) en la documentación de Microsoft.

Cuando esté satisfecho con todas las pruebas, análisis y cualquier modificación necesaria de la aplicación migrada, podrá empezar a utilizar la base de datos de Babelfish para la producción. Para ello, detenga la base de datos original y redirija las aplicaciones cliente activas para utilizar el puerto TDS de Babelfish.

> **Note**
>
> AWS DMS ahora admite la replicación de datos de Babelfish. Para obtener más información, consulte AWS DMS now supports Babelfish for Aurora PostgreSQL as a source.

Evaluación y gestión de diferencias entre SQL Server y Babelfish

Para obtener los mejores resultados, le recomendamos que evalúe el DDL/DML generado y el código de consulta del cliente antes de migrar realmente la aplicación de la base de datos SQL Server a Babelfish. Según la versión de Babelfish y las características específicas de SQL Server que implemente la aplicación, es posible que tenga que refactorizarla o utilizar alternativas para las funcionalidades que aún no son totalmente compatibles con Babelfish.

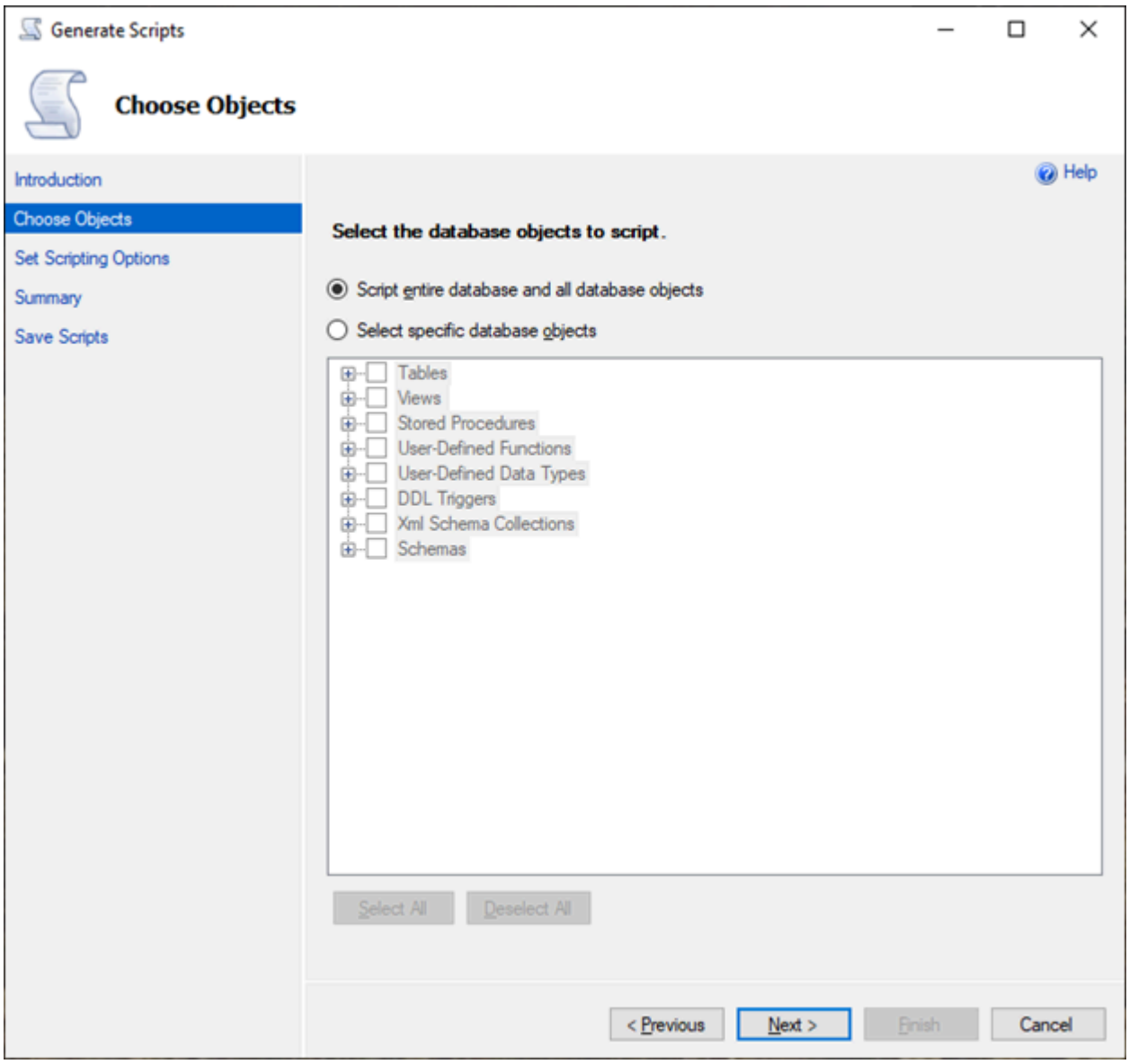
- Para evaluar el código de la aplicación de SQL Server, utilice Babelfish Compass en el DDL generado para determinar hasta qué punto el código T-SQL es compatible con Babelfish.

Identifique el código T-SQL que podría necesitar modificaciones antes de ejecutarse en Babelfish. Para obtener más información sobre esta herramienta, consulte Babelfish Compass en GitHub.

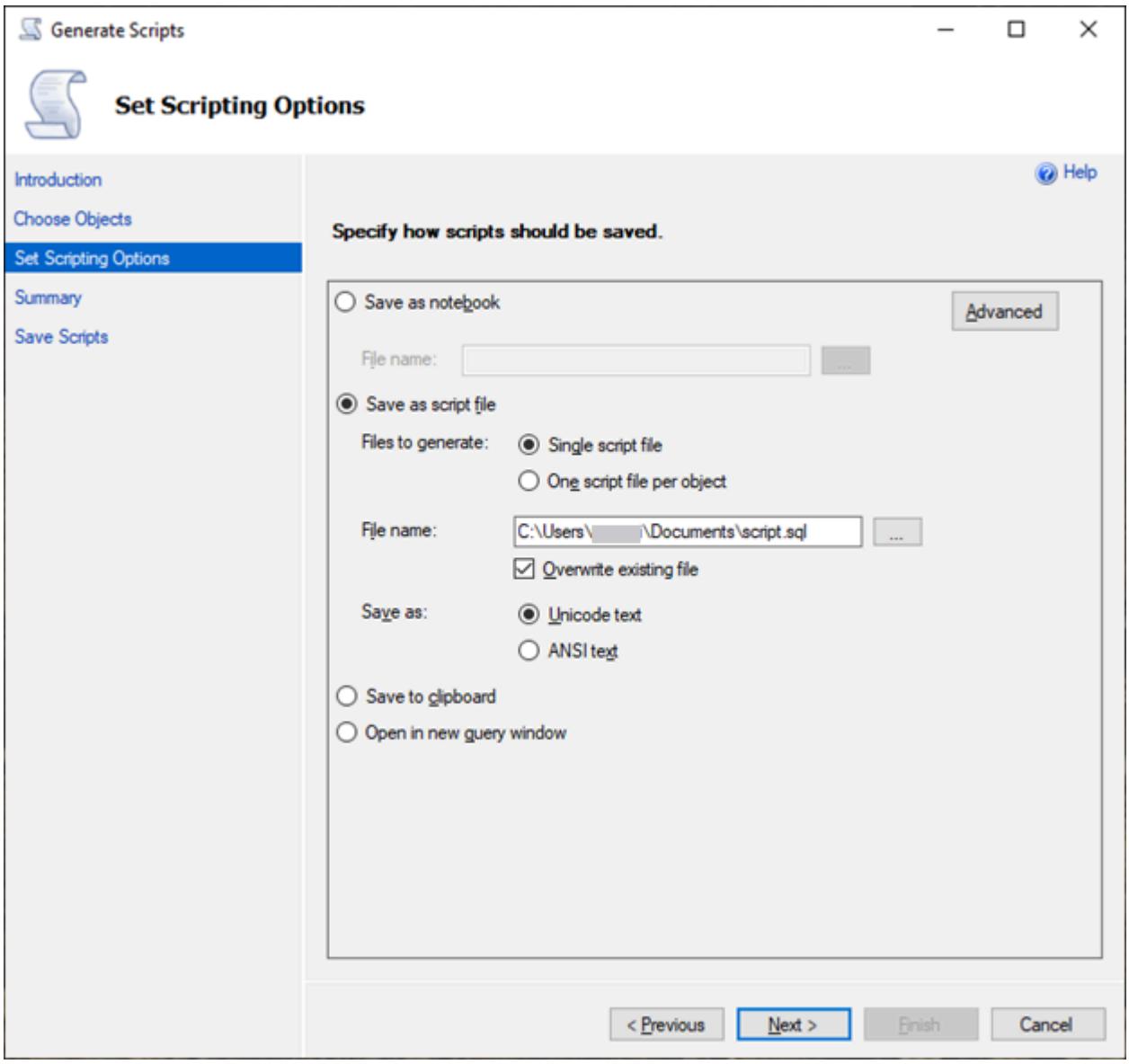
> **Note**
>
> Babelfish Compass es una herramienta de código abierto. Informe de cualquier problema con Babelfish Compass a través de GitHub en lugar de hacerlo a través de AWS Support.

Puede utilizar el asistente de generación de scripts con SQL Server Management Studio (SSMS) para generar el archivo SQL que Babelfish Compass o la CLI AWS Schema Conversion Tool evalúan. Recomendamos los siguientes pasos para agilizar la evaluación.

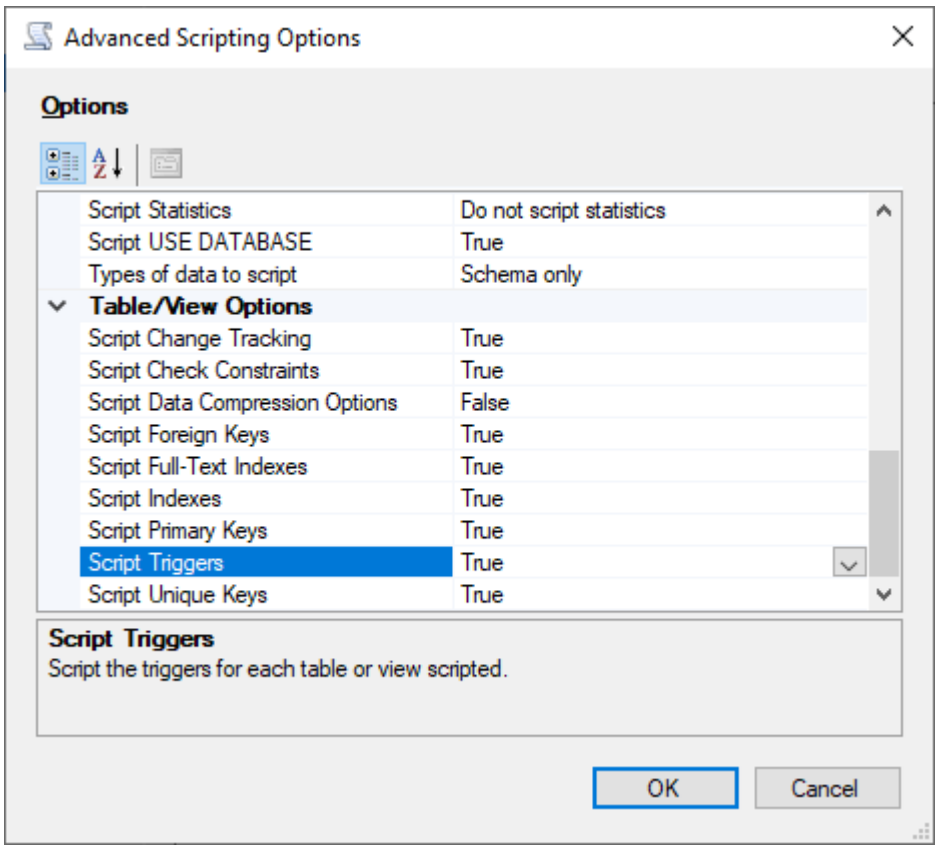
1. En la página Choose Objects (Elegir objetos), elija Script entire database and all database objects (Script de toda la base de datos y todos los objetos de la base de datos).



2. En Set Scripting Options (Definir opciones de scripts), elija Save as script file (Guardar como archivo de script) como Single script file (Archivo de script único).



3. Elija Advanced (Avanzado) para cambiar las opciones de scripts predeterminadas a fin de identificar las características que normalmente se configuran en falsas para una evaluación completa:
   - Script Change Tracking (Rastreo de cambios de script) en True (Verdadero)
   - Script Full-Text Indexes (Índices de texto completo de script) True (Verdadero)
   - Script Triggers (Desencadenadores de scripts) True (Verdadero)
   - Script Logins (Inicios de sesión de scripts) True (Verdadero)
   - Script Owner (Propietario del script) True (Verdadero)
   - Script Object-Level Permissions (Permisos a nivel de objeto del script) True (Verdadero)
   - Script Collations (Intercalaciones de scripts) True (Verdadero)



4. Siga el resto de pasos del asistente para generar el archivo.

Herramientas de importación y exportación para migrar de SQL Server a Babelfish

Le recomendamos que utilice AWS DMS como herramienta principal para migrar de SQL Server a Babelfish. Sin embargo, Babelfish admite otras formas de migrar datos mediante herramientas de SQL Server, entre las que se incluyen las siguientes.

- Servicios de integración de SQL Server (SSIS) para todas las versiones de Babelfish. Para obtener más información, consulte Migrar de SQL Server a Aurora PostgreSQL mediante SSIS y Babelfish.
- Utilice el asistente de importación/exportación de SSMS para las versiones 2.1.0 y posteriores de Babelfish. Esta herramienta está disponible a través de SSMS, pero también está disponible como herramienta independiente. Para obtener más información, consulte Welcome to SQL Server Import and Export Wizard (Asistente de importación y exportación de SQL Server) en la documentación de Microsoft.
- El programa de copia de datos masiva de Microsoft (bcp) le permite copiar datos de una instancia de Microsoft SQL Server a un archivo de datos en el formato que especifique. Para obtener más información, consulte bcp Utility (Utilidad bcp) en la documentación de Microsoft. Babelfish ahora

admite la migración de datos mediante el cliente BCP y la utilidad bcp ahora admite el indicador `-E` (para las columnas de identidad) y el indicador -b (para las inserciones por lotes). Algunas opciones de bcp no son compatibles, como `-C`, `-T`, `-G`, `-K`, `-R`, `-V` y `-h`.

Uso de SQL Server Management Studio (SSMS) para migrar a Babelfish

Se recomienda generar archivos independientes para cada uno de los tipos de objetos específicos. Puede utilizar primero el asistente de generación de scripts de SSMS para cada conjunto de instrucciones DDL y, a continuación, modificar los objetos como un grupo para corregir cualquier problema que se encuentre durante la evaluación.

Siga estos pasos para migrar los datos mediante AWS DMS u otros métodos de migración de datos. Ejecute primero estos tipos de scripts de creación para cargar los datos de las tablas de Babelfish en Aurora PostgreSQL de una manera mejor y más rápida.

1. Ejecute instrucciones `CREATE SCHEMA`.
2. Ejecute instrucciones `CREATE TYPE` para crear tipos de datos definidos por el usuario.
3. Ejecute instrucciones `CREATE TABLE` básicas con las claves principales o restricciones únicas.

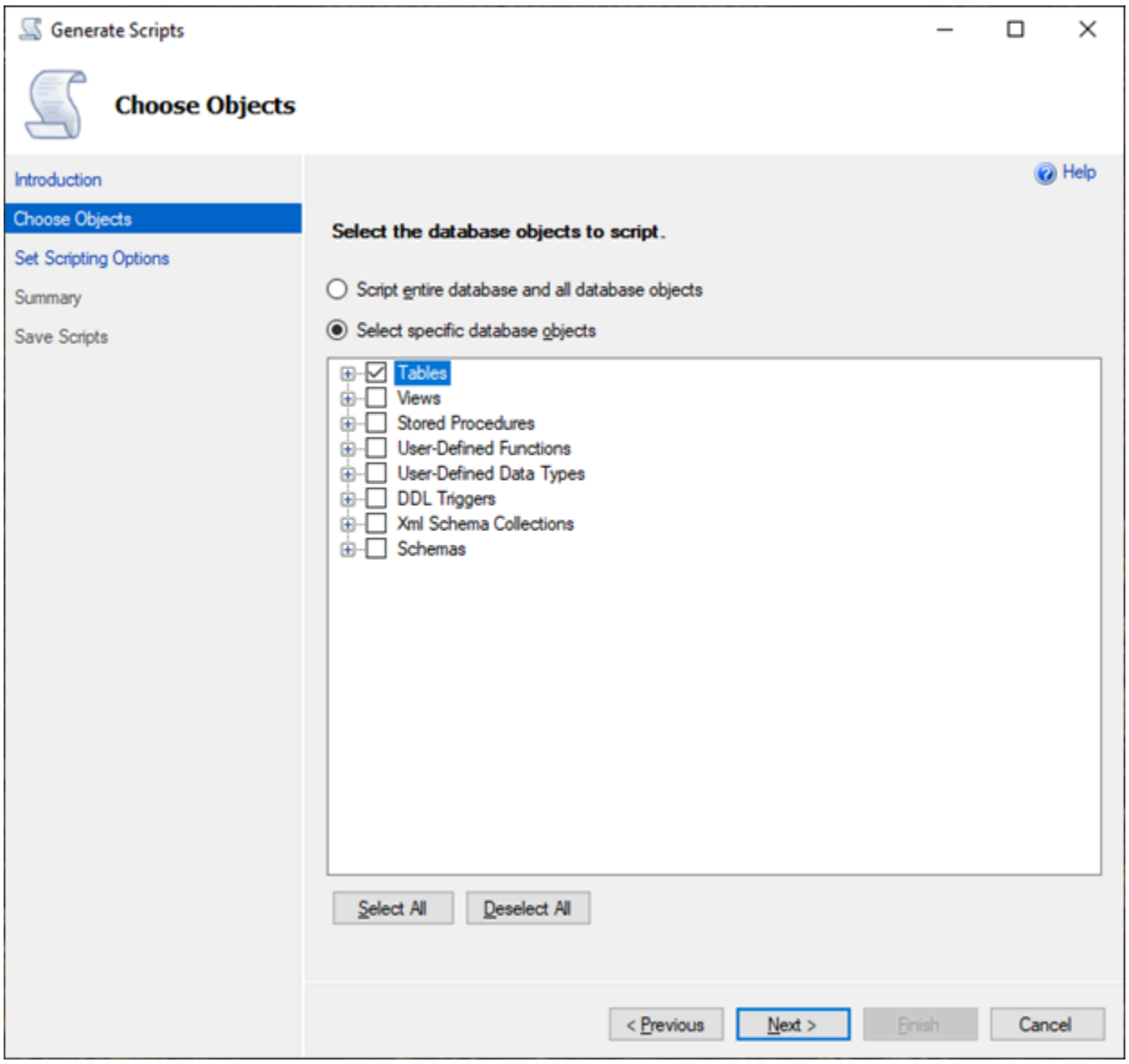
Realice la carga de datos con la herramienta de importación/exportación recomendada. Ejecute los scripts modificados para los siguientes pasos para añadir los objetos de base de datos restantes. Necesita las instrucciones de creación de tablas para ejecutar estos scripts para las restricciones, los desencadenadores y los índices. Una vez generados los scripts, elimine las instrucciones de creación de tablas.

1. Ejecute instrucciones `ALTER TABLE` para las restricciones de verificación, las restricciones de clave externa y las restricciones predeterminadas.
2. Ejecute instrucciones `CREATE TRIGGER`.
3. Ejecute instrucciones `CREATE INDEX`.
4. Ejecute instrucciones `CREATE VIEW`.
5. Ejecute instrucciones `CREATE STORED PROCEDURE`.

Para generar scripts para cada tipo de objeto

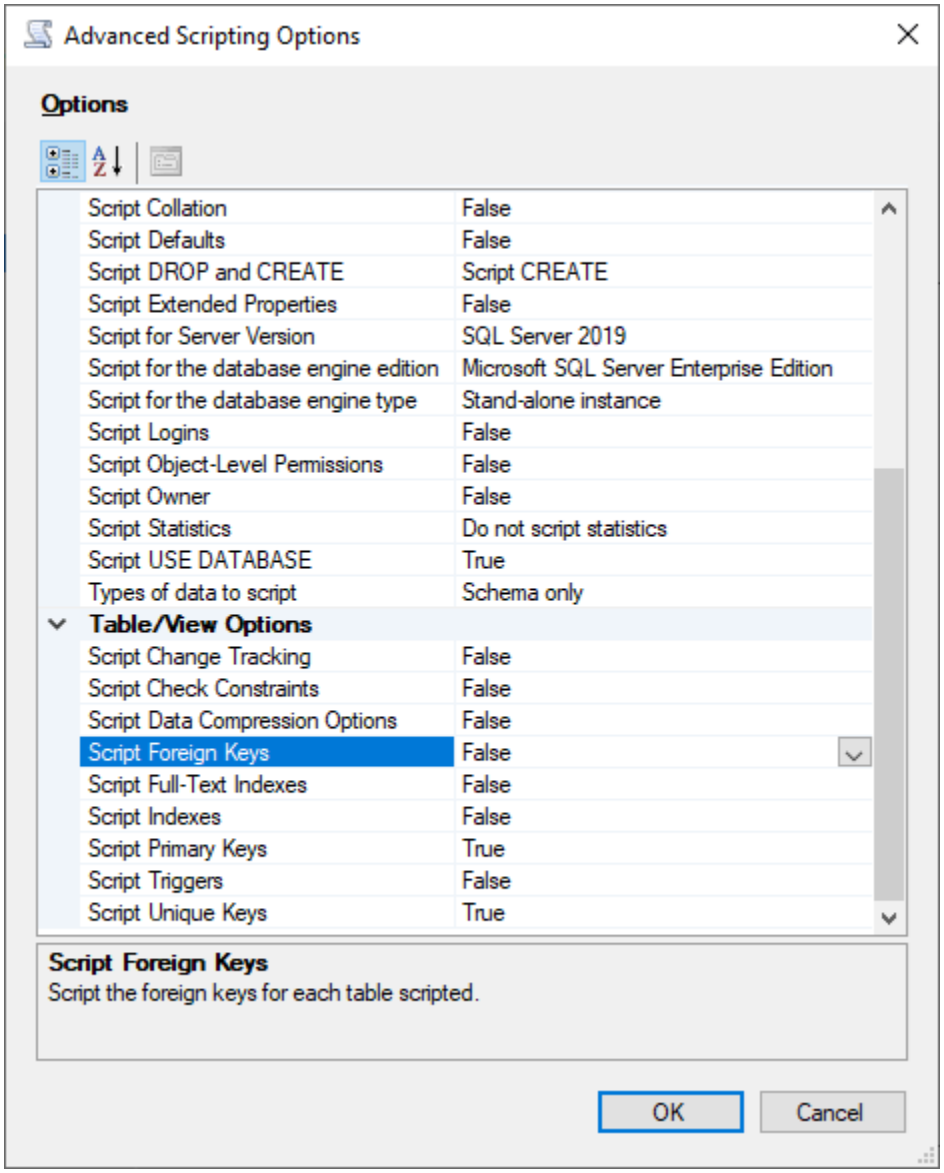
Utilice los siguientes pasos para crear las instrucciones básicas de creación de tablas mediante el asistente para generar scripts en SSMS. Siga los mismos pasos para generar scripts para los diferentes tipos de objetos.

1. Conéctese a la instancia existente de SQL Server.
2. Abra el menú contextual (haga clic con el botón derecho) para obtener un nombre de la base de datos.
3. Elija Tasks (Tareas) y, a continuación, elija Generate Scripts... (Generar scripts).
4. En el panel Choose Objects (Elegir objetos), elija Select specific database objects (Seleccionar objetos específicos de la base de datos). Elija Tables (Tablas), para seleccionar todas las tablas. Elija Siguiente para continuar.

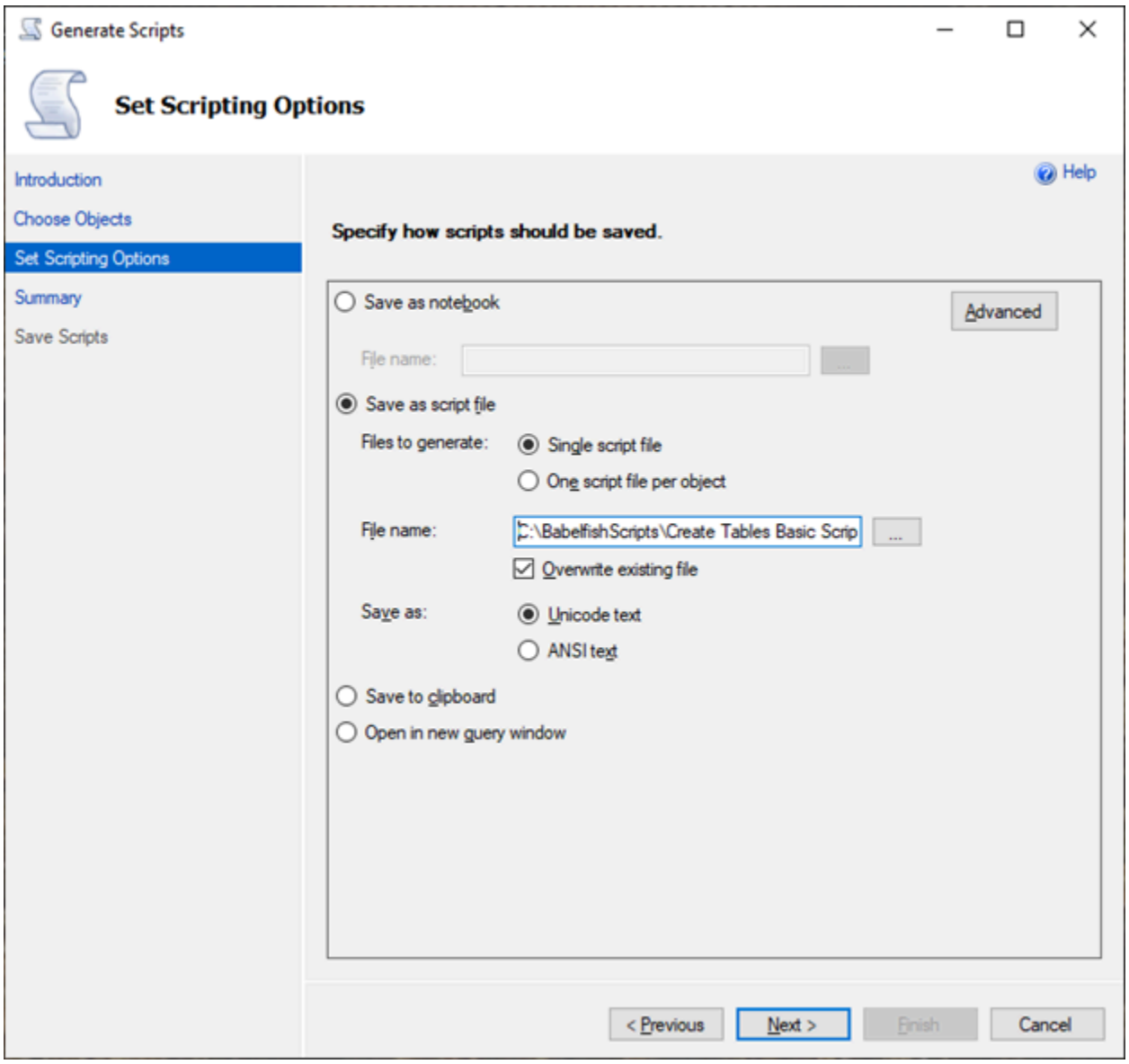


5. En la página Set Scripting Options (Establecer las opciones de scripting), elija Advanced (Avanzado) para abrir la configuración Options (Opciones). Para generar las instrucciones básicas de creación de tablas, cambie los siguientes valores predeterminados:

   - Script Defaults (Valor predeterminado del script) en False (Falso).
   - Script Extended Properties (Propiedades extendidas de script) en False (Falso). Babelfish no admite propiedades extendidas.
   - Script Check Constraints (Restricciones de comprobación de scripts) en False (Falso). Script Foreign Keys (Claves externas de scripts) en False (Falso).



6. Seleccione Aceptar.
7. En Set Scripting Options (Definir opciones de scripts), elija Save as script file (Guardar como archivo de script) y luego elija la opción Single script file (Archivo de script único). Escriba un nombre en File name (Nombre de archivo).



8. Seleccione Next (Siguiente) para ver la página Summary wizard (Asistente de resumen).
9. Elija Next (Siguiente) para iniciar la generación del script.

   Puede seguir generando scripts para los demás tipos de objetos en el asistente. En lugar de elegir Finish (Finalizar) después de guardar el archivo, pulse tres veces el botón Previous (Anterior) para volver a la página Choose Objects (Elegir objetos). A continuación, repita los pasos del asistente para generar scripts para los demás tipos de objetos.

Autenticación de bases de datos con Babelfish para Aurora PostgreSQL

Babelfish para Aurora PostgreSQL admite dos formas de autenticar usuarios de bases de datos. La autenticación de contraseña está disponible de forma predeterminada para todos los clústeres de base de datos de Babelfish. También puede agregar autenticaciones de Kerberos para el mismo clúster de base de datos.

Temas

- Autenticación de contraseña con Babelfish
- Autenticación Kerberos con Babelfish
- Configuración de la autenticación basada en Kerberos mediante los grupos de seguridad de Active Directory para Babelfish

Autenticación de contraseña con Babelfish

Babelfish para Aurora PostgreSQL admite la autenticación de contraseñas. Las contraseñas se almacenan de forma cifrada en el disco. Para obtener más información acerca de la autenticación en un clúster de Aurora PostgreSQL, consulte Seguridad con Amazon Aurora PostgreSQL.

Es posible que se le soliciten credenciales cada vez que se conecte a Babelfish. Cualquier usuario migrado a Aurora PostgreSQL o creado en este puede utilizar las mismas credenciales tanto en el puerto de SQL Server como en el de PostgreSQL. Babelfish no aplica las políticas de contraseñas, pero recomendamos que haga lo siguiente:

- Requiere una contraseña compleja que tenga al menos ocho (8) caracteres.
- Aplique una política de vencimiento de contraseña.

Para revisar una lista completa de usuarios de base de datos, utilice el comando `SELECT * FROM pg_user;`.

Autenticación Kerberos con Babelfish

La versión 15.2 de Babelfish para Aurora PostgreSQL admite la autenticación en su clúster de base de datos mediante Kerberos. Este método le permite utilizar la autenticación de Microsoft Windows para autenticar a los usuarios cuando se conectan a su base de datos Babelfish. Para ello, primero debe configurar el clúster de base de datos para utilizar AWS Directory Service for Microsoft Active Directory para la autenticación Kerberos. Para obtener más información, consulte What is Directory Service? ¿Qué es Directory Service? en la Guía de administración de AWS Directory Service.

Configuración de la autenticación Kerberos

El clúster de base de datos de Babelfish para Aurora PostgreSQL se puede conectar mediante dos puertos diferentes, pero la configuración de la autenticación Kerberos es un proceso que se realiza una sola vez. Por lo tanto, primero debe configurar la autenticación Kerberos para su clúster de base de datos. Para obtener más información, consulte Configuración de autenticación Kerberos. Tras completar la configuración, asegúrese de poder conectarse con un cliente PostgreSQL mediante Kerberos. Para obtener más información, consulte Conexión a PostgreSQL con autenticación Kerberos.

Inicio de sesión y aprovisionamiento de usuarios en Babelfish

Los inicios de sesión de Windows creados desde el puerto Tabular Data Stream (TDS) se pueden usar con el puerto TDS o el puerto PostgreSQL. En primer lugar, el inicio de sesión que puede utilizar Kerberos para la autenticación debe aprovisionarse desde el puerto TDS antes de que los usuarios y las aplicaciones de T-SQL lo utilicen para conectarse a una base de datos Babelfish. Al crear inicios de sesión de Windows, los administradores pueden proporcionar el inicio de sesión mediante el nombre de dominio de DNS o el nombre de dominio de NetBIOS. Normalmente, el dominio de NetBIOS es el subdominio del nombre de dominio de DNS. Por ejemplo, si el nombre de dominio de DNS es `CORP.EXAMPLE.COM`, entonces el dominio de NetBIOS puede ser `CORP`. Si se proporciona el formato de nombre de dominio de NetBIOS para un inicio de sesión, debe existir una asignación al nombre de dominio de DNS.

Administración de la asignación del nombre de dominio de NetBIOS al de DNS

Para administrar las asignaciones entre el nombre de dominio de NetBIOS y el de DNS, Babelfish proporciona procedimientos almacenados del sistema para añadir, eliminar y truncar asignaciones. Solo un usuario con un rol `sysadmin` puede ejecutar estos procedimientos.

Para crear una asignación entre el nombre de dominio de NetBIOS y de DNS, utilice el procedimiento almacenado del sistema `babelfish_add_domain_mapping_entry` proporcionado por Babelfish. Ambos argumentos deben tener un valor válido y no ser NULL.

Example

```
EXEC babelfish_add_domain_mapping_entry 'netbios_domain_name',
 'fully_qualified_domain_name'
```

El siguiente ejemplo muestra cómo crear la asignación entre el nombre CORP de NetBIOS y el nombre de dominio de DNS CORP.EXAMPLE.COM.

Example

```
EXEC babelfish_add_domain_mapping_entry 'corp', 'corp.example.com'
```

Para eliminar una entrada de asignación existente, utilice el procedimiento almacenado del sistema babelfish_remove_domain_mapping_entry.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'netbios_domain_name'
```

En el siguiente ejemplo, se muestra cómo eliminar la asignación del nombre CORP de NetBIOS.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'corp'
```

Para eliminar una entrada de asignación existente, utilice el procedimiento almacenado del sistema babelfish_remove_domain_mapping_entry:

Example

```
EXEC babelfish_truncate_domain_mapping_table
```

Para ver todas las asignaciones entre los nombres de dominio de NetBIOS y DNS, utilice la siguiente consulta.

Example

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

Administración de inicios de sesión

Cree de inicios de sesión

Conéctese a la base de datos a través del punto de conexión TDS utilizando un inicio de sesión que tenga los permisos correctos. Si no se ha creado ningún usuario de base de datos para el inicio de sesión, este se asigna al usuario invitado. Si el usuario invitado no está habilitado, se produce un error en el intento de inicio de sesión.

Cree un inicio de sesión de Windows con la siguiente consulta. La opción `FROM WINDOWS` permite la autenticación mediante Active Directory.

```
CREATE LOGIN login_name FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Example

En el siguiente ejemplo, se muestra la creación de un inicio de sesión para el usuario de Active Directory [corp\test1] con una base de datos predeterminada de db1.

```
CREATE LOGIN [corp\test1] FROM WINDOWS WITH DEFAULT_DATABASE=db1
```

En este ejemplo, se supone que hay una asignación entre el dominio CORP de NetBIOS y el nombre de dominio de DNS CORP.EXAMPLE.COM. Si no hay ninguna asignación, debe proporcionar el nombre del dominio de DNS [CORP.EXAMPLE.COM\test1].

> **Note**
>
> Los inicios de sesión basados en usuarios de Active Directory están limitados a nombres de menos de 21 caracteres.

Borre un inicio de sesión

Para borrar un inicio de sesión, utilice la misma sintaxis que para cualquier inicio de sesión, como se muestra en el siguiente ejemplo:

```
DROP LOGIN [DNS domain name\login]
```

Modifique un inicio de sesión

Para modificar un inicio de sesión, utilice la misma sintaxis que para cualquier inicio de sesión, como en el ejemplo siguiente:

```
ALTER LOGIN [DNS domain name\login] { ENABLE|DISABLE|WITH DEFAULT_DATABASE=[master] }
```

El comando ALTER LOGIN admite opciones limitadas para los inicios de sesión de Windows, entre las que se incluyen las siguientes:

- DISABLE: para deshabilitar un inicio de sesión. No puede usar un inicio de sesión deshabilitado para la autenticación.
- ENABLE: para habilitar un inicio de sesión deshabilitado.
- DEFAULT_DATABASE: para cambiar la base de datos predeterminada de un inicio de sesión.

> **Note**
>
> Toda la administración de contraseñas se realiza mediante Directory Service, por lo que el comando ALTER LOGIN no permite a los administradores de bases de datos cambiar ni establecer contraseñas para los inicios de sesión de Windows.

Conexión a Babelfish para Aurora PostgreSQL con la autenticación Kerberos

Por lo general, los usuarios de bases de datos que se autentican mediante Kerberos lo hacen desde sus máquinas cliente. Estas máquinas son miembros del dominio de Active Directory. Utilizan la autenticación de Windows desde sus aplicaciones cliente para acceder al servidor Babelfish para Aurora PostgreSQL en el puerto TDS.

Conexión a Babelfish para Aurora PostgreSQL en el puerto de PostgreSQL con la autenticación Kerberos

Puede utilizar inicios de sesión de Windows creados desde el puerto TDS o el puerto PostgreSQL. Sin embargo, PostgreSQL usa de forma predeterminada comparaciones que distinguen entre mayúsculas y minúsculas en los nombres de usuario. Para que Aurora PostgreSQL interprete los

nombres de usuario de Kerberos sin distinción entre mayúsculas y minúsculas, debe configurar el parámetro `krb_caseins_users` en `true` en el grupo de parámetros personalizado del clúster de Babelfish. Este parámetro está establecido en `false` de forma predeterminada. Para obtener más información, consulte Configuración del clúster de base de datos de Aurora PostgreSQL para nombres de usuario que no distinguen mayúsculas de minúsculas. Además, debe especificar el nombre de usuario de inicio de sesión en el formato <login@nombre de dominio de DNS> desde las aplicaciones cliente de PostgreSQL. No puede usar el formato <nombre de dominio de DNS\login>.

Errores frecuentes

No puede establecer una relación de confianza de bosque entre Microsoft Active Directory en las instalaciones y AWS Managed Microsoft AD. Para obtener más información, consulte Crear una relación de confianza. A continuación, debe conectarse mediante un punto de conexión específico para un dominio especializado en lugar de utilizar el dominio de Amazon `rds.amazonaws.com` en el punto de conexión del host. Si no utiliza el punto de conexión específico del dominio correcto, es posible que aparezca el siguiente error:

```
Error: “Authentication method "NTLMSSP" not supported (Microsoft SQL Server, Error:
 514)"
```

Este error se produce cuando el cliente TDS no puede almacenar en caché el ticket de servicio de la URL del punto de conexión proporcionada. Para obtener más información, consulte Conexión con Kerberos.

Configuración de la autenticación basada en Kerberos mediante los grupos de seguridad de Active Directory para Babelfish

A partir de la versión 4.2.0 de Babelfish, puede configurar la autenticación Kerberos para Babelfish con grupos de seguridad de Active Directory. Estos son los requisitos previos que debe cumplir para configurar la autenticación Kerberos mediante Active Directory:

- Debe seguir todos los pasos que se mencionan en Autenticación Kerberos con Babelfish.
- Asegúrese de que la instancia de base de datos esté asociada a Active Directory. Para comprobarlo, puede ver el estado de la suscripción al dominio en la consola o ejecutar el comando describe-db-instances de la CLI de AWS.

  El estado de la instancia de base de datos debe estar habilitado para Kerberos. Para obtener más información sobre cómo entender la pertenencia a un dominio, consulte Descripción de la pertenencia a los dominios.

- Compruebe las asignaciones entre el nombre de dominio NetBIOS y el nombre de dominio DNS mediante la siguiente consulta:

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

- Antes de continuar, compruebe que la autenticación Kerberos mediante el inicio de sesión individual funcione según lo previsto. La conexión mediante la autenticación Kerberos como usuario de Active Directory debería realizarse correctamente. Si tiene algún problema, consulte Errores frecuentes.

Configuración de la extensión pg_ad_mapping

Debe seguir todos los pasos que se mencionan en Configuración de la extensión pg_ad_mapping. Para comprobar que la extensión está instalada, ejecute la siguiente consulta desde el punto de conexión de TDS:

```
1> SELECT extname, extversion FROM pg_extension where extname like 'pg_ad_mapping';
2> GO
extname       extversion
------------- ----------
pg_ad_mapping 0.1

(1 rows affected)
```

Administración de inicios de sesión de grupo

Cree inicios de sesión de grupo siguiendo los pasos que se mencionan en Administración de inicios de sesión. Se recomienda que el nombre de inicio de sesión sea el mismo que el nombre del grupo de seguridad de Active Directory (AD) para facilitar el mantenimiento, aunque no es obligatorio. Por ejemplo:

```
CREATE LOGIN [corp\accounts-group] FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Asignación de inicios de sesión de grupos T-SQL con grupos de seguridad de AD

Debe proporcionar explícitamente el inicio de sesión grupal de Windows de T-SQL para cada grupo de seguridad de AD que requiera acceso al servidor de base de datos. Un usuario de AD que forme parte de al menos un grupo de seguridad de AD aprovisionado tendrá acceso al servidor de la base de datos.

> **Note**
>
> Este inicio de sesión de T-SQL ya no se puede autenticar mediante una autenticación basada en contraseñas.

Por ejemplo, accounts-group es un grupo de seguridad de AD y, si quisiese aprovisionar este grupo de seguridad en Babelfish, debería usar el formato [corp\accounts-group].

- Grupo de seguridad de AD: accounts-group
- Inicio de sesión en TSQL: [corp\accounts-group]
- Rol PG equivalente para un inicio de sesión de TSQL determinado: accounts-group@CORP.EXAMPLE.COM

Ahora, el administrador puede crear la asignación entre el grupo de seguridad de AD y el inicio de sesión de T-SQL desde el punto de conexión de PostgreSQL mediante el siguiente comando psql. Para obtener más información sobre el uso de la función, consulte Uso de las funciones de la extensión `pg_ad_mapping`.

> **Note**
>
> El inicio de sesión de T-SQL debe especificarse en el formato login_name@FQDN al añadir la asignación. Al conectarse a través del punto de conexión TDS, las ponderaciones se ignoran. Para obtener más información sobre el uso de ponderaciones, consulte Conexión a Babelfish a través del punto de conexión de PostgreSQL en el puerto PostgreSQL.

```
postgres=>select pgadmap_set_mapping('accounts-group', 'accounts-
group@CORP.EXAMPLE.COM', <SID>, <Weight>);
```

Para obtener información sobre cómo recuperar el SID del grupo de seguridad de AD, consulte Recuperación del SID del grupo de Active Directory en PowerShell.

En la siguiente tabla se muestra un ejemplo de asignación desde grupos de seguridad de AD a inicios de sesión T-SQL:

Grupos de seguridad de AD	Inicios de sesión de TSQL	Rol PG equivalente para un inicio de sesión de TSQL determinado	Peso
accounts-group	[corp\accounts-group]	accounts-group@CORP.EXAMPLE.COM	7
sales-group	[corp\sales-group]	sales-group@CORP.EXAMPLE.COM	10
dev-group	[corp\dev-group]	dev-group@CORP.EXAMPLE.COM	7

```
postgres=> select admap.ad_sid, admap.ad_grp, lgn.orig_loginname, lgn.rolname,
 admap.weight from pgadmap_read_mapping() as admap, sys.babelfish_authid_login_ext as
 lgn where admap.pg_role = lgn.rolname;
    ad_sid    |     ad_grp     |    orig_loginname   |            rolname
 | weight
--------------+----------------+---------------------+--------------------------------
+--------
 S-1-5-67-890 | accounts-group | corp\accounts-group  | accounts-group@CORP.EXAMPLE.COM
 |     7
 S-1-2-34-560 | sales-group    | corp\sales-group     | sales-group@CORP.EXAMPLE.COM
 |     10
 S-1-8-43-612 | dev-group      | corp\dev-group       | dev-group@CORP.EXAMPLE.COM
 |     7
 (7 rows)
```

Conexión a Babelfish a través del punto de conexión de TDS

En el siguiente ejemplo, user1 es miembro de accounts-group y sales-group, mientras que user2 es miembro de accounts-group y dev-group.

Nombre de usuario	Pertenencia a los grupos de seguridad de AD
user1	accounts-group, sales-group
user2	accounts-group, dev-group

Conéctese al servidor de bases de datos Babelfish mediante la utilidad sqlcmd. Puede comprobar si un usuario (user1 en este ejemplo) se ha autenticado mediante Kerberos, como se ve en este ejemplo:

```
1> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();
2>  GO
principal               gss_authenticated
----------------------  -----------------
user1@CORP.EXAMPLE.COM  1

((1 rows affected))
1> select suser_name();
2>  GO
suser_name
----------
corp\user1

(1 rows affected)
```

En este ejemplo, user1 heredará los privilegios de accounts-group y sales-group. Puede verificar la pertenencia al grupo mediante la vista de sistema `sys.login_token`.

```
1> SELECT name, type FROM sys.login_token;
2>  GO
name                type
------------------- ----
corp\accounts-group WINDOWS GROUP
corp\sales-group    WINDOWS GROUP

(2 rows affected)
```

Auditoría y registro

Para determinar la identidad de la entidad principal de seguridad de AD, utilice el siguiente comando:

```
1> select suser_name();
2> GO
suser_name
----------
corp\user1

(1 rows affected)
```

Actualmente, la identidad del usuario de AD no se puede ver en los registros. Puede activar el parámetro `log_connections` para registrar el establecimiento de la sesión de base de datos. Consulte log_connections para obtener más información. La salida incluye la identidad del usuario de AD como entidad principal, como se ve en el siguiente ejemplo. A continuación, el PID del backend asociado a este resultado puede ayudar a atribuir las acciones al usuario real de AD.

```
bbf_group_ad_login@babelfish_db:[615]:LOG: connection authorized:
 user=bbf_group_ad_login database=babelfish_db application_name=sqlcmd GSS
 (authenticated=yes, encrypted=yes, principal=user1@CORP.EXAMPLE.COM)
```

Uso de los privilegios de pertenencia a un grupo de seguridad de AD

Cómo heredar privilegios en el nivel de servidor

Los usuarios de AD que sean miembros de un grupo de seguridad de AD determinado heredarán los privilegios del nivel de servidor otorgados al inicio de sesión del grupo de Windows asignado. Por ejemplo, al grupo de seguridad de AD `accounts-group` se le concede la pertenencia al rol de servidor `sysadmin` en Babelfish. Puede heredar los privilegios de nivel de servidor mediante el siguiente comando:

```
1> ALTER SERVER ROLE sysadmin ADD MEMBER [corp\accounts-group];
```

En consecuencia, cualquier usuario de Active Directory que sea miembro del grupo de seguridad de AD `accounts-group` heredará los privilegios de nivel de servidor asociados al rol `sysadmin`. Esto

significa que un usuario como `corp\user1`, que es miembro de `accounts-group`, podrá realizar operaciones en el nivel de servidor dentro de Babelfish.

> **Note**
>
> Para ejecutar los DDL a nivel de servidor, debe existir el inicio de sesión de Windows para cada usuario de AD. Para obtener más información, consulte Limitaciones.

Cómo heredar privilegios en el nivel de base de datos

Para conceder los privilegios del nivel de base de datos, se debe crear y mapear un usuario de la base de datos con un inicio de sesión grupal de Windows. Los usuarios de AD que sean miembros de un grupo de seguridad de AD determinado heredarán los privilegios de nivel de base de datos otorgados a ese usuario de base de datos. En el siguiente ejemplo, puede ver cómo se asignan los privilegios de nivel de base de datos para el grupo de Windows [corp\accounts-group].

```
1> CREATE DATABASE db1;
2> GO
1> USE db1;
2> GO
Changed database context to 'db1'.
1> CREATE TABLE dbo.t1(a int);
2> GO
```

Cree un usuario de base de datos [corp\sales-group] para el inicio de sesión en grupo de Windows [corp\accounts-group]. Para realizar este paso, conéctese a través del punto de conexión de TDS utilizando el inicio de sesión que sea miembro de sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group];
2> GO
```

Ahora, conéctese como user1 de AD para comprobar el acceso a la tabla t1. Como aún no hemos otorgado los privilegios del nivel de base de datos, se generará un error de permiso denegado.

```
1> SELECT * FROM dbo.t1;
2> GO
Msg 33557097, Level 16, State 1, Server db-inst, Line 1
permission denied for table t1
```

Conceda SELECT en la tabla t1 al usuario de la base de datos [corp\accounts-group]. Para realizar este paso, conéctese a través del punto de conexión de TDS utilizando el inicio de sesión que sea miembro de sysadmin.

```
1> GRANT SELECT ON dbo.t1 TO [corp\accounts-group];
2> GO
```

Conéctese como user1 de AD para validar el acceso.

```
1> SELECT * FROM dbo.t1;
2> GO
a
-----------

(0 rows affected)
```

Gestión del comportamiento de la instrucción DDL en función de un esquema predeterminado o explícito

Cuando se utiliza una sesión autenticada por AD, el esquema predeterminado de la sesión actual viene determinado por las siguientes condiciones:

- Si existe un usuario de base de datos individual, el esquema predeterminado del usuario se considera el esquema predeterminado de la sesión actual.
- Si hay un esquema predeterminado para un usuario de base de datos de grupo, el esquema predeterminado del usuario de la base de datos de grupo se considera el esquema predeterminado de la sesión actual, con el identificador principal más pequeño.

Comprensión del comportamiento de la instrucción CREATE DDL

Si no hay ningún esquema explícito especificado en la instrucción CREATE DDL, la creación del objeto se realizará en el esquema predeterminado de la sesión actual. Si no se puede determinar si el esquema es predeterminado o explícito, la instrucción DDL generará el siguiente error:

```
"Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command."
```

Example : Default schema doesn't exist for Windows group user

El usuario del grupo de Windows [corp\accounts-group] tiene un esquema predeterminado NULL y user1 de AD está intentando ejecutar la DDL sin especificar el esquema de forma explícita. Como no hay un usuario ni un inicio de sesión de Windows individual para user1, solo obtendrá los privilegios del nivel de base de datos del usuario de grupo de Windows [corp\accounts-group].

```
1> create TABLE t2(a int);
2> GO

Msg 33557097, Level 16, State 1, Server db-inst, Line 1
Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command.
```

> **Note**
>
> No hay un usuario ni un inicio de sesión de Windows individual para user1 de AD

Example : Default schema exists for Windows group users

Cree un usuario de grupo de Windows para iniciar sesión en [corp accounts-group] con el esquema predeterminado mediante sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group] WITH
 DEFAULT_SCHEMA = sch_acc;
2> GO
1> CREATE SCHEMA sch_acc AUTHORIZATION [gad\accounts-group];
2> GO
1> SELECT name, principal_id, default_schema_name FROM sys.database_principals WHERE
 name = 'corp\accounts-group';
2> GO

name               principal_id default_schema_name
------------------ ------------ ------------------
corp\accounts-group 24162        sch_acc

(1 rows affected)
```

Intente crear un objeto sin especificar el esquema de forma explícita con user1 de AD. La tabla t2 se creará en el esquema predeterminado del usuario del grupo de Windows [corp\ accounts-group]. El propietario de este objeto será el mismo que el propietario del esquema sch_acc.

```
1> CREATE TABLE t_group(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't_group';
2> GO

name    schema_name
------- -----------
t_group sch_acc

(1 rows affected)
```

> **Note**
>
> No hay un usuario ni un inicio de sesión de Windows individual para user1 de AD

Example : Individual database user also exists for an AD user

Si también existe un usuario de base de datos individual para un usuario de AD, los objetos siempre se crearán en el esquema asociado al usuario de base de datos individual. Si no hay un esquema para el usuario de la base de datos, se utilizará el esquema dbo. Cree un usuario de base de datos y un inicio de sesión de Windows individuales para user1 de AD. Conexión a través del punto de conexión de TDS mediante un inicio de sesión sysadmin

```
1> CREATE LOGIN [corp\user1] FROM WINDOWS;
2> GO
1> CREATE USER [corp\user1] FOR LOGIN [corp\user1] WITH DEFAULT_SCHEMA = sch1;
2> GO
1> CREATE SCHEMA sch1 AUTHORIZATION [corp\user1];
2> GO
1> SELECT name, default_schema_name FROM sys.database_principals WHERE name = 'corp
\user1';
2> GO

name      default_schema_name
--------- -------------------
corp\user1 sch1

(1 rows affected)
```

Conéctese con user1 de AD e intente crear el objeto sin especificar el esquema explícitamente. La tabla t2 se creará en el esquema sch1. Tenga en cuenta que el propietario de este objeto será el mismo que el propietario del esquema sch1.

```
1> CREATE TABLE t2(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't2';
2> GO

name schema_name
---- -----------
t2   sch1

(1 rows affected)
```

Limitaciones

- La utilidad Dump/Restore no admite la descarga de las asignaciones de la extensión pg_ad_mapping. Deberá volver a crear esas asignaciones después de la restauración.
- La implementación azul/verde no es compatible con las instancias de Aurora PostgreSQL y Babelfish con `pg_ad_mapping`.
- No se admite la creación de esquemas implícitos. No se admiten las instrucciones DDL que requieran la creación de esquemas implícitos.
- Los DDL de nivel de servidor ALTER AUTHORIZATION ON DATABASE, CREATE DATABASE, CREATE LOGIN, ALTER LOGIN, ALTER SERVER ROLE y ALTER DATABASE no se admiten en una sesión autenticada de Group AD cuando no existe un inicio de sesión individual de Windows (solo hay inicios de sesión de Windows en grupo). Para evitar esta limitación, es recomendable llevar a cabo estas operaciones en una sesión autenticada por contraseña o crear un inicio de sesión individual en Windows.
- No se admite la creación de usuario implícito. El comportamiento ideal de T-SQL (de momento, no compatible con Babelfish). En algunos casos, como en las instrucciones DDL y de control de acceso (como GRANT/REVOKE) en las que se especifica el nombre del usuario de AD en el comando, pero no existe en la base de datos, se crea implícitamente el usuario de la base de datos, denominado usuario de AD.
- Para los DDL en los procedimientos o funciones de PL/pgSQL que se crean desde el punto de conexión de PSQL y se ejecutan desde el punto de conexión de TDS en una sesión autenticada de Group AD:
  - Se admiten las instrucciones ALTER/DROP.
  - CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE FUNCTION/PROC, CREATE TYPE, CREATE SEQUENCE, CREATE TRIGGER, SELECT INTO, CREATE FULLTEXT INDEX y CREATE UNIQUE INDEX generarán un error si el esquema no se proporciona de forma explícita y el esquema predeterminado es nulo para la sesión actual.
  - CREATE DATABASE, CREATE EXTENSION, todas las demás instrucciones CREATE para objetos específicos de PG (no en T-SQL), CREATE subscription, CREATE tablespace, CREATE policy y CREATE conversion no son compatibles.

- Los DDL del punto de conexión de PostgreSQL no se admiten en la sesión autenticada de Group AD. Como solución alternativa, siempre puede conectarse mediante un usuario maestro o cualquier otro usuario mediante un mecanismo de autenticación basado en contraseñas.
- Los objetos del sistema, como SUSER_SID(), IS_SRVROLEMEMBER(), IS_MEMBER() o sys.dm_exec_sessions, tienen las siguientes limitaciones.
  - SUSER_SID () no devolverá el SID cuando se proporcione el usuario de AD o el grupo de seguridad de AD.
  - IS_SRVROLEMEMBER () no considerará la pertenencia al rol si el usuario actual de AD hereda la pertenencia al rol de servidor de algún miembro del rol de servidor del inicio de sesión en un grupo de Windows.
  - IS_MEMBER () devolverá el valor falso para cualquier consulta relacionada con un grupo de Windows.
  - sys.dm_exec_sessions no mostrará los valores esperados en las columnas login_name y nt_user_name.

Conexión a Babelfish a través del punto de conexión de PostgreSQL en el puerto PostgreSQL

Puede utilizar inicios de sesión de grupo creados desde el puerto TDS para conectarse también a través del puerto PostgreSQL. Para conectarse a través del puerto PostgreSQL, debe especificar el nombre del usuario de AD en el formato `<ad_username@FQDN>` desde las aplicaciones cliente de PostgreSQL. No puede utilizar el formato `<DNS domain name\ad_username>`.

De forma predeterminada, PostgreSQL usa en los nombres de usuario comparaciones que distinguen entre mayúsculas y minúsculas. Para que Aurora PostgreSQL interprete los nombres de usuario de Kerberos sin distinción entre mayúsculas y minúsculas, debe configurar el parámetro krb_caseins_users como verdadero en el grupo de parámetros personalizado del clúster de Babelfish. Este parámetro está establecido en falso de forma predeterminada. Para obtener más información, consulte Configuración del clúster de base de datos de Aurora PostgreSQL para nombres de usuario que no distinguen mayúsculas de minúsculas.

Diferencias de comportamiento entre los puntos de conexión de T-SQL y PostgreSQL cuando un usuario de AD forma parte de varios grupos

Tenga en cuenta que user1 de AD forma parte de dos grupos de seguridad de AD ([corp\accounts-group] y [corp\sales-group]) y que el administrador de base de datos ha configurado la asignación de usuarios de la siguiente manera.

```
postgres=> select * from pgadmap_read_mapping();

ad_sid       | pg_role                          | weight | ad_grp
-------------+----------------------------------+--------+---------------
S-1-5-67-980 | accounts-group@CORP.EXAMPLE.COM | 7      | accounts-group
S-1-2-34-560 | sales-group@CORP.EXAMPLE.COM    | 10     | sales-group
(2 rows)
```

Si el usuario se conecta desde el punto de conexión de T-SQL, heredará durante la autorización los privilegios de todos los inicios de sesión de T-SQL asociados. En este ejemplo, user1 heredará la unión de privilegios del inicio de sesión de ambos grupos T-SQL y se ignorarán las ponderaciones. Esto está en consonancia con el comportamiento estándar de T-SQL.

Sin embargo, si el mismo usuario se conecta desde el punto de conexión de PostgreSQL, solo puede heredar los privilegios de un inicio de sesión de T-SQL asociado, el que tenga la ponderación más alta. Si a los dos inicios de sesión de grupos T-SQL se les asignase la misma ponderación, el usuario de AD heredaría los privilegios del inicio de sesión T-SQL correspondiente a la asignación que se haya efectuado más recientemente. Para PostgreSQL, lo recomendable es especificar ponderaciones que reflejen los permisos y privilegios relativos de los roles de base de datos individuales, a fin de evitar la ambigüedad. En el siguiente ejemplo, user1 se ha conectado a través del punto de conexión de PSQL y ha heredado solo los privilegios de los grupos de ventas.

```
babelfish_db=> select session_user, current_user;

   session_user               |   current_user
------------------------------+--------------------------
 sales-group@CORP.EXAMPLE.COM | sales-group@CORP.EXAMPLE.COM
(1 row)


babelfish_db=> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();

     principal          | gss_authenticated
------------------------+-------------------
 user1@CORP.EXAMPLE.COM | t
(1 row)
```

Conexión a un clúster de base de datos de Babelfish

Para conectarse a Babelfish, conéctese al punto de conexión del clúster de Aurora PostgreSQL que ejecuta Babelfish. El cliente puede utilizar uno de los siguientes controladores de cliente que son conformes con la versión 7.1 a 7.4 de TDS:

- Open Database Connectivity (ODBC)
- OLE DB Driver/MSOLEDBSQL
- Conectividad de base de datos Java (JDBC) versión 8.2.2 (mssql-jdbc-8.2.2) y superior
- Microsoft SqlClient Data Provider for SQL Server
- .NET Data Provider for SQL Server
- SQL Server Native Client 11.0 (obsoleto)
- Proveedor de base de datos OLE/SQLOLEDB (obsoleto)

Con Babelfish, se ejecuta lo siguiente:

- Herramientas, aplicaciones y sintaxis de SQL Server en el puerto TDS, puerto 1433 de forma predeterminada.
- Herramientas, aplicaciones y sintaxis de PostgreSQL en el puerto de PostgreSQL, puerto 5432 de forma predeterminada.

Para obtener más información sobre la conexión a Aurora PostgreSQL en general, consulte Conexión a un clúster de base de datos Amazon Aurora PostgreSQL.

> **Note**
>
> No se admiten herramientas de desarrolladores de terceros que utilizan el proveedor OLEDB de SQL Server para acceder a metadatos. Se recomienda utilizar conexiones de cliente JDBC de SQL Server, ODBC o SQL Native para estas herramientas.

A partir de la versión 5.1.0 de Babelfish, se aplica de forma predeterminada el cifrado de conexión de extremo a extremo. Para garantizar una conectividad continua:

- Configure el cifrado SSL/TLS para las conexiones. Para obtener más información, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

- Importe los certificados necesarios en los equipos cliente. Para obtener más información, consulte Uso de SSL con una instancia de base de datos de Microsoft SQL Server.

Si desea seguir utilizando la configuración de cifrado de una versión anterior de Babelfish (anterior a la versión 5.1.0), puede establecer el parámetro `rds.force_ssl` en `0` en el grupo de parámetros del clúster de base de datos.

Temas

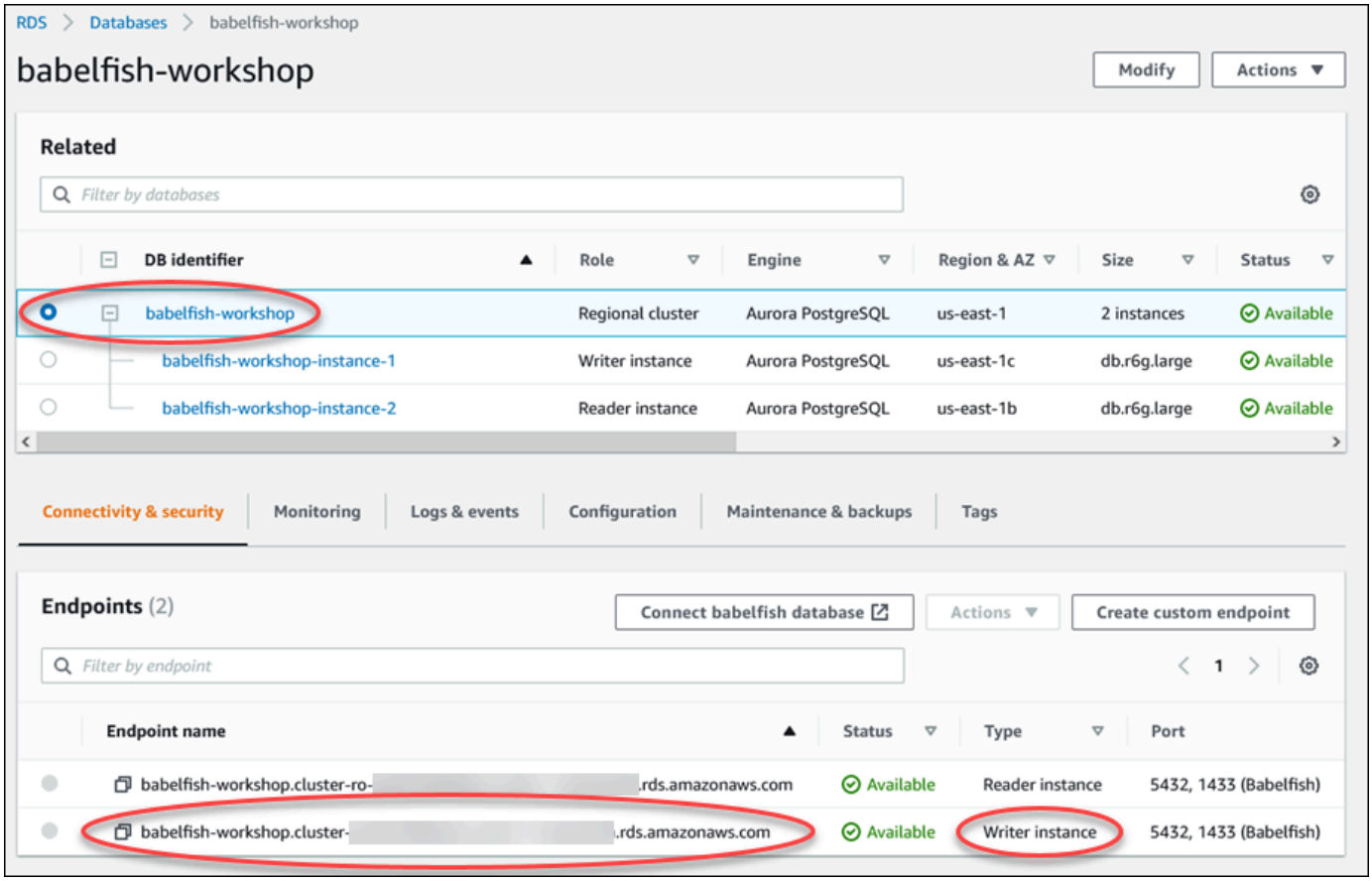
- Búsqueda del punto de conexión y del número de puerto del escritor
- Creación de conexiones de cliente en C# o JDBC a Babelfish
- Uso de un cliente de SQL Server para conectarse al clúster de su base de datos
- Uso de un cliente de PostgreSQL para conectarse al clúster de su base de datos

Búsqueda del punto de conexión y del número de puerto del escritor

Para conectarse a su clúster de base de datos de Babelfish, utilice el punto de conexión asociado a la instancia del escritor (principal) del clúster de base de datos. La instancia debe tener el estado Available (Disponible). Las instancias pueden tardar hasta 20 minutos en estar disponibles después de crear el clúster de base de datos de Babelfish para Aurora PostgreSQL.

Para encontrar el punto de conexión de la base de datos

1. Abra la consola de Babelfish.
2. Elija Databases (Bases de datos) en el panel de navegación.
3. Elija el clúster de base de datos de Babelfish para Aurora PostgreSQL de entre los enumerados para ver sus detalles.
4. En la pestaña Connectivity & security (Conectividad y seguridad), anote los valores de Endpoints (Puntos de conexión) del clúster. Utilice el punto de conexión del clúster para la instancia del escritor en las cadenas de conexión en cualquier aplicación que haga operaciones de lectura o escritura de base de datos.



Para obtener más información sobre los detalles de un clúster de base de datos de Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora.

Creación de conexiones de cliente en C# o JDBC a Babelfish

A continuación, puede encontrar algunos ejemplos de uso de clases C# y JDBC para conectarse a Babelfish para Aurora PostgreSQL.

Example : uso de código C# para conectarse a un clúster de base de datos

```
string dataSource = 'babelfishServer_11_24';

//Create connection
connectionString = @"Data Source=" + dataSource
    +";Initial Catalog=your-DB-name"
    +";User ID=user-id;Password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionString = connectionString + ";ServerCertificate=/path/to/certificate.pem";


SqlConnection cnn = new SqlConnection(connectionString);
cnn.Open();
```

Example : uso de clases e interfaces de API de JDBC genéricas para conectarse a un clúster de base de datos

```
String dbServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";
String connectionUrl = "jdbc:sqlserver://" + dbServer + ":1433;" +
    "databaseName=your-DB-name;user=user-id;password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionUrl = connectionUrl + ";serverCertificate=/path/to/certificate.pem";

// Load the SQL Server JDBC driver and establish the connection.
System.out.print("Connecting Babelfish Server ... ");
Connection cnn = DriverManager.getConnection(connectionUrl);
```

Example : uso de clases e interfaces de JDBC específicas de SQL Server para conectarse a un clúster de base de datos

```
// Create datasource.
SQLServerDataSource ds = new SQLServerDataSource();
ds.setUser("user-id");
```

```
ds.setPassword("password");
String babelfishServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";

ds.setServerName(babelfishServer);
ds.setPortNumber(1433);
ds.setDatabaseName("your-DB-name");

// [optional] To validate server certificate during TLS/SSL connection
ds.setServerCertificate("/path/to/certificate.pem");

Connection con = ds.getConnection();
```

Important

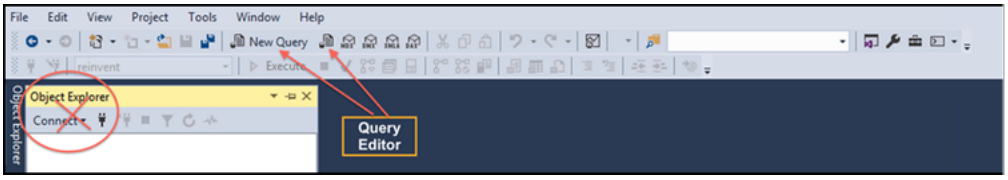
Asegúrese de que el certificado coincide con la autoridad de certificación que se muestra en la configuración del clúster de base de datos en la Consola de administración de AWS.

Uso de un cliente de SQL Server para conectarse al clúster de su base de datos

Puede utilizar un cliente de SQL Server para conectarse con Babelfish en el puerto TDS. A partir de la versión 2.1.0 de Babelfish, puede utilizar el Explorador de objetos de SSMS o el Editor de consultas de SSMS para conectarse al clúster de Babelfish.

Limitaciones

- En Babelfish 2.1.0 y versiones anteriores, el uso de `PARSE` para comprobar la sintaxis de SQL no funciona como debería. En lugar de comprobar la sintaxis sin ejecutar la consulta, el comando `PARSE` ejecuta la consulta pero no muestra ningún resultado. El uso de la combinación de teclas <Ctrl><F5> de SMSS para comprobar la sintaxis tiene el mismo comportamiento anómalo, es decir, Babelfish ejecuta inesperadamente la consulta sin proporcionar ninguna salida.
- Babelfish no es compatible con conjuntos de resultados activos múltiples (MARS). Asegúrese de que cualquier aplicación cliente que utilice para conectarse a Babelfish no esté configurada para usar MARS.
- En el caso de Babelfish 1.3.0 y versiones anteriores, solo se admite el Editor de consultas para SSMS. Para utilizar SSMS con Babelfish, asegúrese de abrir el cuadro de diálogo de conexión del Editor de consultas en SSMS, y no el Explorador de objetos. Si se abre el cuadro de diálogo del Explorador de objetos, cierre el cuadro de diálogo y vuelva a abrir el Editor de consultas. En la siguiente imagen, puede encontrar las opciones de menú que debe elegir cuando se conecte a Babelfish 1.3.0 o a versiones anteriores.



Para obtener más información sobre la interoperabilidad y las diferencias de comportamiento entre SQL Server y Babelfish, consulte Diferencias entre Babelfish for Aurora PostgreSQL y SQL Server.

Uso de sqlcmd para conectarse al clúster de bases de datos

Puede conectarse a un clúster de bases de datos de Aurora PostgreSQL que admita Babelfish, además de interactuar con este, solamente mediante el cliente de línea de comandos `sqlcmd` de SQL Server versión 19.1 y anteriores. La versión 19.2 de SSMS no es compatible con la conexión a un clúster de Babelfish. Para conectarse utilice el siguiente comando.

```
sqlcmd -S endpoint,port -U login-id -P password -d your-DB-name
```

Las opciones son las siguientes:

- `-S` es el punto de conexión y el puerto TDS (opcional) del clúster de bases de datos.
- `-U` es el nombre de inicio de sesión del usuario.
- `-P` es la contraseña asociada al usuario.
- `-d` es el nombre de la base de datos de Babelfish.

Después de conectarse, puede utilizar muchos de los mismos comandos que utiliza con SQL Server. Para ver algunos ejemplos, consulte Obtención de información del catálogo del sistema de Babelfish.

Uso de SSMS para conectarse al clúster de bases de datos

Puede conectarse a un clúster de base de datos de Aurora PostgreSQL que ejecuta Babelfish por medio de Microsoft SQL Server Management Studio (SSMS). SSMS incluye varias herramientas, como el Asistente para importación y exportación de SQL Server que se describe en Migración de una base de datos SQL Server a Babelfish para Aurora PostgreSQL. Para obtener más información sobre SSMS, consulte Download SQL Server Management Studio (SSMS) (Descargar SQL Server Management Studio [SSMS]) en la documentación de Microsoft. Para configurar SSL/TLS, consulte Uso de SSL con una instancia de base de datos de Microsoft SQL Server.

Para conectarse a su base de datos de Babelfish con SSMS

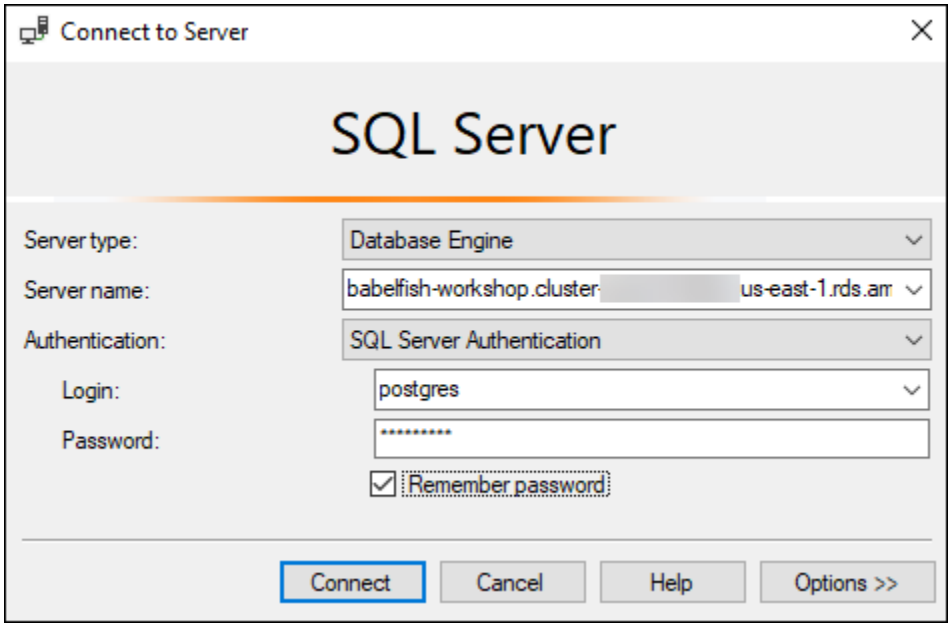
1. Inicie SSMS.
2. Abra el cuadro de diálogo Connect to Server (Conectarse al servidor). Para continuar con la conexión, realice una de las siguientes acciones:
   - Elija New query (Nueva consulta).
   - Si el Editor de consultas está abierto, elija Query (Consulta), Connection (Conexión), Connect (Conectarse).
3. Proporcione la siguiente información para la base de datos:
   a. En Server type (Tipo de servidor), elija Database Engine (Motor de base de datos).
   b. En Server name (Nombre de servidor), ingrese el nombre de DNS. Por ejemplo, el nombre del servidor debería tener un aspecto similar al siguiente.

      ```
      cluster-name.cluster-555555555555.aws-region.rds.amazonaws.com,1433
      ```

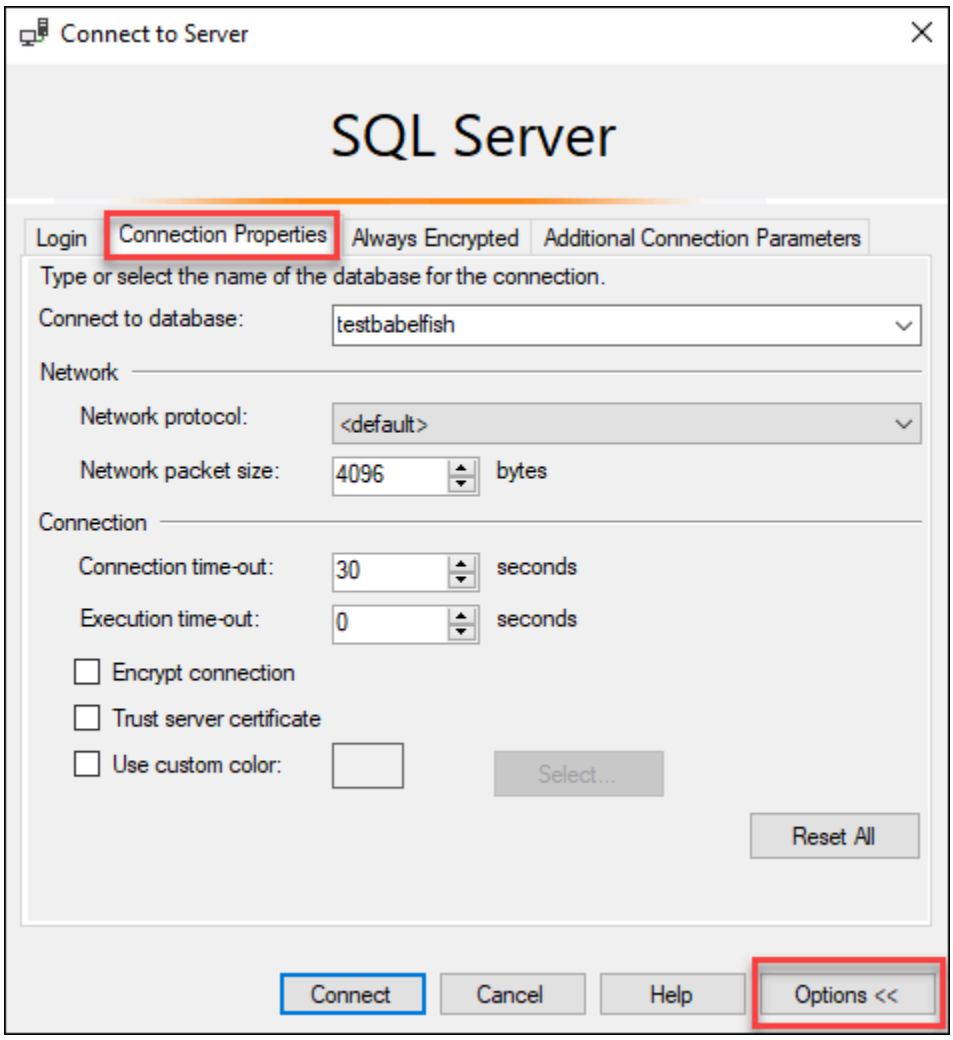
   c. En Authentication, elija SQL Server Authentication.

d. En Login (Inicio de sesión), ingrese el nombre de usuario que eligió al crear la base de datos.

e. En Password (Contraseña), ingrese la contraseña que eligió al crear la base de datos.



4. (Opcional) Elija Options (Opciones) y, después, elija la pestaña Connection Properties (Propiedades de conexión).



5. (Opcional) En Connect to database (Conectarse a la base de datos), especifique el nombre de la base de datos de SQL Server migrada a la que quiera conectarse y elija Connect (Conectar).

   Si aparece un mensaje que indique que SSMS no puede aplicar cadenas de conexión, elija OK (Aceptar).

   Si tiene problemas para conectarse a Babelfish, consulte Error de conexión.

   Para obtener más información sobre problemas de conexión con SQL Server, consulte Solución de problemas de conexión a la instancia de base de datos de SQL Server en la guía del usuario de Amazon RDS.

Uso de un cliente de PostgreSQL para conectarse al clúster de su base de datos

Puede utilizar un cliente de PostgreSQL para conectarse a Babelfish en el puerto de PostgreSQL. A partir de la versión 5.1.0, el servidor de Babelfish aplica el cifrado de conexión de extremo a extremo de forma predeterminada. Actualice la aplicación para trabajar con certificados SSL/TLS. Para obtener más información acerca de la configuración de certificados SSL/TLS, consulte the section called “Protección de los datos de Aurora PostgreSQL con SSL/TLS”.

Conexión al clúster de bases de datos mediante psql

Puede descargar el cliente e PostgreSQL desde el sitio web de PostgreSQL. Para instalar psql, siga las instrucciones específicas de su sistema operativo.

Puede consultar un clúster de bases de datos de Aurora PostgreSQL que admita Babelfish con el cliente de línea de comandos `psql`. Al conectarse, utilice el puerto de PostgreSQL (de forma predeterminada es el puerto 5432). Normalmente, no necesita especificar el número de puerto a menos que haya cambiado el predeterminado. Utilice el siguiente comando para conectarse a Babelfish desde el cliente `psql`:

```
psql -h bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com
-p 5432 -U postgres -d babelfish_db
```

Los parámetros son los siguientes:

- `-h`: el nombre de host del clúster (punto de conexión del clúster) de base de datos al que quiere acceder.
- `-p`: el número de puerto de PostgreSQL usado para conectarse a la instancia de base de datos.
- `-d`: la base de datos a la que desea conectarse. El valor predeterminado es `babelfish_db`.
- `-U`: la cuenta de usuario de base de datos a la que quiere acceder. (En el ejemplo se muestra el nombre de usuario maestro predeterminado).

Cuando ejecute un comando de SQL en el cliente psql, finalícelo con un punto y coma. Por ejemplo, el siguiente comando SQL consulta la vista de sistema pg_tables para devolver información sobre cada tabla de la base de datos.

```
SELECT * FROM pg_tables;
```

El cliente psql también tiene un conjunto de metacomandos integrados. Un metacomando es un atajo que ajusta el formato o proporciona un acceso directo que devuelve metadatos en un formato fácil

de usar. Por ejemplo, el siguiente metacomando devuelve información similar al comando de SQL anterior:

```
\d
```

No es necesario que los metacomandos terminen con punto y coma (;).

Para salir del cliente psql, ingrese \q.

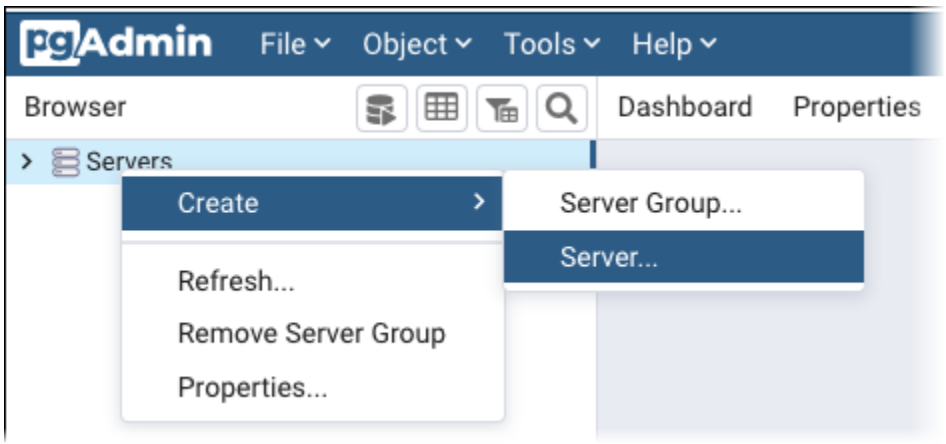
Para obtener más información acerca del uso del cliente psql para consultar un clúster de Aurora PostgreSQL, consulte la documentación de PostgreSQL.

Uso de pgAdmin para conectarse al clúster de bases de datos

Puede utilizar el cliente pgAdmin para acceder a sus datos en dialecto de PostgreSQL nativo.

Para conectarse al clúster con el cliente pgAdmin

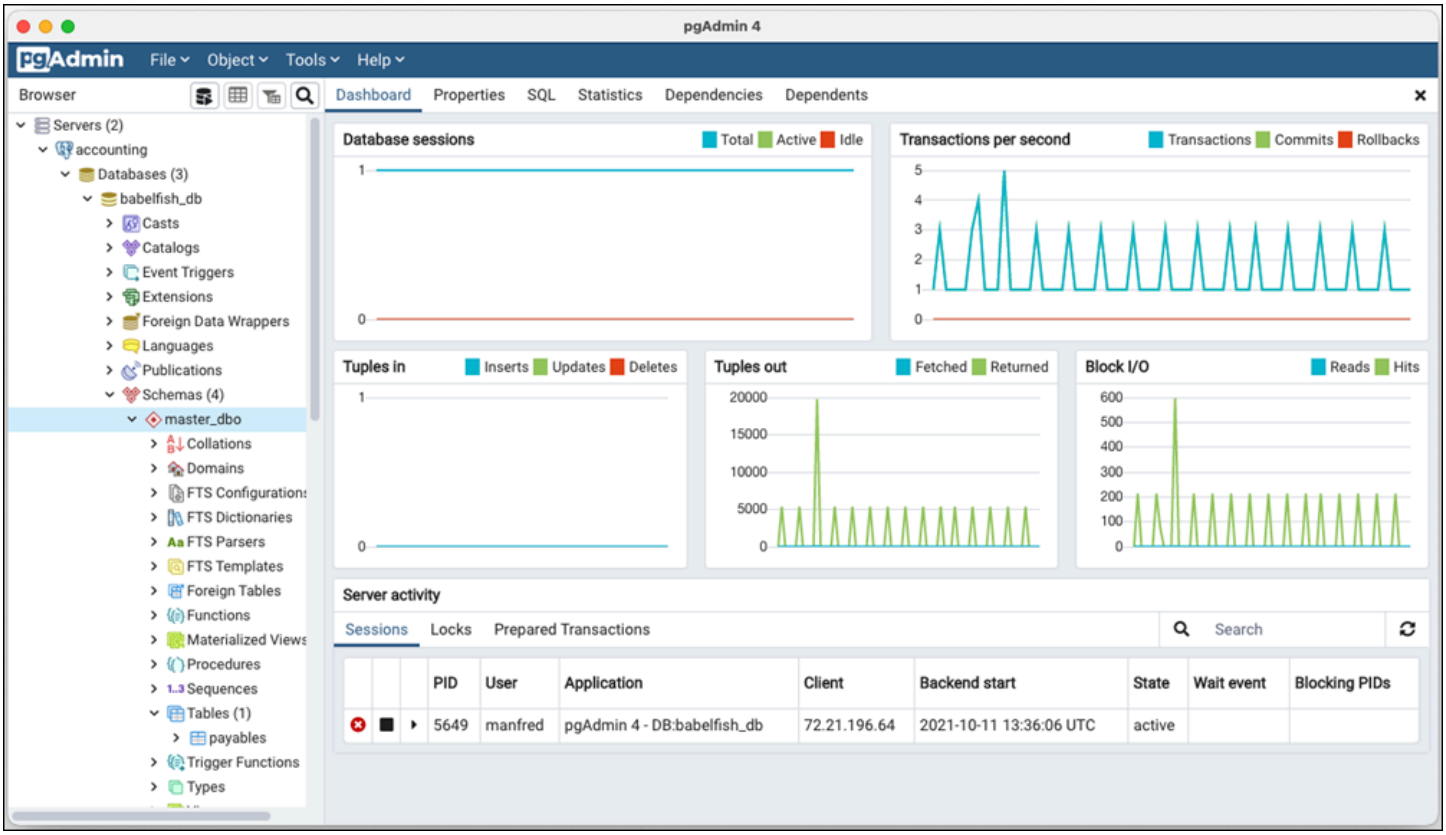
1. Descargue e instale el cliente pgAdmin del Sitio web de pgAdmin.
2. Abra el cliente y autentíquese con pgAdmin.
3. Abra el menú contextual (haga clic con el botón derecho) de Servers (Servidores) y, después, elija Create (Crear), Server (Servidor).



4. Ingrese la información en el cuadro de diálogo Create - Server (Crear - Servidor).

   En la página Connection (Conexión), agregue la dirección del clúster de Aurora PostgreSQL para Host y el número de puerto de PostgreSQL (de forma predeterminada, 5432) para Port (Puerto). Proporcione detalles de autenticación y elija Save (Guardar).

Después de conectarse, puede utilizar la funcionalidad de pgAdmin para monitorear y administrar el clúster de Aurora PostgreSQL en el puerto de PostgreSQL.



Para obtener más información, la página web de pgAdmin.

Uso de Babelfish

A continuación, puede encontrar información de uso para Babelfish, incluidas algunas de las diferencias entre trabajar con Babelfish y SQL Server, y entre Babelfish y las bases de datos PostgreSQL.

Temas

- Obtención de información del catálogo del sistema de Babelfish
- Administración de permisos y control de acceso de Babelfish para Aurora PostgreSQL
- Diferencias entre Babelfish for Aurora PostgreSQL y SQL Server
- Uso de las características de Babelfish con implementación limitada
- Mejora del rendimiento de consultas de Babelfish
- Uso de las extensiones Aurora PostgreSQL con Babelfish
- Babelfish admite servidores enlazados
- Uso de la búsqueda de texto completo en Babelfish
- Babelfish admite tipos de datos geoespaciales
- Descripción de las particiones en Babelfish

Obtención de información del catálogo del sistema de Babelfish

Puede obtener información sobre los objetos de base de datos que están almacenados en su clúster de Babelfish mediante la consulta de muchas de las mismas vistas del sistema que se utilizan en SQL Server. Cada nueva versión de Babelfish agrega compatibilidad con más vistas del sistema. Para ver una lista de las vistas disponibles actualmente, consulte la tabla SQL Server system catalog views.

Estas vistas del sistema proporcionan información del catálogo del sistema (`sys.schemas`). En el caso de Babelfish, estas vistas contienen esquemas de sistema tanto de SQL Server como de PostgreSQL. Para consultar la información del catálogo del sistema en Babelfish, puede utilizar el puerto de TDS o el puerto de PostgreSQL, como se muestra en los siguientes ejemplos.

- Consulte el puerto T-SQL mediante **`sqlcmd`** u otro cliente de SQL Server.

```
1> SELECT * FROM sys.schemas
2> GO
```

Esta consulta devuelve los esquemas del sistema de SQL Server y Aurora PostgreSQL, como se muestra a continuación.

```
name
------------------------------------------------------
demographic_dbo
public
sys
master_dbo
tempdb_dbo
...
```

- Consultar el puerto de PostgreSQL mediante **`psql`** o **`pgAdmin`**. En este ejemplo se utiliza el metacomando de esquemas de listas `psql` (`\dn`):

```
babelfish_db=> \dn
```

La consulta devuelve el mismo conjunto de resultados que el que devuelve `sqlcmd` en el puerto de T-SQL.

```
        List of schemas
             Name
-----------------------------

 demographic_dbo

 public
 sys
 master_dbo
 tempdb_dbo
...
```

Catálogos del sistema de SQL Server disponibles en Babelfish

En la tabla siguiente encontrará las vistas de SQL Server implementadas actualmente en Babelfish. Para obtener más información sobre los catálogos del sistema en SQL Server, consulte System Catalog Views (Transact-SQL) (Vistas del catálogo del sistema [Transact-SQL]) en la documentación de Microsoft.

Nombre de la vista	Descripción o limitación de babelfish (si procede)
`sys.all_columns`	Todas las columnas de todas las tablas y vistas
`sys.all_objects`	Todos los objetos de todos los esquemas
`sys.all_sql_modules`	La unión de `sys.sql_modules` y `sys.system_sql_modules`
`sys.all_views`	Todas las vistas de todos los esquemas
`sys.columns`	Todas las columnas de tablas y vistas definidas por el usuario
`sys.configurations`	Compatibilidad con Babelfish limitada a una configuración de solo lectura.
`sys.data_spaces`	Contiene una fila para cada espacio de datos. Puede ser un grupo de archivos, un esquema de particiones o un grupo de archivos de datos FILESTREAM.
`sys.database_files`	Una vista por base de datos que contiene una fila por cada archivo de una base de datos tal y como se almacena en la propia base de datos.
`sys.database_mirroring`	Para obtener información, consulte sys.database_mirroring en la documentación de Microsoft Transact-SQL.
`sys.database_principals`	Para obtener información, consulte sys.database_principals en la documentación de Microsoft Transact-SQL.
`sys.database_role_members`	Para obtener información, consulte sys.database_role_members en la documentación de Microsoft Transact-SQL.

Nombre de la vista	Descripción o limitación de babelfish (si procede)
`sys.databases`	Todas las bases de datos de todos los esquemas
`sys.dm_exec_connections`	Para obtener información, consulte sys.dm_exec_connections en la documentación de Microsoft Transact-SQL.
`sys.dm_exec_sessions`	Para obtener información, consulte sys.dm_exec_sessions en la documentación de Microsoft Transact-SQL.
`sys.dm_hadr_database_replica_states`	Para obtener información, consulte sys.dm_hadr_database_replica_states en la documentación de Microsoft Transact-SQL.
`sys.dm_os_host_info`	Para obtener información, consulte sys.dm_os_host_info en la documentación de Microsoft Transact-SQL.
`sys.endpoints`	Para obtener información, consulte sys.endpoints en la documentación de Microsoft Transact-SQL.
`sys.indexes`	Para obtener información, consulte sys.indexes en la documentación de Microsoft Transact-SQL.
`sys.languages`	Para obtener información, consulte sys.languages en la documentación de Microsoft Transact-SQL.
`sys.schemas`	Todos los esquemas
`sys.server_principals`	Todos los inicios de sesión y roles

Nombre de la vista	Descripción o limitación de babelfish (si procede)
`sys.sql_modules`	Para obtener información, consulte sys.sql_modules en la documentación de Microsoft Transact-SQL.
`sys.sysconfigures`	Compatibilidad con Babelfish limitada a una sola configuración de lectura.
`sys.syscurconfigs`	Compatibilidad con Babelfish limitada a una configuración de solo lectura.
`sys.sysprocesses`	Para obtener información, consulte sys.sysprocesses en la documentación de Microsoft Transact-SQL.
`sys.system_sql_modules`	Para obtener información, consulte sys.system_sql_modules en la documentación de Microsoft Transact-SQL.
`sys.table_types`	Para obtener información, consulte sys.table_types en la documentación de Microsoft Transact-SQL.
`sys.tables`	Todas las tablas de un esquema
`sys.xml_schema_collections`	Para obtener información, consulte sys.xml_schema_collections en la documentación de Microsoft Transact-SQL.

PostgreSQL implementa catálogos de sistemas similares a las vistas de catálogo de objetos de SQL Server. Para obtener una lista completa de catálogos de sistemas, consulte System Catalogs en la documentación de PostgreSQL.

Exportaciones DDL compatibles con Babelfish

En las versiones 2.4.0 y 3.1.0 de Babelfish, Babelfish admite la exportación de DDL con varias herramientas. Por ejemplo, puede utilizar esta funcionalidad de SQL Server Management Studio

(SSMS) para generar los scripts de definición de datos para diversos objetos de una base de datos Babelfish para Aurora PostgreSQL A continuación, puede utilizar los comandos DDL generados en este script para crear los mismos objetos en otra base de datos Babelfish para Aurora PostgreSQL o SQL Server.

Babelfish permite exportaciones de DDL para los siguientes objetos en las versiones especificadas.

Lista de objetos	2.4.0	3.1.0
Tablas de usuario	Sí	Sí
Claves principales	Sí	Sí
Claves externas	Sí	Sí
Restricciones únicas	Sí	Sí
Índices	Sí	Sí
Restricciones de comprobación	Sí	Sí
Vistas	Sí	Sí
Procedimientos almacenados	Sí	Sí
Funciones definidas por el usuario	Sí	Sí
Funciones con valores de tabla	Sí	Sí
Desencadenadores	Sí	Sí
Tipos de datos definidos por el usuario	No	No
Tipos de tablas definidas por el usuario	No	No
Usuarios	No	No
Inicios de sesión	No	No
Secuencias	No	No
Roles	No	No

Limitaciones con los DDL exportados

- Utilice escotillas de escape antes de volver a crear los objetos con los DDL exportados: Babelfish no admite todos los comandos del script de DDL exportado. Utilice escotillas de escape para evitar los errores que se producen al volver a crear los objetos con los comandos de DDL en Babelfish. Para obtener más información acerca de las escotillas de escape, consulte Administración de la gestión de errores de Babelfish con escotillas de escape.
- Objetos que contienen restricciones CHECK con cláusulas COLLATE explícitas: los scripts con estos objetos generados desde una base de datos SQL Server tienen intercalaciones diferentes pero equivalentes a las de la base de datos Babelfish. Por ejemplo, algunas intercalaciones, como sql_latin1_general_cp1_cs_as, sql_latin1_general_cp1251_cs_as y latin1_general_cs_as, se generan como latin1_general_cs_as, que es la intercalación de Windows más cercana.

Administración de permisos y control de acceso de Babelfish para Aurora PostgreSQL

En Babelfish para Aurora PostgreSQL, puede administrar los permisos y el control de acceso para bases de datos, esquemas y objetos. En las siguientes tablas se describen los comandos de SQL específicos para conceder permisos en Babelfish con el fin de lograr diversos escenarios de control de acceso. Abarcará los casos de uso admitidos que se pueden implementar, así como las soluciones provisionales para los casos que actualmente no se admiten. Esto le permitirá configurar los permisos adecuados para cumplir con sus requisitos de seguridad y conformidad cuando trabaje con bases de datos de Babelfish.

Casos de uso admitidos

En la siguiente tabla se explican los casos de uso admitidos en Babelfish. En la tabla se muestran la acción necesaria para lograrlo y ejemplos de comandos de SQL para cada caso de uso.

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish
Permitir el inicio de sesión para realizar las operaciones SELECT/ DML/DDL en	Añadir el inicio de sesión al rol del servidor sysadmin	ALTER SERVER ROLE sysadmin ADD MEMBER `login`	Ninguno	Todas las versiones

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish
cualquier base de datos				
Permitir el inicio de sesión para realizar las operaciones SELECT/DML/DDL en una base de datos	Hacer que el propietario de la base de datos inicie sesión	ALTER AUTHORIZATION ON DATABASE::`database` TO `login`	Una base de datos solo puede tener .un propietario.	Versión 3.4 y posteriores
Permitir que el usuario de la base de datos realice las operaciones SELECT/DML en un esquema	Conceder permiso al usuario de la base de datos en un esquema	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON SCHEMA::`schema` TO `user`	Ninguno	Versión 3.6 y posteriores, 4.2 y posteriores
Permitir que el usuario de la base de datos realice las operaciones SELECT/DML en un esquema	Hacer que el usuario de la base de datos sea propietario del esquema en el momento de crearlo	CREATE SCHEMA `schema` AUTHORIZATION `user`	Actualmente no se admite el cambio de propiedad del esquema después de crearlo.	Versión 1.2 y posteriores
Permitir que el usuario de la base de datos realice las operaciones SELECT/DML en un objeto	Conceder permiso al usuario de la base de datos en un objeto	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT::`object` TO `user`	Ninguno	Todas las versiones

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish
Permitir que el usuario de la base de datos realice las operaciones SELECT/DML/DDL en una base de datos, incluida la eliminación de la misma	Agregación de usuario al rol fijo de base de datos de db_owner	ALTER ROLE db_owner ADD MEMBER `user`	Solo se pueden agregar usuarios al rol fijo de base de datos de db_owner. Actualmente no se admite agregar roles al rol db_owner.	Versión 4.5 y posteriores, 5.1 y posteriores
Permiso al usuario o a los miembros de un rol de base de datos personalizado para realizar solo operaciones SELECT en una base de datos	Agregación del usuario o rol al rol fijo de base de datos db_datareader	ALTER ROLE db_datareader ADD MEMBER `user` / `role`	Ninguno	Versión 4.5 y posteriores, 5.1 y posteriores
Permiso para que el usuario o los miembros de un rol de base de datos personalizado realicen solo operaciones DML en una base de datos	Agregación de usuario o rol al rol fijo de base de datos db_datawriter	ALTER ROLE db_datawriter ADD MEMBER `user` / `role`	Ninguno	Versión 4.5 y posteriores, 5.1 y posteriores

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish
Permiso al usuario o a los miembros de un rol de base de datos personalizado para que realicen solo operaciones DDL en una base de datos	Agregación de usuario o rol al rol fijo de base de datos db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Ninguno	Versión 4.5 y posteriores, 5.1 y posteriores
Permitir que el usuario o los miembros de un rol de base de datos personalizado solo puedan realizar las operaciones CREATE/ALTER/DROP roles personalizados, GRANT/REVOKE permisos sobre los objetos de una base de datos o CREATE SCHEMA en una base de datos	Agregación de usuario o rol al rol fijo de base de datos db_securityadmin	ALTER ROLE db_securityadmin ADD MEMBER `user` / `role`	Ninguno	Versión 4.5 y posteriores, 5.1 y posteriores

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish
Permita que el usuario o los miembros de un rol de base de datos personalizado solo puedan realizar las acciones CREATE/ALTER/DROP cualquier usuario, conceder y revocar el acceso a la base de datos y asignar las cuentas de usuario a los inicios de sesión o CREATE SCHEMA en una base de datos	Agregación de usuario o rol al rol fijo de base de datos db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Ninguno	Versión 4.5 y posteriores, 5.1 y posteriores
Permiso de inicio de sesión solo para las operaciones CREATE/DROP/ALTER en cualquier base de datos	Agregación de inicio de sesión al rol fijo del servidor dbcreator	ALTER SERVER ROLE dbcreator ADD MEMBER `login`	Solo se pueden modificar las bases de datos a las que tiene acceso el inicio de sesión de dbcreator.	Versión 4.5 y posteriores, 5.1 y posteriores

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish
Permiso del inicio de sesión solo para las operaciones CREATE/ALTER/DROP en cualquier inicio de sesión	Agregación de inicio de sesión al rol fijo del servidor securityadmin	ALTER SERVER ROLE securityadmin ADD MEMBER `login`	Ninguno	Versión 4.5 y posteriores, 5.1 y posteriores

Casos de uso no admitidos con las soluciones provisionales

En la siguiente tabla, se explican los casos de uso que Babelfish no admite, pero que se pueden llevar a cabo con una solución provisional.

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish para soluciones alternativas
Permiso al usuario para realizar operaciones SELECT/DML en un objeto/esquema junto con la opción de CONCEDER estos permisos a otros usuarios	CONCESIÓN de los permisos a todos los demás usuarios directamente	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT/SCHEMA::`object/sc` TO `user`	GRANT ... Actualmente no se admite WITH GRANT OPTION.	Versión 3.6 y posteriores, 4.2 y posteriores

Caso de uso	Acción	Comandos SQL	Comentarios	Compatibilidad de versiones de Babelfish para soluciones alternativas
Permitir que el usuario de la base de datos realice las operaciones SELECT/DML/DDL en una base de datos, incluida la eliminación de la misma	Agregación de miembros del rol al rol fijo de base de datos db_owner	ALTER ROLE db_owner ADD MEMBER `user`	Actualmente no se admite la agregación de roles al rol db_owner.	Versión 4.5 y posteriores, 5.1 y posteriores

Casos de uso no admitidos

En la siguiente tabla se explican los casos de uso no admitidos en Babelfish.

Caso de uso	Comentarios
Denegación al usuario o a los miembros de un rol de base de datos personalizado de la posibilidad de realizar SELECT en una base de datos	Aún no se admite el rol fijo de base de datos db_denydatareader
Denegación a un usuario o a los miembros de un rol de base de datos personalizado de la posibilidad de realizar DML en una base de datos	Actualmente no se admite el rol fijo de base de datos db_denydatawriter.
Permiso del inicio de sesión solo para KILL en cualquier conexión de base de datos	Actualmente no se admite el rol fijo de servidor processadmin.

Caso de uso	Comentarios
Permiso del inicio de sesión solo para agregar o eliminar servidores enlazados	Actualmente no se admite el rol de servidor fijo setupadmin.

Tratamiento de las diferencias de propiedad de los objetos tras la actualización

Las versiones 4.6 y posteriores, y 5.2 y posteriores de Babelfish incluyen un cambio en el manejo de la propiedad de los objetos a través del punto de conexión de TDS. Al crear nuevos objetos a través del punto de conexión de TDS, estos objetos ahora son propiedad del propietario del esquema y no del usuario actual. Es posible que este cambio de propiedad afecte al comportamiento de los permisos de los objetos nuevos en comparación con los objetos existentes al actualizar desde versiones anteriores a la 4.6 o 5.2.

Para resolver estas diferencias de propiedad, Babelfish proporciona la función `sys.generate_alter_ownership_statements()`. Esta función genera instrucciones SQL que alinean la propiedad del objeto con la propiedad del esquema.

Tenga en cuenta las siguientes limitaciones al abordar la propiedad del objeto:

- Los usuarios con permisos CREATE concedidos a través del punto de conexión de PostgreSQL no pueden crear objetos a través del punto de conexión de TDS en esos esquemas.
- No se recomienda modificar los permisos de los objetos de T-SQL a través del punto de conexión de PostgreSQL, ya que podría provocar un comportamiento incorrecto de T-SQL.
- Los permisos de acceso pueden diferir entre los objetos antiguos y los nuevos debido a la falta de coincidencia en su propiedad. Por ejemplo, tenga en cuenta un esquema propiedad de `sch_own` que incluye objetos propiedad de `dbo`. En este caso, los objetos propiedad de `dbo` que se crearon antes de la actualización podrían tener permisos de acceso diferentes en comparación con los objetos propiedad de `sch_own` que se crearon después de la actualización. Esto puede afectar a operaciones como SELECT e INSERT.

Si el clúster de base de datos incluye objetos creados en versiones de Babelfish anteriores a la 4.6 o 5.2, tenga en cuenta la posibilidad de alinear su propiedad.

Tratamiento de las diferencias de propiedad de los objetos

1. Conéctese a la base de datos `babelfish_db` en el clúster de base de datos mediante el punto de conexión de PostgreSQL.
2. Ejecuta el siguiente comando:

   ```
   SELECT * from sys.generate_alter_ownership_statements();
   ```

   Esto genera una lista de instrucciones SQL destinadas a estandarizar la propiedad entre los objetos.
3. Ejecute primero las instrucciones generadas en un entorno de prueba para validar su efecto antes de aplicarlas en el entorno de producción.

Le recomendamos que ejecute estas instrucciones para lograr un modelo de propiedad de objetos coherente en toda la base de datos.

Diferencias entre Babelfish for Aurora PostgreSQL y SQL Server

Babelfish es una función de Aurora PostgreSQL en evolución, con nuevas funciones añadidas en cada versión desde el lanzamiento inicial de Aurora PostgreSQL 13.4. Se ha diseñado para proporcionar semántica T-SQL sobre PostgreSQL a través del dialecto T-SQL mediante el puerto TDS. Cada nueva versión de Babelfish agrega más características y funciones que se alinean mejor con la funcionalidad y el comportamiento de T-SQL, tal como se muestra en la tabla Funcionalidades compatibles con Babelfish por versión. Para obtener mejores resultados al trabajar con Babelfish, le recomendamos que comprenda las diferencias que existen actualmente entre el T-SQL admitido por SQL Server y Babelfish de la última versión. Para obtener más información, consulte Diferencias de T-SQL en Babelfish.

Además de las diferencias entre T-SQL compatible con Babelfish y SQL Server, es posible que también deba considerar los problemas de interoperabilidad entre Babelfish y PostgreSQL en el contexto del clúster de base de datos de Aurora PostgreSQL. Tal como se ha mencionado anteriormente, Babelfish admite la semántica T-SQL sobre PostgreSQL a través del dialecto T-SQL mediante el puerto TDS. Además, también se puede acceder a la base de datos de Babelfish a través del puerto estándar de PostgreSQL con instrucciones SQL de PostgreSQL. Si está considerando usar las funciones de PostgreSQL y Babelfish en una implementación de producción, debe conocer los posibles problemas de interoperabilidad entre los nombres de esquema, los identificadores, los permisos, la semántica transaccional, los conjuntos de resultados múltiples, las

intercalaciones predeterminadas, etc. En términos simples, cuando las instrucciones de PostgreSQL o el acceso a PostgreSQL se producen en el contexto de Babelfish, pueden producirse interferencias entre PostgreSQL y Babelfish, lo que puede afectar potencialmente a la sintaxis, la semántica y la compatibilidad cuando se lancen nuevas versiones de Babelfish. Para obtener información completa y orientación sobre todas las consideraciones, consulte la guía sobre la interoperabilidad de Babelfish en la documentación de Babelfish for PostgreSQL.

> **Note**
>
> Antes de usar las funciones nativas de PostgreSQL y las funciones de Babelfish en el mismo contexto de la aplicación, le recomendamos encarecidamente que tenga en cuenta los problemas analizados en la guía sobre la interoperabilidad de Babelfish de la documentación de Babelfish for PostgreSQL. Estos problemas de interoperabilidad (Aurora, PostgreSQL y Babelfish) solo son relevantes si tiene previsto usar la instancia de base de datos de PostgreSQL en el mismo contexto de la aplicación que Babelfish.

Temas

- Volcado y restauración de Babelfish
- Diferencias de T-SQL en Babelfish
- Niveles de aislamiento de transacciones en Babelfish

Volcado y restauración de Babelfish

A partir de las versiones 4.0.0 y 3.4.0, los usuarios de Babelfish pueden ya utilizar el volcado y la restauración para hacer copias de seguridad y restaurar sus bases de datos. Para obtener más información, consulte Volcado y restauración de Babelfish. Esta característica se basa en las utilidades de volcado y restauración de PostgreSQL. Para obtener más información, consulte pg_dump y pg_restore. Para utilizar eficazmente esta característica en Babelfish, es necesario utilizar herramientas basadas en PostgreSQL que estén adaptadas específicamente para Babelfish. La característica de copia de seguridad y restauración de Babelfish difiere considerablemente de la de SQL Server. Para obtener más información sobre estas diferencias, consulte Dump and restore functionality differences: Babelfish and SQL Server. Babelfish para Aurora PostgreSQL proporciona capacidades adicionales para realizar copias de seguridad y restaurar clústeres de bases de datos de Amazon Aurora PostgreSQL. Para obtener más información, consulte Copias de seguridad y restauración de un clúster de base de datos de Amazon Aurora.

Diferencias de T-SQL en Babelfish

A continuación, encontrará una tabla de la funcionalidad de T-SQL compatible en la versión actual de Babelfish con algunas notas sobre las diferencias en el comportamiento con respecto al de SQL Server.

Para obtener más información acerca de la compatibilidad con versiones secundarias, consulte Funcionalidades compatibles con Babelfish por versión. Para obtener información acerca de las características no compatibles actualmente, consulte Funcionalidades no compatibles con Babelfish.

Babelfish está disponible con Aurora PostgreSQL-Compatible Edition. Para obtener más información sobre las versiones de Babelfish, consulte las Notas de la versión de Aurora PostgreSQL.

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia
\ (carácter de continuación de línea)	El carácter de continuación de línea (barra invertida antes de una nueva línea) para cadenas de caracteres y hexadecimales no es actualmente compatible. Para las cadenas de caractere s, la barra invertida de nueva línea se interpreta como caractere s en la cadena. Para las cadenas hexadecimales, la barra invertida de nueva línea da como resultado un error de sintaxis.
@@version	El formato del valor devuelto por `@@version` es ligeramente diferente del valor devuelto por SQL Server. Es posible que el código no funcione correctamente si depende del formato de `@@version` .
Funciones de agregación	Las funciones agregadas con parcialmente compatibles (se admiten AVG, COUNT, COUNT_BIG, GROUPING, MAX, MIN, STRING_AGG y SUM). Para ver una lista de las funciones agregadas no admitidas, consulte Funciones que no son compatibles.
ALTER TABLE	Admite la adición o eliminación de una sola columna o restricci ón.
ALTER TABLE..ALTER COLUMN	Actualmente no se pueden especificar NULL y NOT NULL. Para cambiar la nulabilidad de una columna, utilice la instrucción postgreSQL ALTER TABLE..{SET|DROP} NOT NULL.

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia

AT TIME ZONE	Durante la transición del horario de verano (DST) al horario estándar, el periodo superpuesto se muestra utilizando la diferencia de hora estándar. Para aclarar, considere el siguiente ejemplo:
	`SELECT CONVERT(DATETIME2(0), '2022-10-30T02:00:00', 126) AT TIME ZONE 'Central European Standard Time';`
	`GO;`
	`Result: 2022-10-30 02:00:00 +01:00`

Nombres de columna en blanco sin alias de columna	Las utilidades `sqlcmd` y `psql` gestionan columnas con nombres en blanco de manera diferente: • `sqlcmd` de SQL Server devuelve un nombre de columna en blanco. • `psql` de PostgreSQL devuelve un nombre de columna generado.
Función CHECKSUM	Babelfish y SQL Server utilizan diferentes algoritmos de hash para la función CHECKSUM. Como resultado, los valores hash generados por la función CHECKSUM en Babelfish pueden ser diferentes de los generados por la función CHECKSUM en SQL Server.
Columna predeterminada	Al crear un valor predeterminado de columna, se ignora el nombre de la restricción. Para eliminar un valor predeterminado de columna, utilice la sintaxis siguiente: `ALTER TABLE...ALTER COLUMN..DROP DEFAULT...`

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia
Constraint_name	En SQL Server, los nombres de las restricciones deben ser únicos dentro del esquema al que pertenece la tabla. Sin embargo, en Babelfish, esto solo se aplica a las restricciones PRIMARY KEY y UNIQUE. Otros tipos de restricciones no están sujetos a esta limitación.
Restricciones	PostgreSQL no admite activar y desactivar restricciones individuales. La instrucción se omite y se produce una advertenc ia.
Restricciones con IGNORE_DUP_KEY	Las restricciones se crean sin esta propiedad.
CREATE, ALTER, DROP SERVER ROLE	ALTER SERVER ROLE solo se admite para `sysadmin`. No se admite el resto de sintaxis. El usuario de T-SQL de Babelfish tiene una experiencia similar a SQL Server para los conceptos de inicio de sesión (entidad principal de servidor), base de datos y usuario de base de datos (entidad principal de base de datos).
Las cláusulas CREATE, ALTER LOGIN se admiten con sintaxis limitada	CREATE LOGIN... Se admiten las cláusulas PASSWORD, ...DEFAULT_DATABASE y ... DEFAULT_L ANGUAGE. ALTER LOGIN... Se admite la cláusula PASSWORD, pero la cláusula ALTER LOGIN... OLD_PASSW ORD no se admite. Solo un inicio de sesión que sea miembro del administrador del sistema puede modificar una contraseña.
Intercalación que distingue entre mayúsculas de minúsculas CREATE DATABASE	La instrucción CREATE DATABASE no admite las intercala ciones que distinguen mayúsculas de minúsculas.

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia
Palabras clave y cláusulas de CREATE DATABASE	No se admiten las opciones excepto COLLATE y CONTAINMENT=NONE. La cláusula COLLATE se acepta y siempre se establece en el valor de `babelfishpg_tsql.server_collation_name` .
Cláusulas CREATE SCHEMA... admitidas	Puede utilizar el comando CREATE SCHEMA para crear un esquema vacío. Utilice comandos adicionales para crear objetos de esquema.
Los valores de ID de base de datos son diferentes en Babelfish	Las bases de datos maestra y tempdb no serán los ID de base de datos 1 y 2.
La función de tipo de fecha FORMAT se admite con las siguientes limitaciones	No se admite el meridiano de un solo carácter. El formato "yyy" de SQL Server devuelve 4 dígitos para los años superiores a 1000, pero solo 3 dígitos para los demás. No se admiten los formatos "g" y "R" La configuración regional "vi-VN" es ligeramente diferente.
Identificadores superiores a 63 caracteres	PostgreSQL admite un máximo de 63 caracteres para los identificadores. Babelfish convierte los identificadores de más de 63 caracteres en un nombre que incluye un hash del nombre original. Por ejemplo, una tabla creada como "AB(ABC12345678901234567890123456789012345678901234567890123456789012345678901234567890" se podría convertir a "ABC12345678901234567890123456789012345678901234567890123456789012345678901234567890".

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia
Compatibilidad con columnas IDENTITY	Se admiten las columnas `IDENTITY` para los tipos de datos `tinyint`, `smallint`, `int`, `bigint`, `numeric` y `decimal`. SQL Server admite precisión de 38 lugares para los tipos de datos `numeric` y `decimal` en columnas IDENTITY. PostgreSQL admite precisión de 19 lugares para los tipos de datos `numeric` y `decimal` en columnas IDENTITY.
Índices con IGNORE_DUP_KEY	La sintaxis que crea un índice que incluye IGNORE_DUP_KEY crea un índice como si se omitiera esta propiedad.
Índices con más de 32 columnas	Un índice no puede incluir más de 32 columnas. Las columnas de índice incluidas se cuentan hacia el máximo en PostgreSQL, pero no en SQL Server.
Índices (en clúster)	Los índices en clúster se crean como si NONclústerED se hubiera especificado.
Cláusulas de índices	Se ignoran las siguientes cláusulas: FILLFACTOR, ALLOW_PAGE_LOCKS, ALLOW_ROW_LOCKS, PAD_INDEX, STATISTICS_NORECOMPUTE, OPTIMIZE_FOR_SEQUENTIAL_KEY, SORT_IN_TEMPDB, DROP_EXISTING, ONLINE, COMPRESSION_DELAY, MAXDOP y DATA_COMPRESSION
Compatibilidad con JSON	El orden de los pares nombre-valor no está garantizado. Sin embargo, el tipo de matriz no se ve afectado.
Objetos LOGIN	No se admiten todas las opciones para objetos LOGIN, excepto para DEFAULT_DATABASE, DEFAULT_LANGUAGE, ENABLE, DISABLE.
Función NEWSEQUENTIALID	Implementado como NEWID; el comportamiento secuencial no está garantizado. Al llamar a `NEWSEQUENTIALID` , PostgreSQL genera un nuevo valor de GUID.

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia
Las siguientes limitaciones admiten la cláusula OUTPUT	OUTPUT y OUTPUT INTO no se admiten en la misma consulta de DML. No se admiten las referencias a la tabla que no es de destino de las operaciones UPDATE o DELETE en una cláusula OUTPUT. OUTPUT... DELETED *, INSERTED * no se admiten en la misma consulta.
Límite de parámetros de procedimiento o función	Babelfish admite un máximo de 100 parámetros para un procedimiento o función.
ROWGUIDCOL	Esta cláusula se omite actualmente. Las consultas que referenci an a `$GUIDGOL` provocan un error de sintaxis.
Compatibilidad con objetos SEQUENCE	Los objetos SEQUENCE se admiten en los tipos de datos tinyint, smallint, int, bigint, numéricos y decimales. Aurora PostgreSQL admite la precisión de 19 posiciones para tipos de datos numéricos y decimales en un objeto SEQUENCE.
Roles de nivel de servidor	Se puede usar el rol de nivel de servidor `sysadmin`. No se pueden usar otros roles de nivel de servidor (distintos de `sysadmin`).
Roles de nivel de base de datos distintos de `db_owner`	Se admiten los roles de nivel de base de datos `db_owner` y los roles de nivel de base de datos definidos por el usuario. No se pueden usar otros roles de nivel de base de datos (distintos de db_owner).
Palabra clave `SPARSE` de SQL	La palabra clave SPARSE se acepta e ignora.
Cláusula de palabras clave de SQL `ON filegroup`	Esta cláusula se omite actualmente.
Palabras clave de SQL `CLUSTERED` y `NONCLUSTE RED` para índices y restricci ones	Babelfish acepta e ignora las palabras clave `CLUSTERED` y `NONCLUSTERED` .

Funcionalidad o sintaxis	Descripción del comportamiento o de la diferencia
`sysdatabases.cmptlevel`	`sysdatabases.cmptlevel` siempre se establece en 120.
tempdb no se reinicializa al reiniciar	Los objetos permanentes (como tablas y procedimientos) creados en tempdb no se eliminan cuando se reinicia la base de datos.
Grupo de archivos TEXTIMAGE_ON	Babelfish ignora la cláusula `TEXTIMAGE_ON` *`filegroup`* .
Precisión del tiempo	Babelfish admite una precisión de 6 dígitos para los segundos fraccionados. No se prevén efectos negativos con este comportamiento.
Niveles de aislamiento de transacciones	READUNCOMMITTED se trata igual que READCOMMITTED.
Columnas calculadas virtuales (no persistentes)	Las columnas calculadas virtuales se crean como persistentes.
Sin cláusula SCHEMABINDING	Esta cláusula no se admite en funciones, procedimientos, desencadenadores ni vistas. Se crea el objeto, pero como si se hubiera especificado WITH SCHEMABINDING.

Niveles de aislamiento de transacciones en Babelfish

Babelfish admite los niveles de aislamiento de transacciones `READ UNCOMMITTED`, `READ COMMITTED` y `SNAPSHOT` A partir de la versión 3.4 de Babelfish, se admiten los niveles de aislamiento adicionales: `REPEATABLE READ` y `SERIALIZABLE`. Todos los niveles de aislamiento de Babelfish son compatibles con el comportamiento de los niveles de aislamiento correspondientes en PostgreSQL. SQL Server y Babelfish utilizan diferentes mecanismos subyacentes para implementar los niveles de aislamiento de las transacciones (bloqueo del acceso simultáneo, bloqueo de las transacciones, gestión de errores, etc.). Además, existen algunas diferencias sutiles en el posible funcionamiento del acceso simultáneo para diferentes cargas de trabajo. Para obtener más información sobre este comportamiento de PostgreSQL, consulte Aislamiento de transacciones.

Temas

- Información general sobre los niveles de aislamiento de transacciones
- Configuración de los niveles de aislamiento de las transacciones
- Activación o desactivación de los niveles de aislamiento de transacciones
- Comparación de los niveles de aislamiento de Babelfish y SQL Server

Información general sobre los niveles de aislamiento de transacciones

Los niveles de aislamiento de transacciones originales de SQL Server se basan en un bloqueo pesimista en el que solo existe una copia de los datos y las consultas deben bloquear los recursos, por ejemplo las filas, antes de acceder a ellos. Más adelante, se introdujo una variación del nivel de aislamiento `READ COMMITTED`. Esto permite el uso de versiones de filas para proporcionar una mejor simultaneidad entre lectores y escritores mediante el acceso sin bloqueo. Además, está disponible un nuevo nivel de aislamiento denominado `SNAPSHOT`. También utiliza versiones de filas para ofrecer una mayor simultaneidad que el nivel de aislamiento `REPEATABLE READ`, ya que evita el bloqueo compartido de los datos de lectura, que se retienen hasta el final de la transacción.

A diferencia de SQL Server, todos los niveles de aislamiento de transacciones de Babelfish se basan en el bloqueo positivo (MVCC). Cada transacción ve una instantánea de los datos al principio de la declaración (`READ COMMITTED`) o al principio de la transacción (`REPEATABLE READ`, `SERIALIZABLE`), independientemente del estado actual de los datos subyacentes. Por lo tanto, el comportamiento de ejecución de las transacciones simultáneas en Babelfish puede diferir del de SQL Server.

Por ejemplo, consideremos una transacción con un nivel de aislamiento SERIALIZABLE que inicialmente está bloqueada en SQL Server, pero que se realiza correctamente más adelante. Puede terminar fallando en Babelfish debido a un conflicto de serialización con una transacción simultánea que lee o actualiza las mismas filas. También puede haber casos en los que la ejecución de varias transacciones simultáneas arroje un resultado final diferente en Babelfish en comparación con SQL Server. Las aplicaciones que utilizan niveles de aislamiento deben probarse exhaustivamente para detectar escenarios de simultaneidad.

Niveles de aislamiento en SQL Server	Nivel de aislamiento de Babelfish	Nivel de aislamiento de PostgreSQL	Comentarios
READ UNCOMMITTED	READ UNCOMMITTED	READ UNCOMMITTED	READ UNCOMMITTED es igual que READ COMMITTED en Babelfish o PostgreSQL
READ COMMITTED	READ COMMITTED	READ COMMITTED	El READ COMMITTED de SQL Server se basa en el bloqueo negativo, mientras que el READ COMMITTED de Babelfish se basa en instantáneas (MVCC).
READ COMMITTED SNAPSHOT	READ COMMITTED	READ COMMITTED	Ambas están basadas en instantáneas (MVCC), pero no son exactamente iguales.
SNAPSHOT	SNAPSHOT	REPEATABLE READ	Exactamente igual.
REPEATABLE READ	REPEATABLE READ	REPEATABLE READ	El REPEATABLE READ de SQL Server se basa en el bloqueo negativo, mientras que el REPEATABLE

Niveles de aislamiento en SQL Server	Nivel de aislamiento de Babelfish	Nivel de aislamiento de PostgreSQL	Comentarios
			READ de Babelfish se basa en instantáneas (MVCC).
SERIALIZABLE	SERIALIZABLE	SERIALIZABLE	El SERIALIZABLE de SQL Server se basa en un aislamiento negativo y el SERIALIZABLE de Babelfish se basa en instantáneas (MVCC).

> **Note**
>
> Las sugerencias de la tabla no son compatibles actualmente y su comportamiento se controla mediante la escotilla de escape predefinida de Babelfish `escape_hatch_table_hints`.

Configuración de los niveles de aislamiento de las transacciones

Utilice el siguiente comando para establecer el nivel de aislamiento de las transacciones:

Example

```
SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |
 SNAPSHOT | SERIALIZABLE }
```

Activación o desactivación de los niveles de aislamiento de transacciones

Los niveles de aislamiento de transacciones `REPEATABLE READ` y `SERIALIZABLE` están deshabilitados de forma predeterminada en Babelfish y hay que activarlos de manera explícita configurando la escotilla de escape `babelfishpg_tsql.isolation_level_serializable` o `babelfishpg_tsql.isolation_level_repeatable_read` como `pg_isolation` utilizando

`sp_babelfish_configure`. Para obtener más información, consulte Administración de la gestión de errores de Babelfish con escotillas de escape.

A continuación se muestran ejemplos de cómo habilitar o deshabilitar el uso de `REPEATABLE READ` y `SERIALIZABLE` en la sesión actual configurando sus respectivas escotillas de escape. Si lo desea, puede incluir un parámetro `server` para establecer la escotilla de escape de la sesión actual y de todas las sesiones nuevas subsiguientes.

Para habilitar el uso de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` solo en la sesión actual.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation'
```

Para habilitar el uso de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` en la sesión actual y en todas las nuevas sesiones simultáneas.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation',
 'server'
```

Para deshabilitar el uso de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` en la sesión actual y en las nuevas sesiones posteriores.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'off', 'server'
```

Para habilitar el uso de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` solo en la sesión actual.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation'
```

Para habilitar el uso de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` en la sesión actual y en todas las nuevas sesiones simultáneas.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation', 'server'
```

Para deshabilitar el uso de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` en la sesión actual y en las nuevas sesiones posteriores.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'off', 'server'
```

Comparación de los niveles de aislamiento de Babelfish y SQL Server

A continuación se muestran algunos ejemplos sobre los matices de la forma en que SQL Server y Babelfish implementan los niveles de aislamiento ANSI.

> **Note**
>
> - El nivel de aislamiento `REPEATABLE READ` y `SNAPSHOT` son iguales en Babelfish.
> - El nivel de aislamiento `READ UNCOMMITTED` y `READ COMMITTED` son iguales en Babelfish.

En el ejemplo siguiente, se muestra cómo crear la tabla base para todos los ejemplos que se mencionan a continuación:

```
CREATE TABLE employee (
    id sys.INT NOT NULL PRIMARY KEY,
    name sys.VARCHAR(255)NOT NULL,
    age sys.INT NOT NULL
);
INSERT INTO employee (id, name, age) VALUES (1, 'A', 10);
```

```
INSERT INTO employee (id, name, age) VALUES (2, 'B', 20);
INSERT INTO employee (id, name, age) VALUES (3, 'C', 30);
```

Temas

- READ UNCOMMITTED de Babelfish en comparación con el nivel de aislamiento de SQL Server READ UNCOMMITTED
- READ COMMITTED de Babelfish en comparación con el nivel de aislamiento de SQL Server READ COMMITTED
- READ COMMITTED de Babelfish en comparación con el nivel de aislamiento de SQL Server READ COMMITTED SNAPSHOT
- REPEATABLE READ de Babelfish en comparación con el nivel de aislamiento de SQL Server REPEATABLE READ
- SERIALIZABLE de Babelfish en comparación con el nivel de aislamiento de SQL Server SERIALIZABLE

READ UNCOMMITTED de Babelfish en comparación con el nivel de aislamiento de SQL Server **READ UNCOMMITTED**

La siguiente tabla proporciona detalles sobre las lecturas sucias cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `READ UNCOMMITTED` en SQL Server en comparación con la implementación de Babelfish.

Transacción 1	Transacción 2	**READ UNCOMMITTED** de SQL Server	**READ UNCOMMITTED** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	Ninguna	Ninguna
Inactividad en la transacción	`UPDATE employee SET age=0;`	Actualización realizada correctamente.	Actualización realizada correctamente.

Transacción 1	Transacción 2	**READ UNCOMMITTED** de SQL Server	**READ UNCOMMITTED** de Babelfish
Inactividad en la transacción	`INSERT INTO employee VALUES (4, 'D', 40);`	La inserción se ha realizado correctamente.	La inserción se ha realizado correctamente.
`SELECT * FROM employee;`	Inactividad en la transacción	La transacción 1 puede ver los cambios no confirmados de la transacción 2.	Igual que `READ COMMITTED` en Babelfish. Los cambios no confirmados de la transacción 2 no son visibles en la transacción 1.
Inactividad en la transacción	`COMMIT`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ve los cambios efectuados por la transacción 2.	Ve los cambios efectuados por la transacción 2.

`READ COMMITTED` de Babelfish en comparación con el nivel de aislamiento de SQL Server **`READ COMMITTED`**

La siguiente tabla proporciona detalles sobre el comportamiento de bloqueo de lectura y escritura cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `READ COMMITTED` en SQL Server en comparación con la implementación de Babelfish.

Transacción 1	Transacción 2	**READ COMMITTED** de SQL Server	**READ COMMITTED** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna

Transacción 1	Transacción 2	**READ COMMITTED** de SQL Server	**READ COMMITTED** de Babelfish
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`UPDATE employee SET age=100 WHERE id = 1;`	Actualización realizada correctamente.	Actualización realizada correctamente.
`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	Inactividad en la transacción	Paso bloqueado hasta que se confirme la transacción 2.	Los cambios en la transacción 2 aún no están visibles. Actualiza la fila con id=3.
Inactividad en la transacción	`COMMIT`	La transacción 2 se confirma correctamente. La transacción 1 ahora está desbloqueada y recibe la actualización de la transacción 2.	La transacción 2 se confirma correctamente.
`SELECT * FROM employee;`	Inactividad en la transacción	La transacción 1 actualiza la fila con el id = 1.	La transacción 1 actualiza la fila con el id = 3.

`READ COMMITTED` de Babelfish en comparación con el nivel de aislamiento de SQL Server **`READ COMMITTED SNAPSHOT`**

La siguiente tabla proporciona detalles sobre el comportamiento de bloqueo de las filas recién insertadas cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al

utilizar el nivel de aislamiento `READ COMMITTED SNAPSHOT` en SQL Server en comparación con la implementación de Babelfish `READ COMMITTED`.

Transacción 1	Transacción 2	**`READ COMMITTED SNAPSHOT`** de SQL Server	**`READ COMMITTED`** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Ninguna	Ninguna
`INSERT INTO employee VALUES (4, 'D', 40);`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`UPDATE employee SET age = 99;`	Paso bloqueado hasta que se confirme la transacción 1. La fila insertada está bloqueada por la transacción 1.	Se actualizaron tres filas. La fila recién insertada aún no está visible.
`COMMIT`	Inactividad en la transacción	Confirmación realizada correctamente. La transacción 2 ahora está desbloqueada.	Confirmación realizada correctamente.
Inactividad en la transacción	`SELECT * FROM employee;`	Las 4 filas tienen una age=99.	La fila con un id = 4 tiene un valor de antigüedad de 40, ya que la transacción 2 no la pudo ver durante la consulta de actualización.

Transacción 1	Transacción 2	**READ COMMITTED SNAPSHOT** de SQL Server	**READ COMMITTED** de Babelfish
			Las demás filas se actualizan a una age=99.

REPEATABLE READ de Babelfish en comparación con el nivel de aislamiento de SQL Server **REPEATABLE READ**

La siguiente tabla proporciona detalles sobre el comportamiento de bloqueo de lectura y escritura cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `REPEATABLE READ` en SQL Server en comparación con la implementación de Babelfish `REPEATABLE READ`.

Transacción 1	Transacción 2	**REPEATABLE READ** de SQL Server	**REPEATABLE READ** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ninguna	Ninguna
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Inactividad en la transacción	Ninguna	Ninguna

Transacción 1	Transacción 2	**REPEATABLE READ** de SQL Server	**REPEATABLE READ** de Babelfish
Inactividad en la transacción	`SELECT * FROM employee WHERE id != 1;`	Ninguna	Ninguna
Inactividad en la transacción	`SELECT * FROM employee;`	La transacción 2 se bloquea hasta que se confirme la transacción 1.	La transacción 2 se lleva a cabo con normalidad.
`COMMIT`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`SELECT * FROM employee;`	La actualización de la transacción 1 está visible.	La actualización de la transacción 1 no está visible.
`COMMIT`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`SELECT * FROM employee;`	ve la actualización de la transacción 1.	ve la actualización de la transacción 1.

La siguiente tabla proporciona detalles sobre el comportamiento de bloqueo de lectura y escritura cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `REPEATABLE READ` en SQL Server en comparación con la implementación de Babelfish `REPEATABLE READ`.

Transacción 1	Transacción 2	**REPEATABLE READ** de SQL Server	**REPEATABLE READ** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION`	`SET TRANSACTION ISOLATION`	Ninguna	Ninguna

Transacción 1	Transacción 2	**REPEATABLE READ** de SQL Server	**REPEATABLE READ** de Babelfish
`LEVEL REPEATABL E READ;`	`LEVEL REPEATABL E READ;`		
`UPDATE employee SET name='A_T XN1' WHERE id=1;`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`UPDATE employee SET name='A_T XN2' WHERE id=1;`	Transacción 2 bloqueada.	Transacción 2 bloqueada.
`COMMIT`	Inactividad en la transacción	La confirmación se ha realizado correctam ente y la transacción 2 se ha desbloque ado.	La confirmación se ha realizado correctam ente y la transacci ón 2 falló debido a un error de tipo “No se pudo serializar el acceso debido a una actualización simultánea”.
Inactividad en la transacción	`COMMIT`	Confirmación realizada correctam ente.	La transacción 2 ya se ha cancelado.
Inactividad en la transacción	`SELECT * FROM employee;`	La fila con id=1 tiene name='A_TX2'.	La fila con id=1 tiene name='A_TX1'.

La siguiente tabla proporciona detalles sobre el comportamiento de las lecturas fantasma cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `REPEATABLE READ` en SQL Server en comparación con la implementación de Babelfish `REPEATABLE READ`.

Transacción 1	Transacción 2	**REPEATABLE READ** de SQL Server	**REPEATABLE READ** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`INSERT INTO employee VALUES (4, 'NewRowName', 20);`	La transacción 2 se lleva a cabo sin ningún tipo de bloqueo.	La transacción 2 se lleva a cabo sin ningún tipo de bloqueo.
Inactividad en la transacción	`SELECT * FROM employee;`	La fila recién insertada está visible.	La fila recién insertada está visible.
Inactividad en la transacción	`COMMIT`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	La nueva fila insertada por la transacción 2 está visible.	La nueva fila insertada por la transacción 2 no está visible.
`COMMIT`	Inactividad en la transacción	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	La fila recién insertada está visible.	La fila recién insertada está visible.

La siguiente tabla proporciona detalles sobre cuándo se ejecutan transacciones simultáneas y los diferentes resultados finales cuando se utiliza el nivel de aislamiento `REPEATABLE READ` en SQL Server en comparación con la implementación de Babelfish `REPEATABLE READ`.

Transacción 1	Transacción 2	**`REPEATABLE READ`** de SQL Server	**`REPEATABLE READ`** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Ninguna	Ninguna
`UPDATE employee SET age = 100 WHERE age IN (SELECT MIN(age) FROM employee);`	Inactividad en la transacción	La transacción 1 actualiza la fila con el id 1.	La transacción 1 actualiza la fila con el id 1.
Inactividad en la transacción	`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	La transacción 2 está bloqueada porque la instrucción SELECT intenta leer las filas bloqueadas por la consulta UPDATE en la transacción 1.	La transacción 2 se realiza sin ningún bloqueo, ya que la lectura nunca se bloquea, la instrucción SELECT se ejecuta y, finalmente, la fila con el id = 3 se actualiza, ya que los cambios en la transacción 1 aún no están visibles.
Inactividad en la transacción	`SELECT * FROM employee;`	Este paso se ejecuta después de que se haya confirmado la transacción 1. La fila con el id = 1	La fila con un id = 3 se actualiza mediante la transacción 2.

Transacción 1	Transacción 2	**REPEATABLE READ** de SQL Server	**REPEATABLE READ** de Babelfish
		se actualiza con la transacción 2 del paso anterior y está visible aquí.	
`COMMIT`	Inactividad en la transacción	La transacción 2 ahora está desbloqueada.	Confirmación realizada correctamente.
Inactividad en la transacción	`COMMIT`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ambas transacciones ejecutan una actualización en la fila con un id = 1.	Las distintas filas se actualizan por las transacciones 1 y 2.

SERIALIZABLE de Babelfish en comparación con el nivel de aislamiento de SQL Server **SERIALIZABLE**

La siguiente tabla proporciona detalles sobre los bloqueos de rango cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `SERIALIZABLE` en SQL Server en comparación con la implementación de Babelfish `SERIALIZABLE`.

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Ninguna	Ninguna

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`SELECT * FROM employee;`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`INSERT INTO employee VALUES (4, 'D', 35);`	La transacción 2 se bloquea hasta que se confirme la transacción 1.	La transacción 2 se lleva a cabo sin ningún tipo de bloqueo.
Inactividad en la transacción	`SELECT * FROM employee;`	Ninguna	Ninguna
`COMMIT`	Inactividad en la transacción	La transacción 1 se confirma correctamente. La transacción 2 ahora está desbloqueada.	La transacción 1 se confirma correctamente.
Inactividad en la transacción	`COMMIT`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	La fila recién insertada está visible.	La fila recién insertada está visible.

La siguiente tabla proporciona detalles sobre cuándo se ejecutan transacciones simultáneas y los diferentes resultados finales cuando se utiliza el nivel de aislamiento `SERIALIZABLE` en SQL Server en comparación con la implementación de Babelfish `SERIALIZABLE`.

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Ninguna	Ninguna
Inactividad en la transacción	`INSERT INTO employee VALUES (4, 'D', 40);`	Ninguna	Ninguna
`UPDATE employee SET age =99 WHERE id = 4;`	Inactividad en la transacción	La transacción 1 se bloquea hasta que se confirme la transacción 2.	La transacción 1 se lleva a cabo sin ningún tipo de bloqueo.
Inactividad en la transacción	`COMMIT`	La transacción 2 se confirma correctamente. La transacción 1 ahora está desbloqueada.	La transacción 2 se confirma correctamente.
`COMMIT`	Inactividad en la transacción	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	La fila recién insertada es visible con un valor de age = 99.	La fila recién insertada es visible con un valor de age = 40.

En la siguiente tabla se proporcionan detalles cuando realiza la operación `INSERT` en una tabla con una restricción única. Muestra los resultados observados al utilizar el nivel de aislamiento `SERIALIZABLE` en SQL Server en comparación con la implementación de Babelfish `SERIALIZABLE`.

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Ninguna	Ninguna
Inactividad en la transacción	`INSERT INTO employee VALUES (4, 'D', 40);`	Ninguna	Ninguna
`INSERT INTO employee VALUES ((SELECT MAX(id)+1 FROM employee), 'E', 50);`	Inactividad en la transacción	La transacción 1 se bloquea hasta que se confirme la transacción 2.	La transacción 1 se bloquea hasta que se confirme la transacción 2.
Inactividad en la transacción	`COMMIT`	La transacción 2 se confirma correctamente. La transacción 1 ahora está desbloqueada.	La transacción 2 se confirma correctamente. La transacción 1 abortada por un error de tipo “el valor clave duplicado infringe una restricción única”.
`COMMIT`	Inactividad en la transacción	La transacción 1 se confirma correctamente.	La confirmación de la transacción 1 falla y no se pudo serializar el acceso debido a las dependencias de lectura o escritura entre las transacciones.

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`SELECT * FROM employee;`	Inactividad en la transacción	se inserta la fila (5, 'E', 50).	Solo existen 4 filas.

En Babelfish, las transacciones simultáneas que se ejecutan con nivel de aislamiento Serializable fallarán y generarán un error de anomalía de serialización si la ejecución de estas transacciones no es coherente con todas las posibles ejecuciones en serie (una a la vez) de esas transacciones.

Las siguientes tablas proporcionan detalles sobre la anomalía de la serialización cuando se ejecutan transacciones simultáneas. Muestra los resultados observados al utilizar el nivel de aislamiento `SERIALIZABLE` en SQL Server en comparación con la implementación de Babelfish `SERIALIZABLE`.

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ninguna	Ninguna
`UPDATE employee SET age=5 WHERE age=10;`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`SELECT * FROM employee;`	La transacción 2 se bloquea hasta que se confirme la transacción 1.	La transacción 2 se lleva a cabo sin ningún tipo de bloqueo.

Transacción 1	Transacción 2	**SERIALIZABLE** de SQL Server	**SERIALIZABLE** de Babelfish
Inactividad en la transacción	`UPDATE employee SET age=35 WHERE age=30;`	Ninguna	Ninguna
`COMMIT`	Inactividad en la transacción	La transacción 1 se confirma correctamente.	La transacción 1 se confirma primero y puede confirmarse correctamente.
Inactividad en la transacción	`COMMIT`	La transacción 2 se confirma correctamente.	La confirmación de la transacción 2 falla con un error de serialización, se revierte toda la transacción. Vuelva a intentar la transacción 2.
`SELECT * FROM employee;`	Inactividad en la transacción	Los cambios de ambas transacciones son visibles.	La transacción 2 se revirtió. Solo se ven los cambios en la transacción 1.

En Babelfish, la anomalía de serialización solo es posible si todas las transacciones simultáneas se ejecutan en el nivel de aislamiento `SERIALIZABLE`. En la siguiente tabla, tomemos el ejemplo anterior, pero establezcamos la transacción 2 en el nivel de aislamiento `REPEATABLE READ`.

Transacción 1	Transacción 2	Niveles de aislamiento de SQL Server	Niveles de aislamiento de Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Ninguna	Ninguna

Transacción 1	Transacción 2	Niveles de aislamiento de SQL Server	Niveles de aislamiento de Babelfish
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Ninguna	Ninguna
`SELECT * FROM employee;`	Inactividad en la transacción	Ninguna	Ninguna
`UPDATE employee SET age=5 WHERE age=10;`	Inactividad en la transacción	Ninguna	Ninguna
Inactividad en la transacción	`SELECT * FROM employee;`	La transacción 2 se bloquea hasta que se confirme la transacción 1.	La transacción 2 se lleva a cabo sin ningún tipo de bloqueo.
Inactividad en la transacción	`UPDATE employee SET age=35 WHERE age=30;`	Ninguna	Ninguna
`COMMIT`	Inactividad en la transacción	La transacción 1 se confirma correctamente.	La transacción 1 se confirma correctamente.
Inactividad en la transacción	`COMMIT`	La transacción 2 se confirma correctamente.	La transacción 2 se confirma correctamente.
`SELECT * FROM employee;`	Inactividad en la transacción	Los cambios de ambas transacciones son visibles.	Los cambios de ambas transacciones son visibles.

Uso de las características de Babelfish con implementación limitada

Cada nueva versión de Babelfish agrega compatibilidad con más características que se alinean mejor con la funcionalidad y el comportamiento de T-SQL. No obstante, hay algunas características y diferencias que no se admiten en la implementación actual. A continuación, encontrará información sobre las diferencias funcionales entre Babelfish y T-SQL, con algunas soluciones alternativas o notas de uso.

A partir de la versión 1.2.0 de Babelfish, actualmente, las siguientes características tienen implementaciones limitadas:

- Catálogos de SQL Server (vistas del sistema): los catálogos`sys.sysconfigures`, `sys.syscurconfigs` y `sys.configurations` admiten una única configuración de solo lectura. `sp_configure` no es compatible actualmente. Para obtener más información acerca de otras vistas de SQL Server implementadas por Babelfish, consulte Obtención de información del catálogo del sistema de Babelfish.
- Permisos GRANT: GRANT...TO PUBLIC se puede usar, pero GRANT..TO PUBLIC WITH GRANT OPTION no se puede usar actualmente.
- Cadena de propiedad SQL Server y limitación del mecanismo de permisos: en Babelfish, la cadena de propiedad de SQL Server funciona para vistas pero no para procedimientos almacenados. Esto significa que se debe conceder a los procedimientos acceso explícito a otros objetos propiedad del mismo propietario que los procedimientos de llamada. En SQL Server, otorgar permisos EXECUTE a la entidad que llama en el procedimiento es suficiente para llamar a otros objetos del mismo propietario. En Babelfish, también se deben conceder permisos a la entidad que llama sobre los objetos a los que se accede mediante el procedimiento.
- Resolución de referencias de objetos no calificadas (sin nombre de esquema): cuando un objeto SQL (procedimiento, vista, función o desencadenador) hace referencia a un objeto sin calificarlo con un nombre de esquema, SQL Server resuelve el nombre del esquema del objeto utilizando el nombre del esquema del objeto SQL en el que se produce la referencia. En la actualidad, Babelfish lo resuelve de forma diferente, utilizando el esquema predeterminado del usuario de la base de datos que ejecuta el procedimiento.
- Cambios de esquema, sesiones y conexiones predeterminados: si los usuarios cambian su esquema predeterminado con `ALTER USER...WITH DEFAULT SCHEMA`, el cambio surte efecto inmediatamente en esa sesión. No obstante, para otras sesiones conectadas actualmente pertenecientes al mismo usuario, los tiempos difieren de la siguiente manera:
  - Para SQL Server: el cambio surte efecto en todas las demás conexiones del usuario de manera inmediata.

  - Para Babelfish: el cambio surte efecto para este usuario solo en nuevas conexiones.
- Implementación de tipos de datos ROWVERSION y TIMESTAMP y configuración de escotilla de escape: los tipos de datos ROWVERSION y TIMESTAMP ahora son compatibles con Babelfish. Para utilizar ROWVERSION o TIMESTAMP en Babelfish, debe cambiar la configuración de la escotilla de escape `babelfishpg_tsql.escape_hatch_rowversion` de su valor predeterminado (estricto) a `ignore`. La implementación de Babelfish de los tipos de datos ROWVERSION y TIMESTAMP es en su mayoría idéntica semánticamente a SQL Server, con las siguientes excepciones:
  - La función @@DBTS incorporada se comporta de forma similar a la de SQL Server, pero con pequeñas diferencias. En lugar de devolver el último valor utilizado para `SELECT @@DBTS`, Babelfish genera una nueva marca de tiempo, debido al motor de base de datos PostgreSQL subyacente y a su implementación de control de simultaneidad multiversión (MVCC).
  - En SQL Server, cada fila insertada o actualizada obtiene un valor ROWVERSION/TIMESTAMP único. En Babelfish, a cada fila insertada actualizada con la misma instrucción se le asigna el mismo valor ROWVERSION/TIMESTAMP.

    Por ejemplo, cuando una instrucción UPDATE o una instrucción INSERT-SELECT afecta a varias filas, en SQL Server, las filas afectadas tienen todas valores diferentes en su columna ROWVERSION/TIMESTAMP. En Babelfish (PostgreSQL), las filas tienen el mismo valor.
  - En SQL Server, al crear una nueva tabla con SELECT-INTO, puede convertir un valor explícito (como NULL) a una columna ROWVERSION/TIMESTAMP que crear. Cuando se hace lo mismo en Babelfish, Babelfish asigna un valor real ROWVERSION/TIMESTAMP a cada fila de la nueva tabla por el usuario.

  Estas pequeñas diferencias en los tipos de datos ROWVERSION/TIMESTAMP no deberían tener un impacto negativo en las aplicaciones que se ejecutan en Babelfish.
- Creación de esquemas, propiedad y permisos: los permisos para crear objetos, y acceder a ellos en un esquema propiedad de un usuario que no pertenece a DBO (mediante `CREATE SCHEMA` *`schema name`* `AUTHORIZATION` *`user name`*) son diferentes para los usuarios de SQL Server y Babelfish que no pertenecen a DBO, tal y como se muestra en la siguiente tabla:

El usuario de la base de datos (no DBO) que es propietario del esquema puede hacer lo siguiente:	SQL Server	Babelfish

El usuario de la base de datos (no DBO) que es propietario del esquema puede hacer lo siguiente:	SQL Server	Babelfish
¿Crear objetos en el esquema sin concesion es adicionales por parte del DBO?	No	Sí
¿Acceder a objetos creados por DBO en el esquema sin concesiones adicionales?	Sí	No

- Sintaxis de CREATE OR ALTER VIEW / ALTER VIEW: la compatibilidad con estas sintaxis en Babelfish tiene las siguientes limitaciones:
  - Estas instrucciones no se pueden usar en vistas que tengan un disparador INSTEAD OF adjunto.
  - Estas instrucciones no se pueden utilizar en vistas que tengan otra vista basada en esta vista.

Mejora del rendimiento de consultas de Babelfish

Puede lograr un procesamiento de consultas más rápido en Babelfish si utiliza las sugerencias de consulta y el optimizador de PostgreSQL.

Temas

- Uso del plan de EXPLAIN para mejorar el rendimiento de las consultas en Babelfish
- Uso de sugerencias de consulta de T-SQL para mejorar el rendimiento de las consultas de Babelfish

También puede mejorar el rendimiento de la consulta mediante el procedimiento `sp_babelfish_volatility`. Para obtener más información, consulte sp_babelfish_volatility.

También puede mejorar el rendimiento de las consultas mediante la transformación de subconsultas y la caché de subconsultas. Para obtener más información, consulte Optimización de subconsultas correlacionadas en Aurora PostgreSQL.

Uso del plan de EXPLAIN para mejorar el rendimiento de las consultas en Babelfish

A partir de la versión 2.1.0, Babelfish incluye dos funciones que utilizan de forma transparente el optimizador de PostgreSQL para generar planes de consulta estimados y reales para consultas T-

SQL en el puerto de TDS. Estas funciones son similares al uso de SET STATISTICS PROFILE o SET SHOWPLAN_ALL con las bases de datos de SQL Server para identificar y mejorar las consultas de ejecución lenta.

> **Note**
>
> Actualmente no se admite la obtención de planes de consulta a partir de funciones, flujos de control y cursores.

En la tabla puede encontrar una comparación de las funciones de EXPLAIN del plan de consulta en SQL Server, Babelfish y PostgreSQL.

SQL Server	Babelfish	PostgreSQL
SHOWPLAN_ALL	BABELFISH_SHOWPLAN_ALL	EXPLAIN
STATISTICS PROFILE	BABELFISH_STATISTICS PROFILE	EXPLAIN ANALYZE
Utiliza el optimizador de SQL Server	Utiliza el optimizador de PostgreSQL	Utiliza el optimizador de PostgreSQL
Formato de entrada y salida de SQL Server	Formato de entrada de SQL Server y salida de PostgreSQL	Formato de entrada y salida de PostgreSQL
Se establece para la sesión	Se establece para la sesión	Se aplica a una instrucción específica
Admite lo siguiente: • SELECT • INSERT • UPDATE • DELETE • CURSOR	Admite lo siguiente: • SELECT • INSERT • UPDATE • DELETE • CREATE	Admite lo siguiente: • SELECT • INSERT • UPDATE • DELETE • CURSOR

SQL Server	Babelfish	PostgreSQL
• CREATE • EXECUTE • EXEC y funciones, incluido el flujo de control (CASE, WHILE-BREAK-CONTINUE, WAITFOR, BEGIN-END, IF-ELSE, etc.)	• EXECUTE • EXEC • RAISEERROR • THROW • PRINT • USE	• CREATE • EXECUTE

Utilice las funciones de Babelfish de la siguiente manera:

- SET BABELFISH_SHOWPLAN_ALL [ON|OFF]: configúrela en ON para generar un plan de ejecución de consultas estimado. Esta función implementa el comportamiento del comando `EXPLAIN` de PostgreSQL. Utilice este comando para obtener el plan de EXPLAIN de una consulta determinada.
- SET BABELFISH_STATISTICS PROFILE [ON|OFF]: configúrela en ON para los planes de ejecución de consultas reales. Esta función implementa el comportamiento del comando `EXPLAIN ANALYZE` de PostgreSQL.

Para obtener más información sobre `EXPLAIN` y `EXPLAIN ANALYZE` de PostgreSQL, consulte EXPLAIN en la documentación de PostgreSQL.

> **Note**
>
> A partir de la versión 2.2.0, puede configurar el parámetro `escape_hatch_showplan_all` en ignore a fin de evitar el uso del prefijo BABELFISH_ en la sintaxis de SQL Server para los comandos SET `SHOWPLAN_ALL` y `STATISTICS PROFILE`.

Por ejemplo, la siguiente secuencia de comandos activa la planificación de consultas y, a continuación, devuelve un plan de ejecución de consultas estimado para la sentencia SELECT sin ejecutar la consulta. En este ejemplo se utiliza la base de datos `northwind` de muestra de SQL Server mediante la herramienta de línea de comandos `sqlcmd` para consultar el puerto de TDS:

```
1> SET BABELFISH_SHOWPLAN_ALL ON
```

```
2> GO
1> SELECT t.territoryid, e.employeeid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE e.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO

QUERY PLAN


------------------------------------------------------------------------------


Query Text: SELECT t.territoryid, e.employeeid FROM
dbo.employeeterritories e, dbo.territories t
WHERE e.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=6231.74..6399.22 rows=66992 width=10)
  Sort Key: t.territoryid NULLS FIRST
  ->  Nested Loop  (cost=0.00..861.76 rows=66992 width=10)
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1264 width=4)
              Filter: ((territoryid)::"varchar" IS NOT NULL)
        ->  Materialize  (cost=0.00..1.79 rows=53 width=6)
              ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Cuando termine de revisar y ajustar la consulta, desactive la función como se muestra a continuación:

```
1> SET BABELFISH_SHOWPLAN_ALL OFF
```

Con BABELFISH_STATISTICS PROFILE configurado en ON, cada consulta ejecutada devuelve su conjunto de resultados habitual seguido de un conjunto de resultados adicional que muestra los planes de ejecución de consulta reales. Babelfish genera el plan de consulta que proporciona el conjunto de resultados más rápido cuando invoca la instrucción SELECT.

```
1> SET BABELFISH_STATISTICS PROFILE ON
1>
2> GO
1> SELECT e.employeeid, t.territoryid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE t.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO
```

Se devuelve el conjunto de resultados y el plan de consulta (en este ejemplo se muestra solo el plan de consulta).

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT e.employeeid, t.territoryid FROM
dbo.employeeterritories e, dbo.territories t
WHERE t.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=42.44..43.28 rows=337 width=10)
  Sort Key: t.territoryid NULLS FIRST


  ->  Hash Join  (cost=2.19..28.29 rows=337 width=10)
       Hash Cond: ((e.territoryid)::"varchar" = (t.territoryid)::"varchar")
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1270 width=36)
        ->  Hash  (cost=1.53..1.53 rows=53 width=6)
             ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Para obtener más información sobre cómo analizar sus consultas y los resultados que devuelve el optimizador de PostgreSQL, consulte explain.depesz.com. Para obtener más información sobre EXPLAIN y EXPLAIN ANALYZE de PostgreSQL, consulte EXPLAIN en la documentación de PostgreSQL.

Parámetros que controlan las opciones de EXPLAIN de Babelfish

Puede utilizar los parámetros que se muestran en la tabla siguiente para controlar el tipo de información que muestra el plan de consulta.

Parámetro	Descripción
babelfishpg_tsql.explain_buffers	Un valor booleano que activa (y desactiva) la información de uso del búfer para el optimizador. (Valor predeterminado: on; permitido: off, on).
babelfishpg_tsql.explain_costs	Un valor booleano que activa (y desactiva) la información estimada de inicio y el coste total

Parámetro	Descripción
	para el optimizador. (Valor predeterminado: on; permitido: off, on).
babelfishpg_tsql.explain_format	Especifica el formato de salida del plan de `EXPLAIN`. (Valor predeterminado: text; permitido: text, xml, json, yaml).
babelfishpg_tsql.explain_settings	Un valor booleano que activa (o desactiva) la inclusión de información sobre los parámetros de configuración en la salida del plan de EXPLAIN. (Valor predeterminado: on; permitido: off, on).
babelfishpg_tsql.explain_summary	Un valor booleano que activa (o desactiva) la información de resumen, como el tiempo total después del plan de consulta. (Valor predeterminado: on; permitido: off, on).
babelfishpg_tsql.explain_timing	Un valor booleano que activa (o desactiva) el tiempo de inicio real y el tiempo transcurrido en cada nodo de la salida. (Valor predeterminado: on; permitido: off, on).
babelfishpg_tsql.explain_verbose	Un valor booleano que activa (o desactiva) la versión más detallada de un plan de EXPLAIN. (Valor predeterminado: on; permitido: off, on).
babelfishpg_tsql.explain_wal	Un valor booleano que activa (o desactiva) la generación de información del registro WAL como parte de un plan de EXPLAIN. (Valor predeterminado: on; permitido: off, on).

Puede consultar los valores de cualquier parámetro relacionado con Babelfish en su sistema mediante el cliente de PostgreSQL o el cliente de SQL Server. Ejecute el siguiente comando para obtener los valores de los parámetros actuales:

```
1> execute sp_babelfish_configure '%explain%';
2> GO
```

En la siguiente salida, puede ver que todas las opciones de configuración de este clúster de base de datos de Babelfish en concreto están en sus valores predeterminados. En este ejemplo no se muestra toda la salida.

```
             name                    setting                     short_desc
--------------------------------- --------
 ---------------------------------------------------------
babelfishpg_tsql.explain_buffers   off      Include information on buffer usage
babelfishpg_tsql.explain_costs     on       Include information on estimated startup
 and total cost
babelfishpg_tsql.explain_format    text     Specify the output format, which can be
 TEXT, XML, JSON, or YAML
babelfishpg_tsql.explain_settings  off      Include information on configuration
 parameters
babelfishpg_tsql.explain_summary   on       Include summary information (e.g.,totaled
 timing information) after the query plan
babelfishpg_tsql.explain_timing    on       Include actual startup time and time spent
 in each node in the output
babelfishpg_tsql.explain_verbose   off      Display additional information regarding
 the plan
babelfishpg_tsql.explain_wal       off      Include information on WAL record
 generation

(8 rows affected)
```

Puede cambiar la configuración de estos parámetros con `sp_babelfish_configure`, tal y como se muestra en el siguiente ejemplo.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on';
2> GO
```

Si desea que la configuración sea permanente a nivel de todo el clúster, incluya la palabra clave server, como se muestra en el siguiente ejemplo.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on', 'server';
2> GO
```

Uso de sugerencias de consulta de T-SQL para mejorar el rendimiento de las consultas de Babelfish

A partir de la versión 2.3.0, Babelfish admite el uso de sugerencias de consultas con `pg_hint_plan`. En Aurora PostgreSQL, `pg_hint_plan` se instala de forma predeterminada. Para obtener más información sobre la extensión de PostgreSQL `pg_hint_plan`, consulte https://github.com/ossc-db/pg_hint_plan. Para obtener más información sobre la versión de esta extensión que admite Aurora PostgreSQL, consulte las versiones de extensiones de Amazon Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

El optimizador de consultas está bien diseñado para encontrar el plan de ejecución óptimo de una instrucción SQL. Al seleccionar un plan, el optimizador de consultas considera tanto el modelo de costes del motor como las estadísticas de columnas y tablas. Sin embargo, es posible que el plan sugerido no satisfaga las necesidades de sus conjuntos de datos. Por lo tanto, las sugerencias de consulta abordan los problemas de rendimiento para mejorar los planes de ejecución. Una `query hint` es la sintaxis que se añade al SQL estándar que indica al motor de base de datos cómo ejecutar la consulta. Por ejemplo, una sugerencia puede indicar al motor que siga un análisis secuencial y anule cualquier plan que haya seleccionado el optimizador de consultas.

Activación de las sugerencias de consulta de T-SQL en Babelfish

Actualmente, Babelfish ignora todas las sugerencias de T-SQL de forma predeterminada. Para aplicar las sugerencias de T-SQL, ejecute el comando `sp_babelfish_configure` con el valor de enable_pg_hint como ON.

```
EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on' [, 'server']
```

Para que la configuración sea permanente a nivel de todo el clúster, incluya la palabra clave server. Para configurar el ajuste solo para la sesión actual, no utilice server.

Una vez que `enable_pg_hint` está activado, Babelfish aplica las siguientes sugerencias de T-SQL.

- Sugerencias de INDEX
- Sugerencias de JOIN
- Sugerencia de FORCE ORDER
- Sugerencia de MAXDOP

Por ejemplo, se activa la siguiente secuencia de comandos en `pg_hint_plan`.

```
1> CREATE TABLE t1 (a1 INT PRIMARY KEY, b1 INT);
2> CREATE TABLE t2 (a2 INT PRIMARY KEY, b2 INT);
3> GO
1> EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on';
2> GO
1> SET BABELFISH_SHOWPLAN_ALL ON;
2> GO
1> SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2; --NO HINTS (HASH JOIN)
2> GO
```

No se aplica ninguna sugerencia a la instrucción SELECT. Se devuelve el plan de consulta sin sugerencias.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2
Hash Join (cost=60.85..99.39 rows=2260 width=16)
 Hash Cond: (t1.a1 = t2.a2)
 -> Seq Scan on t1 (cost=0.00..32.60 rows=2260 width=8)
 -> Hash (cost=32.60..32.60 rows=2260 width=8)
 -> Seq Scan on t2 (cost=0.00..32.60 rows=2260 width=8)
```

```
1> SELECT * FROM t1 INNER MERGE JOIN t2 ON t1.a1 = t2.a2;
2> GO
```

La sugerencia de consulta se aplica a la instrucción SELECT. El siguiente resultado muestra que se devuelve el plan de consulta con combinación de fusión.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT/*+ MergeJoin(t1 t2) Leading(t1 t2)*/ * FROM t1 INNER JOIN t2 ON
 t1.a1 = t2.a2
Merge Join (cost=0.31..190.01 rows=2260 width=16)
 Merge Cond: (t1.a1 = t2.a2)
```

```
 -> Index Scan using t1_pkey on t1 (cost=0.15..78.06 rows=2260 width=8)
 -> Index Scan using t2_pkey on t2 (cost=0.15..78.06 rows=2260 width=8)
```

```
1> SET BABELFISH_SHOWPLAN_ALL OFF;
2> GO
```

Limitaciones

Cuando utilice las sugerencias de consulta, tenga en cuenta las siguientes limitaciones:

- Si un plan de consultas se almacena en caché antes de que `enable_pg_hint` se active, las sugerencias no se aplicarán en la misma sesión. Se aplicarán en la nueva sesión.
- Si los nombres de los esquemas se proporcionan de forma explícita, las sugerencias no se pueden aplicar. Puede utilizar alias de tablas como solución alternativa.
- No se puede aplicar una sugerencia de consulta a las vistas ni a las subconsultas.
- Las sugerencias no funcionan para las instrucciones UPDATE/DELETE en las que haya JOIN.
- Se omite cualquier sugerencia de índice de un índice o una tabla que no existen.
- La sugerencia FORCE ORDER no funciona para JOIN HASH ni para JOIN que no sean ANSI.

Uso de las extensiones Aurora PostgreSQL con Babelfish

Aurora PostgreSQL proporciona extensiones para trabajar con otros servicios de AWS. Se trata de extensiones opcionales que admiten varios casos de uso, como el uso de Simple Storage Service (Amazon S3) con su clúster de bases de datos para importar o exportar datos.

- Para importar datos desde un bucket de Amazon S3 al clúster de base de datos de Babelfish, debe configurar la extensión `aws_s3` Aurora PostgreSQL. Esta extensión también le permite exportar datos desde el clúster de la base de datos de Aurora PostgreSQL a un bucket de Simple Storage Service (Amazon S3).
- AWS Lambda es un servicio automático que permite ejecutar código sin aprovisionar ni administrar servidores. Puede usar funciones de Lambda para hacer cosas como procesar notificaciones de eventos desde su instancia de base de datos. Para obtener más información sobre Lambda, consulte ¿Qué es AWS Lambda? en la Guía para desarrolladores de AWS Lambda. Para invocar funciones de Lambda desde el clúster de Babelfish, debe configurar la extensión `aws_lambda` Aurora PostgreSQL.

A fin de configurar estas extensiones para el clúster de Babelfish, primero debe conceder permiso al usuario interno de Babelfish para cargar las extensiones. Después de conceder el permiso, puede cargar las extensiones Aurora PostgreSQL.

Habilitación de las extensiones Aurora PostgreSQL en el clúster de la base de datos de Babelfish

Antes de poder cargar las extensiones `aws_s3` o `aws_lambda`, debe otorgar los privilegios necesarios al clúster de la base de datos de Babelfish.

El siguiente procedimiento usa la herramienta de la línea de comandos `psql` PostgreSQL para conectarse al clúster de la base de datos. Para obtener más información, consulte Conexión al clúster de bases de datos mediante psql. También puede usar pgAdmin. Para obtener más información, consulte Uso de pgAdmin para conectarse al clúster de bases de datos.

Este procedimiento carga tanto `aws_s3` como `aws_lambda`, uno tras otro. No es necesario que cargue ambas si desea usar solo una de estas extensiones. La extensión `aws_commons` es requerida por cada una de ellas y se carga de forma predeterminada como se muestra en la salida.

Para configurar el clúster de la base de datos de Babelfish con privilegios para las extensiones de Aurora PostgreSQL

1. Conéctese al clúster de base de datos de Babelfish. Use el nombre para el usuario “maestro” (-U) que especificó cuando creó el clúster de bases de datos de Babelfish. El valor predeterminado (`postgres`) se muestra en los ejemplos.

   Para Linux, macOS o Unix:

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com \
   -U postgres \
   -d babelfish_db \
   -p 5432
   ```

   Para Windows:

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com ^
   -U postgres ^
   -d babelfish_db ^
   -p 5432
   ```

   El comando responde con una solicitud para ingresar la contraseña del nombre de usuario (-U).

```
Password:
```

Ingrese la contraseña del nombre de usuario (-U) para el clúster de bases de datos. Cuando se conecte correctamente, verá una respuesta similar a la siguiente.

```
psql (13.4)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

2. Conceda privilegios al usuario interno de Babelfish para crear y cargar extensiones.

```
babelfish_db=> GRANT rds_superuser TO master_dbo;
GRANT ROLE
```

3. Cree y cargue la extensión `aws_s3`. La extensión `aws_commons` es necesaria y se instala de forma automática cuando se instala el `aws_s3`.

```
babelfish_db=> create extension aws_s3 cascade;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

4. Cree y cargue la extensión `aws_lambda`.

```
babelfish_db=> create extension aws_lambda cascade;
CREATE EXTENSION
babelfish_db=>
```

Uso de Babelfish con Simple Storage Service (Amazon S3)

Si aún no tiene un bucket de Simple Storage Service (Amazon S3) para usar con su clúster de bases de datos de Babelfish, puede crear uno. Debe proporcionar acceso a cualquier bucket de Simple Storage Service (Amazon S3) que desee usar.

Antes de intentar importar o exportar datos con un bucket de Simple Storage Service (Amazon S3), complete los siguientes pasos de una sola vez.

Para configurar el acceso de la instancia de base de datos de Babelfish al bucket de Simple Storage Service (Amazon S3)

1. Cree un bucket de Simple Storage Service (Amazon S3) para la instancia de Babelfish, si es necesario. Para ello, siga las instrucciones de Crear un bucket en la Guía del usuario de Amazon Simple Storage Service.
2. Cargue los archivos en el bucket de Simple Storage Service (Amazon S3). Para ello, siga los pasos de Agregar un objeto a un bucket en la Guía del usuario de Amazon Simple Storage Service.
3. Configure los permisos necesarios:

   - Para importar datos de Amazon S3, el clúster de base de datos de Babelfish necesita permiso para acceder al bucket. Se recomienda usar un rol de AWS Identity and Access Management (IAM) y adjuntar una política de IAM a ese rol para el clúster. Para ello, siga los pasos que se indican en Uso de un rol de IAM para obtener acceso a un bucket de Amazon S3.
   - Para exportar datos del clúster de base de datos de Babelfish, el clúster debe tener acceso al bucket de Amazon S3. Al igual que con la importación, se recomienda usar un rol y una política de IAM. Para ello, siga los pasos que se indican en Configuración del acceso a un bucket de Amazon S3.

Ahora puede usar Simple Storage Service (Amazon S3) con la extensión `aws_s3` con el clúster de base de datos de Babelfish.

Para importar datos de Simple Storage Service (Amazon S3) a Babelfish y para exportar datos de Babelfish a Amazon S3

1. Use la extensión `aws_s3` con el clúster de bases de datos de Babelfish.

   Cuando lo haga, asegúrese de hacer referencia a las tablas, tal y como existen en el contexto de PostgreSQL. Es decir, si quiere importar a una tabla de Babelfish llamada `[database].[schema].[tableA]`, haga referencia a esa tabla como `database_schema_tableA` en la función `aws_s3`:

   - Para ver un ejemplo del uso de una función `aws_s3` para importar datos, consulte Importación de datos de Amazon S3 a un clúster de base de datos Aurora PostgreSQL.
   - Para ver ejemplos del uso de las funciones de `aws_s3` para exportar datos, consulte Exportación de datos de consulta mediante la función aws_s3.query_export_to_s3.

2. Asegúrese de hacer referencia a las tablas de Babelfish mediante la nomenclatura de PostgreSQL cuando utilice la extensión `aws_s3` y Simple Storage Service (Amazon S3), como se muestra en la siguiente tabla.

Tabla de Babelfish	Tabla de Aurora PostgreSQL
database.*schema*.*table*	*database*_*schema*_*table*

Para obtener más información sobre el uso de Simple Storage Service (Amazon S3) con Aurora PostgreSQL, consulte Importación de datos de Amazon S3 en un clúster de base de datos Aurora PostgreSQL y Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3.

Uso de Babelfish con AWS Lambda

Después de cargar la extensión `aws_lambda` en el clúster de bases de datos de Babelfish, pero antes de poder invocar las funciones de Lambda, debe dar acceso a Lambda al clúster de bases de datos mediante este procedimiento.

Para configurar el acceso al clúster de la base de datos de Babelfish para trabajar con Lambda

Este procedimiento usa la AWS CLI para crear el rol y la política de IAM, así como para asociarlos al clúster de base de datos de Babelfish.

1. Cree una política de IAM que permita el acceso a Lambda desde el clúster de bases de datos de Babelfish.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
 "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

2. Cree un rol de IAM que la política pueda asumir en tiempo de ejecución.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Asocie la política de al rol.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Adjunte el rol al clúster de base de datos de Babelfish.

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Después de completar estas tareas, puede invocar las funciones de Lambda. Para obtener más información y ejemplos de configuración de AWS Lambda para el clúster de bases de datos de Aurora PostgreSQL con AWS Lambda, consulte Paso 2: configure IAM para su clúster de base de datos de Aurora PostgreSQL y AWS Lambda..

Para invocar una función Lambda desde el clúster de bases de datos de Babelfish

AWS Lambda admite funciones escritas en Java, Node.js, Python, Ruby y otros lenguajes. Si la función devuelve texto cuando se invoca, puede invocarla desde el clúster de bases de datos de Babelfish. El siguiente ejemplo es una función Python de marcador de posición que muestra un saludo.

```
lambda_function.py
import json
def lambda_handler(event, context):
    #TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
```

Actualmente, Babelfish no es compatible con JSON. Si la función devuelve JSON, debe usar una capa para gestionar el resultado JSON. Por ejemplo, digamos que `lambda_function.py`, como hemos mostrado anteriormente, se almacena en Lambda como `my-function`.

1. Conéctese al clúster de bases de datos de Babelfish con el cliente `psql` (o el cliente pgAdmin). Para obtener más información, consulte Conexión al clúster de bases de datos mediante psql.
2. Cree la capa. En este ejemplo se usa el lenguaje procedimental de PostgreSQL para SQL, `PL/pgSQL`. Para obtener más información, consulte PL/pgSQL–SQL Procedural Language (Lenguaje procedimental PL/pgSQL-SQL).

   ```
   create or replace function master_dbo.lambda_wrapper()
   returns text
   language plpgsql
   as
   $$
   declare
      r_status_code integer;
      r_payload text;
   begin
      SELECT payload INTO r_payload
        FROM aws_lambda.invoke(  aws_commons.create_lambda_function_arn('my-function',
    'us-east-1')
                               ,'{"body": "Hello from Postgres!"}'::json );
      return r_payload ;
   end;
   $$;
   ```

   La función ahora se puede ejecutar desde el puerto TDS de Babelfish (1433) o desde el puerto de PostgreSQL (5433).

   a. Para invocar (llamar) esta función desde el puerto de PostgreSQL:

```
SELECT * from aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function', 'us-east-1'), '{"body": "Hello from Postgres!"}'::json );
```

El resultado es similar al siguiente:

```
status_code |                        payload                         |
 executed_version | log_result
-------------+--------------------------------------------------------
+------------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
         |
(1 row)
```

b. Para invocar (llamar) esta función desde el puerto TDS, conéctese al puerto con el cliente de línea de comandos `sqlcmd` de SQL Server. Para obtener más información, consulte Uso de un cliente de SQL Server para conectarse al clúster de su base de datos. Cuando se haya conectado, ejecute lo siguiente:

```
1> select lambda_wrapper();
2> go
```

El comando devuelve un resultado similar al siguiente:

```
{"statusCode": 200, "body": "\"Hello from Lambda!\""}
```

Para obtener más información sobre el uso de Lambda con Aurora PostgreSQL, consulte Invocación de una función de AWS Lambda desde un clúster de base de datos de Aurora PostgreSQL.
Para obtener más información acerca de cómo trabajar con las funciones de Lambda, consulte Introducción a Lambda en la Guía para desarrolladores de AWS Lambda.

Uso de pg_stat_statements en Babelfish

Babelfish para Aurora PostgreSQL admite la extensión `pg_stat_statements` desde la versión 3.3.0 Para obtener más información, consulte pg_stat_statements.

Para obtener más información sobre la versión de esta extensión que admite Aurora PostgreSQL, consulte las versiones de extensiones.

Creación de la extensión pg_stat_statements

Para activar `pg_stat_statements`, debes activar el cálculo del identificador de consulta. Esto se hace automáticamente si `compute_query_id` está establecido en `on` o `auto` en el grupo de parámetros. El valor predeterminado del parámetro `compute_query_id` es `auto`. También debe crear esta extensión para activar esta función. Utilice el siguiente comando para instalar la extensión desde el punto de conexión de T-SQL:

```
1>EXEC sp_execute_postgresql 'CREATE EXTENSION pg_stat_statements WITH SCHEMA sys';
```

Puede acceder a las estadísticas de la consulta de la siguiente forma:

```
postgres=>select * from pg_stat_statements;
```

> **Note**
>
> Durante la instalación, si no proporciona el nombre de esquema de la extensión, de forma predeterminada, la creará en un esquema público. Para acceder a él, debe usar corchetes con el calificador de esquema, como se muestra a continuación:

```
postgres=>select * from [public].pg_stat_statements;
```

También puede crear la extensión desde el punto de conexión de PSQL.

Autorización de la extensión

De forma predeterminada, puede ver las estadísticas de las consultas realizadas en su base de datos de T-SQL sin necesidad de autorización alguna.

Para acceder a las estadísticas de consultas creadas por otros usuarios, debe tener el rol de `pg_read_all_stats` de PostgreSQL. Siga los pasos que se mencionan a continuación para crear el comando GRANT pg_read_all_stats.

1. En T-SQL, utilice la siguiente consulta que devuelve el nombre del rol de PG interno.

```
SELECT rolname FROM pg_roles WHERE oid = USER_ID();
```

2. Conéctese a la base de datos Babelfish para Aurora PostgreSQL con privilegios de rds_superuser y utilice el siguiente comando:

```
GRANT pg_read_all_stats TO <rolname_from_above_query>
```

Ejemplo

Desde el punto de conexión de T-SQL:

```
1>SELECT rolname FROM pg_roles WHERE oid = USER_ID();
2>go
```

```
rolname
-------
master_dbo
(1 rows affected)
```

Desde el punto de conexión de PSQL:

```
babelfish_db=# grant pg_read_all_stats to master_dbo;
```

```
GRANT ROLE
```

Puede acceder a las estadísticas de la consulta mediante la vista pg_stat_statements:

```
1>create table t1(cola int);
2>go
1>insert into t1 values (1),(2),(3);
2>go
```

```
(3 rows affected)
```

```
1>select userid, dbid, queryid, query from pg_stat_statements;
```

```
2>go
```

```
userid dbid queryid             query
------ ---- -------             -----
37503 34582 6487973085327558478 select * from t1
37503 34582 6284378402749466286 SET QUOTED_IDENTIFIER OFF
37503 34582 2864302298511657420 insert into t1 values ($1),($2),($3)
10    34582 NULL                <insufficient privilege>
37503 34582 5615368793313871642 SET TEXTSIZE 4096
37503 34582 639400815330803392  create table t1(cola int)
(6 rows affected)
```

Restablecer las estadísticas de las consultas

Puede utilizar `pg_stat_statements_reset()` para restablecer las estadísticas recopiladas hasta ahora por pg_stat_statements. Para obtener más información, consulte pg_stat_statements. Actualmente, solo se admite a través del punto de conexión de PSQL. Conéctese a la base de datos Babelfish para Aurora PostgreSQL con privilegios de `rds_superuser` y utilice el siguiente comando:

```
SELECT pg_stat_statements_reset();
```

Limitaciones

- Actualmente, no se admite `pg_stat_statements()` a través del punto de conexión de T-SQL. Se recomienda usar la vista `pg_stat_statements` para recopilar las estadísticas.
- Es posible que algunas de las consultas las reescriba el analizador de T-SQL implementado por el motor de Aurora PostgreSQL; la vista `pg_stat_statements` mostrará la consulta reescrita en lugar de la original.

  Ejemplo

  ```
  select next value for [dbo].[newCounter];
  ```

  La consulta anterior se reescribe de la siguiente manera en la vista pg_stat_statements.

```
select nextval($1);
```

- Según el flujo de ejecución de las instrucciones, es posible que pg_stat_statements no realice el seguimiento de algunas consultas y que no se muestren en la vista. Esta política incluye las siguientes instrucciones: `use dbname`, `goto`, `print`, `raise error`, `set`, `throw`, `declare cursor`.
- En las instrucciones CREATE LOGIN y ALTER LOGIN, no se mostrarán los parámetros query ni queryid. Mostrará privilegios insuficientes.
- La vista `pg_stat_statements` siempre contiene las dos entradas siguientes, ya que el cliente `sqlcmd` las ejecuta internamente.
  - DESACTIVAR QUOTED_IDENTIFIER
  - ESTABLECER TEXTSIZE 4096

Uso de pgvector en Babelfish

pgvector, una extensión de código abierto, le permite buscar datos similares directamente en la base de datos de Postgres. Babelfish ahora admite esta extensión a partir de las versiones 15.6 y 16.2. Para obtener más información, consulte pgvector Open source Documentation.

Requisitos previos

Para habilitar la funcionalidad pgvector, instale la extensión en el esquema sys mediante uno de los siguientes métodos:

- Ejecute el siguiente comando en el cliente sqlcmd:

```
exec sys.sp_execute_postgresql 'CREATE EXTENSION vector WITH SCHEMA sys';
```

- Conéctese a `babelfish_db` y ejecute el siguiente comando en el cliente psql:

```
CREATE EXTENSION vector WITH SCHEMA sys;
```

> **Note**
>
> Tras instalar la extensión pgvector, el tipo de datos vectoriales solo estará disponible en las nuevas conexiones de bases de datos que establezca. Las conexiones existentes no reconocerán el nuevo tipo de datos.

Funcionalidad admitida

Babelfish amplía la funcionalidad de T-SQL para admitir lo siguiente:

- Almacenamiento

  Babelfish ahora admite una sintaxis compatible con tipos de datos vectoriales, lo que mejora su compatibilidad con T-SQL. Para obtener más información sobre el almacenamiento de datos con pgvector, consulte Storing.
- Consultas

  Babelfish amplía la compatibilidad de expresiones de T-SQL para incluir operadores de similitud vectorial. Sin embargo, para todas las demás consultas, sigue siendo necesaria la sintaxis T-SQL estándar.

  > **Note**
  >
  > T-SQL no admite el tipo Array y los controladores de base de datos no tienen ninguna interfaz para gestionarlos. Como solución alternativa, Babelfish utiliza cadenas de texto (varchar/nvarchar) para almacenar datos vectoriales. Por ejemplo, cuando solicita un valor vectorial [1,2,3], Babelfish devolverá una cadena “[1,2,3]” como respuesta. Puede analizar y dividir esta cadena en la aplicación según sus necesidades.

  Para obtener más información sobre la consulta de datos con pgvector, consulte Querying.
- Indexación

  `Create Index` de T-SQL ahora admite la sintaxis `USING INDEX_METHOD`. Ahora puede definir un operador de búsqueda por similitud para usarlo en una columna específica al crear un índice.

  La gramática también se ha ampliado para admitir las operaciones de similitud vectorial en la columna requerida (consulte la gramática column_name_list_with_order_for_vector).

```
CREATE [UNIQUE] [clustered] [COLUMNSTORE] INDEX <index_name> ON <table_name> [USING
 vector_index_method] (<column_name_list_with_order_for_vector>)
Where column_name_list_with_order_for_vector is:
    <column_name> [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS | VECTOR_L2_OPS]
 (COMMA simple_column_name [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS |
 VECTOR_L2_OPS])
```

Para obtener más información sobre la indexación de datos con pgvector, consulte Indexing.

- Desempeño
  - Utilice `SET BABELFISH_STATISTICS PROFILE ON` para depurar los planes de consultas desde el punto de conexión de T-SQL.
  - Aumente `max_parallel_workers_get_gather` con la función `set_config` admitida en T-SQL.
  - Utilice `IVFFlat` para búsquedas aproximadas. Para obtener más información, consulte IVFFlat.

  Para mejorar el rendimiento con pgvector, consulte Performance.

Limitaciones

- Babelfish no admite la búsqueda de texto completo para la búsqueda híbrida. Para obtener más información, consulte Hybrid Search.
- Babelfish no admite actualmente la función de volver a indexar. Sin embargo, sigue teniendo la posibilidad de utilizar el punto de conexión de PostgreSQL para volver a indexar. Para obtener más información, consulte Vacuuming.

Uso de machine learning de Amazon Aurora con Babelfish

Puede ampliar las capacidades de su clúster de base de datos de Babelfish para Aurora PostgreSQL integrándolo con machine learning de Amazon Aurora. Esta integración perfecta le otorga acceso a una gama de servicios eficientes como Amazon Comprehend, IA de Amazon SageMaker o Amazon Bedrock, cada uno diseñado para abordar las distintas necesidades de machine learning.

Como usuario de Babelfish, puede utilizar los conocimientos existentes sobre la sintaxis y la semántica de T-SQL cuando trabaje con machine learning de Aurora. Siga las instrucciones que se proporcionan en la documentación de AWS de Aurora PostgreSQL. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora PostgreSQL.

Requisitos previos

- Antes de intentar configurar el clúster de base de datos de Babelfish para Aurora PostgreSQL para usar machine learning de Aurora, asegúrese de comprender los siguientes requisitos y requisitos previos relacionados. Para obtener más información, consulte Requisitos para usar machine learning de Aurora con Aurora PostgreSQL.
- Asegúrese de instalar la extensión `aws_ml` mediante el punto de conexión de Postgres o el procedimiento de almacenamiento `sp_execute_postgresql`.

```
exec sys.sp_execute_postgresql 'Create Extension aws_ml'
```

> **Note**
>
> Actualmente, Babelfish no admite operaciones en cascada con `sp_execute_postgresql` en Babelfish. Como `aws_ml` se basa en `aws_commons`, deberá instalarlo por separado mediante el punto de conexión de Postgres.

```
create extension aws_common;
```

Tratamiento de la sintaxis y la semántica de T-SQL con funciones **`aws_ml`**

En los siguientes ejemplos se explica cómo se aplican la sintaxis y la semántica de T-SQL a los servicios de Amazon ML:

Example : aws_bedrock.invoke_model: una consulta sencilla utilizando las funciones de Amazon Bedrock

```
aws_bedrock.invoke_model(
   model_id      varchar,
   content_type  text,
   accept_type   text,
   model_input   text)
Returns Varchar(MAX)
```

El siguiente ejemplo muestra cómo invocar un modelo de Anthropic Claude 2 para Bedrock mediante invoke_model.

```
SELECT aws_bedrock.invoke_model (
    'anthropic.claude-v2', -- model_id
    'application/json', -- content_type
    'application/json', -- accept_type
    '{"prompt": "\n\nHuman:
    You are a helpful assistant that answers questions directly
    and only using the information provided in the context below.
    \nDescribe the answerin detail.\n\nContext: %s \n\nQuestion:
    %s \n\nAssistant:","max_tokens_to_sample":4096,"temperature"
    :0.5,"top_k":250,"top_p":0.5,"stop_sequences":[]}' -- model_input
);
```

Example : aws_comprehend.detect_sentiment: una consulta sencilla utilizando las funciones de Amazon Comprehend

```
aws_comprehend.detect_sentiment(
   input_text varchar,
   language_code varchar,
   max_rows_per_batch int)
Returns table (sentiment varchar, confidence real)
```

En el siguiente ejemplo se muestra cómo invocar el servicio de Amazon Comprehend.

```
select sentiment from aws_comprehend.detect_sentiment('This is great', 'en');
```

Example : aws_sagemaker.invoke_endpoint: una consulta sencilla utilizando las funciones de Amazon SageMaker

```
aws_sagemaker.invoke_endpoint(
  endpoint_name varchar,
  max_rows_per_batch int,
  VARIADIC model_input "any") -- Babelfish inherits PG's variadic parameter type
Rerurns Varchar(MAX)
```

Puesto que model_input está marcado como VARIADIC y es del tipo “any”, los usuarios pueden pasar una lista de cualquier longitud y tipo de datos a la función, que actuará como entrada o entrada para el modelo. En el siguiente ejemplo se muestra cómo invocar el servicio de Amazon SageMaker.

```
SELECT CAST (aws_sagemaker.invoke_endpoint(
    'sagemaker_model_endpoint_name',
    NULL,
    arg1, arg2 -- model inputs are separate arguments )
AS INT) -- cast the output to INT
```

Para obtener información detallada sobre el uso de machine learning de Aurora con Aurora PostgreSQL, consulte Uso de machine learning de Amazon Aurora con Aurora PostgreSQL.

Limitaciones

- Aunque Babelfish no permite la creación de matrices, sí puede tratar datos que representen matrices. Cuando se utilizan funciones como `aws_bedrock.invoke_model_get_embeddings` que devuelven matrices, los resultados se entregan como una cadena que contiene los elementos de la matriz.

Babelfish admite servidores enlazados

Babelfish para Aurora PostgreSQL admite servidores enlazados mediante el uso de la extensión `tds_fdw` de PostgreSQL en la versión 3.1.0. Para trabajar con servidores enlazados, debe instalar la extensión `tds_fdw`. Para obtener más información acerca de la extensión `tds_fdw`, consulte Uso de los contenedores de datos externos compatibles para Amazon Aurora PostgreSQL.

Instalación de la extensión **`tds_fdw`**

Puede instalar una extensión `tds_fdw` utilizando los métodos siguientes.

Uso de CREATE EXTENSION desde el punto de conexión de PostgreSQL

1. Conéctese a su instancia de base de datos de PostgreSQL en la base de datos de Babelfish en el puerto PostgreSQL. Utilice una cuenta que tenga el rol rds_superuser.

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=test --dbname=babelfish_db --password
   ```

2. Instale la extensión de `tds_fdw`. Este proceso de instalación se realiza una sola vez. No tiene que volver a instalarla cuando el clúster de base de datos se reinicie.

```
babelfish_db=> CREATE EXTENSION tds_fdw;
CREATE EXTENSION
```

Llamar al procedimiento almacenado de **`sp_execute_postgresql`** desde un punto de conexión TDS

Babelfish admite la instalación de la extensión `tds_fdw` llamando al procedimiento `sp_execute_postgresql` desde la versión 3.3.0. Puede ejecutar instrucciones PostgreSQL desde el punto de conexión de T-SQL sin salir del puerto T-SQL. Para obtener más información, consulte Uso de los procedimientos de Babelfish para Aurora PostgreSQL

1. Conéctese a su instancia de base de datos de PostgreSQL en la base de datos de Babelfish en el puerto T-SQL.

```
sqlcmd -S your-DB-instance.aws-region.rds.amazonaws.com -U test -P password
```

2. Instale la extensión de `tds_fdw`.

```
1>EXEC sp_execute_postgresql N'CREATE EXTENSION tds_fdw';
2>go
```

Funcionalidades compatibles

Babelfish permite añadir puntos de conexión remotos de RDS para SQL Server o Babelfish para Aurora PostgreSQL como servidor enlazado. También puede añadir otras instancias remotas de SQL Server como servidores enlazados. A continuación, utilice `OPENQUERY( )` para recuperar datos de estos servidores enlazados. A partir de la versión 3.2.0 de Babelfish, también se admiten nombres de cuatro partes.

Se admiten los siguientes procedimientos almacenados y vistas de catálogo para utilizar los servidores enlazados.

Procedimientos almacenados

- sp_addlinkedserver: Babelfish no admite el parámetro `@provstr`.

- sp_addlinkedsrvlogin
  - Debe proporcionar un nombre de usuario y una contraseña remotos explícitos para conectarse al origen de datos remoto. No puede conectarse con las credenciales propias del usuario. Babelfish solo admite `@useself = false`.
  - Babelfish no admite el parámetro `@locallogin`, ya que no se permite la configuración del acceso al servidor remoto específico para el inicio de sesión local.
- sp_linkedservers
- sp_helplinkedsrvlogin
- sp_dropserver
- sp_droplinkedsrvlogin: Babelfish no admite el parámetro `@locallogin`, ya que no se permite la configuración del acceso al servidor remoto específico para el inicio de sesión local.
- sp_serveroption: Babelfish admite las siguientes opciones de servidor:
  - tiempo de espera de consulta (de la versión 3.2.0 de Babelfish)
  - tiempo de espera de conexión (de la versión 3.3.0 de Babelfish)
- sp_testlinkedserver (de la versión 3.3.0 de Babelfish)
- sp_enum_oledb_providers (de la versión 3.3.0 de Babelfish)

Vistas de catálogo

- sys.servers
- sys.linked_logins

Uso de cifrado en tránsito para la conexión

La conexión desde el servidor Babelfish para Aurora PostgreSQL de origen al servidor remoto de destino utiliza crifrado en tránsito (TLS/SSL), dependiendo de la configuración de la base de datos del servidor remoto. Si el servidor remoto no está configurado para el cifrado, el servidor Babelfish que realiza la solicitud a la base de datos remota se revierte a no cifrado.

Para aplicar el cifrado en las conexiones

- Si el servidor enlazado de destino es una instancia de RDS para SQL Server, configure `rds.force_ssl = on` para la instancia de SQL Server de destino. Para obtener más información sobre la configuración de SSL/TLS para RDS para SQL Server, consulte Uso de SSL con una instancia de base de datos de Microsoft SQL Server.

- Si el servidor enlazado de destino es un clúster de Babelfish para Aurora PostgreSQL, configure `babelfishpg_tds.tds_ssl_encrypt = on` y `ssl = on` para el servidor de destino. Para obtener más información acerca de SSL/TLS, consulte Configuración SSL de Babelfish y conexiones de cliente.

Adición de Babelfish como servidor enlazado desde SQL Server

Babelfish para Aurora PostgreSQL se puede añadir como un servidor enlazado desde un SQL Server. En una base de datos de SQL Server, puede añadir Babelfish como servidor enlazado mediante el proveedor OLE DB de Microsoft para ODBC: MSDASQL.

Hay dos maneras de configurar Babelfish como un servidor enlazado desde SQL Server mediante el proveedor MSDASQL:

- Proporcionar la cadena de conexión de ODBC como cadena del proveedor.
- Proporcionar el DSN del sistema del origen de datos de ODBC al agregar el servidor enlazado.

Limitaciones

- OPENQUERY() solo funciona para SELECT y no para DML.
- Los nombres de objetos de cuatro partes solo sirven para leer y no para modificar la tabla remota. Un UPDATE puede hacer referencia a una tabla remota de la cláusula FROM sin modificarla.
- No se admite la ejecución de procedimientos almacenados en servidores enlazados de Babelfish.
- Es posible que la actualización de la versión principal de Babelfish no funcione si hay objetos dependientes en `OPENQUERY()` u objetos a los que se hace referencia mediante nombres de cuatro partes. Debe asegurarse de que todos los objetos que hagan referencia a `OPENQUERY()` o nombre de cuatro partes se eliminen antes de realizar una actualización de la versión principal.
- Los siguientes tipos de datos no funcionan como se esperaba en el servidor remoto de Babelfish: `nvarchar(max)`, `varchar(max)`, `varbinary(max)`, `binary(max)` y `time`. Recomendamos utilizar la función CAST para convertirlos en tipos de datos compatibles.

Ejemplo

En el siguiente ejemplo, una instancia de Babelfish para Aurora PostgreSQL se conecta a una instancia de RDS para SQL Server en la nube.

```
EXEC master.dbo.sp_addlinkedserver @server=N'rds_sqlserver', @srvproduct=N'',
 @provider=N'SQLNCLI', @datasrc=N'myserver.CB2XKFSFFMY7.US-WEST-2.RDS.AMAZONAWS.COM';
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'rds_sqlserver',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassw
```

Cuando el servidor enlazado esté implementado, podrá utilizar OPENQUERY() de T-SQL o una nomenclatura estándar de cuatro partes para hacer referencia a una tabla, vista u otros objetos compatibles en el servidor remoto:

```
SELECT * FROM OPENQUERY(rds_sqlserver, 'SELECT * FROM TestDB.dbo.t1');
SELECT * FROM rds_sqlserver.TestDB.dbo.t1;
```

Para eliminar el servidor enlazado y todos los inicios de sesión asociados:

```
EXEC master.dbo.sp_dropserver @server=N'rds_sqlserver', @droplogins=N'droplogins';
```

Resolución de problemas

Puede usar el mismo grupo de seguridad para los servidores de origen y remotos para permitir que se comuniquen entre sí. El grupo de seguridad solo debe permitir el tráfico entrante en el puerto TDS (1433 de forma predeterminada) y la IP de origen del grupo de seguridad se puede configurar como el propio ID del grupo de seguridad. Para obtener más información sobre cómo establecer las reglas para conectarse a una instancia desde otra instancia con el mismo grupo de seguridad, consulte Reglas para conectarse a instancias desde una instancia con el mismo grupo de seguridad.

Si el acceso no está configurado correctamente, aparecerá un mensaje de error similar al del siguiente ejemplo cuando intente consultar el servidor remoto.

```
TDS client library error: DB #: 20009, DB Msg: Unable to connect: server is unavailable
 or does not exist (mssql2019.aws-region.rds.amazonaws.com), OS #: 110, OS Msg:
 Connection timed out, Level: 9
```

Uso de la búsqueda de texto completo en Babelfish

A partir de la versión 4.0.0, Babelfish ofrece un soporte limitado para la búsqueda de texto completo (FTS). La FTS es una potente característica de las bases de datos relacionales que permite buscar e indexar datos con mucho texto de forma eficiente. Le permite realizar búsquedas de texto complejas y recuperar rápidamente los resultados relevantes. La FTS es particularmente valiosa para aplicaciones que gestionan grandes volúmenes de datos de texto, como los sistemas de gestión de contenido, las plataformas de comercio electrónico y los archivos de documentos.

Descripción de las características admitidas con la búsqueda de texto completo en Babelfish

Babelfish admite las siguientes funciones de búsqueda de texto completo:

- Cláusula CONTAINS:
  - Compatibilidad básica para la cláusula CONTAINS.

```
CONTAINS (
     {
        column_name
     }
     , '<contains_search_condition>'
)
```

> **Note**
>
> En la actualidad, solo se admite el idioma inglés.

  - Gestión y traducción integrales de las cadenas de búsqueda `simple_term`.
- Cláusula `FULLTEXT INDEX`:
  - Solo admite la instrucción `CREATE FULLTEXT INDEX ON table_name(column_name [...n]) KEY INDEX index_name`.
  - Admite la instrucción `DROP FULLTEXT INDEX` completa.

> **Note**
>
> Para volver a indexar el índice de texto completo, debe eliminarlo y crear uno nuevo en la misma columna.

- Caracteres especiales en la condición de búsqueda:

- Babelfish garantiza que las apariciones únicas de caracteres especiales en las cadenas de búsqueda se gestionen de forma eficaz.

  > **Note**
  >
  > Si bien Babelfish ahora identifica los caracteres especiales en las cadenas de búsqueda, es fundamental reconocer que los resultados obtenidos pueden variar en comparación con los obtenidos con T-SQL.

- Alias de tabla en column_name:
  - Gracias a la compatibilidad con los alias de tablas, los usuarios pueden crear consultas SQL más concisas y legibles para la búsqueda de texto completo.

Limitaciones en la búsqueda de texto completo en Babelfish

- Actualmente, Babelfish no admite las siguientes opciones para la cláusula `CONTAINS`.
  - No se admiten caracteres especiales ni idiomas distintos del inglés. Recibirá un mensaje de error genérico para los caracteres e idiomas no compatibles

    ```
    Full-text search conditions with special characters or languages other than English
     are not currently supported in Babelfish
    ```

  - Varias columnas como `column_list`
  - Atributo PROPERTY
  - `prefix_term`, `generation_term`, `generic_proximity_term`, `custom_proximity_term` y `weighted_term`
  - No se admiten operadores booleanos y recibirá el siguiente mensaje de error cuando los utilice:

    ```
    boolean operators not supported
    ```

  - No se admiten nombres de identificadores con puntos.
- Actualmente, Babelfish no admite las siguientes opciones para la cláusula `CREATE FULLTEXT INDEX`.
  - [TYPE COLUMN type_column_name]
  - [LANGUAGE language_term]

  - [STATISTICAL_SEMANTICS]
  - opciones de grupos de archivos de catálogo
  - con opciones
- No se admite la creación de un catálogo de texto completo. La creación de un índice de texto completo no requiere un catálogo de texto completo.
- `CREATE FULLTEXT INDEX` no admite nombres de identificadores con puntos.
- Babelfish no admite actualmente caracteres especiales consecutivos en las cadenas de búsqueda. Si se utilizan, recibirá el siguiente mensaje de error:

```
Consecutive special characters in the full-text search condition are not currently
 supported in Babelfish
```

Babelfish admite tipos de datos geoespaciales

A partir de las versiones 3.5.0 y 4.1.0, Babelfish admite los dos tipos de datos espaciales siguientes:

- Tipo de datos geométricos: este tipo de datos está indicado para almacenar datos planares o euclidianos (tierra plana).
- Tipo de datos geográficos: este tipo de datos está diseñado para almacenar datos elipsoidales o de tierra redonda, como las coordenadas de latitud y longitud de GPS.

Estos tipos de datos permiten el almacenamiento y la manipulación de datos espaciales, pero con limitaciones.

Descripción de los tipos de datos geoespaciales en Babelfish

- Los tipos de datos geoespaciales se admiten en varios objetos de bases de datos, como vistas, procedimientos y tablas.

- Admite el tipo de datos de puntos bidimensionales para almacenar los datos de ubicación como puntos definidos por la latitud, la longitud y un identificador del sistema de referencia espacial (SRID) válido.
- Las aplicaciones que se conectan a Babelfish a través de controladores como JDBC, ODBC, DOTNET y PYTHON pueden utilizar esta característica geoespacial.

Funciones de tipos de datos geométricos admitidos en Babelfish

- STGeomFromText (***geometry_tagged_text***, SRID): crea una instancia de geometría con la representación Well-Known Text (WKT).
- STPointFromText (***point_tagged_text***, SRID): crea una instancia de punto con la representación WKT.
- Point (X, Y, SRID): crea una instancia de punto con valores flotantes de las coordenadas x e y.
- <geometry_instance>.STAsText ( ): extrae la representación WKT de la instancia de geometría.
- <geometry_instance>.STDistance (other_geometry): calcula la distancia entre dos instancias de geometría.
- <geometry_instance>.STX: extrae la coordenda X (longitud) para la instancia de geometría.
- <geometry_instance>.STY: extrae la coordenada Y (latitud) para la instancia de geometría.

Funciones de tipo de datos geográficos admitidos en Babelfish

- STGeomFromText (***geometry_tagged_text***, SRID): crea una instancia de geometría con la representación WKT.
- STPointFromText (***point_tagged_text***, SRID): crea una instancia de punto con la representación WKT.
- Point (Lat, Long, SRID): crea una instancia de punto utilizando valores flotantes de latitud y longitud.
- <geography_instance>.STAsText ( ): extrae la representación WKT de la instancia de geografía.
- <geometry_instance>.STDistance (other_geometry): calcula la distancia entre dos instancias de geografía.
- <geography_instance>.Lat: extrae el valor de latitud para la instancia de geografía.
- <geography_instance>.Long: extrae el valor de longitud para la instancia de geografía.

Limitaciones de Babelfish para tipos de datos geoespaciales

- Actualmente, Babelfish no admite características más avanzadas, como los marcadores Z-M para instancias de punto de tipos de datos geoespaciales.
- Actualmente no se admiten otros tipos de geometría distintos de las instancias de punto:
  - LineString
  - CircularString

  - CompoundCurve
  - Polygon
  - CurvePolygon
  - MultiPoint
  - MultiLineString
  - MultiPolygon
  - GeometryCollection
- Actualmente, la indexación espacial no se admite para los tipos de datos geoespaciales.
- Para estos tipos de datos solo se admiten actualmente las funciones enumeradas. Para obtener más información, consulte Funciones de tipos de datos geométricos admitidos en Babelfish y Funciones de tipo de datos geográficos admitidos en Babelfish.
- El resultado de la función StDistance para datos geográficos podría tener pequeñas variaciones de precisión en comparación con T-SQL. Esto se debe a la implementación subyacente de PostGIS. Para obtener más información, consulte ST_Distance
- Para lograr un rendimiento óptimo, utilice los tipos de datos geoespaciales integrados, sin crear capas adicionales de abstracción en Babelfish.

> **Tip**
>
> Aunque puede crear tipos de datos personalizados, no se recomienda crearlos sobre datos geoespaciales. Esto podría introducir complejidades y provocar un comportamiento inesperado, debido a la compatibilidad limitada.

- En Babelfish, los nombres de las funciones geoespaciales se utilizan como palabras clave y solo realizarán operaciones espaciales si se utilizan de la forma prevista.

> **Tip**
>
> Al crear funciones y procedimientos definidos por el usuario en Babelfish, evite utilizar los mismos nombres que las funciones geoespaciales integradas. Si tiene algún objeto de base de datos existente con el mismo nombre, utilice `sp_rename` para cambiarle el nombre.

Descripción de las particiones en Babelfish

A partir de la versión 4.3.0, Babelfish introduce la partición de tablas e índices con una compatibilidad limitada. En las siguientes secciones, se proporciona información sobre la creación de funciones de partición, la definición de esquemas de partición y la implementación de tablas e índices particionados en Babelfish.

Temas

- Introducción a las particiones en Babelfish
- Limitaciones y soluciones provisionales

Introducción a las particiones en Babelfish

- Funciones de partición:
  - `CREATE PARTITION FUNCTION:` define cómo se particiona una tabla o un índice especificando el tipo de columna de partición y el rango de valores de cada partición.
  - `DROP PARTITION FUNCTION:` elimina una función de partición existente.
- Esquemas de partición:
  - `CREATE PARTITION SCHEME:` define la asignación entre particiones y grupos de archivos.

    > **Note**
    >
    > En Babelfish, los grupos de archivos se tratan como objetos ficticios y no representan ubicaciones físicas de almacenamiento.

  - `DROP PARTITION SCHEME:` elimina un esquema de partición existente.
- Función del sistema:
  - `$PARTITION:` es una función del sistema que devuelve el número de partición al que pertenecería un valor especificado en una columna de particiones de una determinada tabla con particiones.
- Tablas e índices con particiones:
  - `CREATE TABLE ... ON partition_scheme_name (partition_column_name):` crea una tabla con particiones basada en un esquema de particiones y una columna de particiones específicos.

  - `CREATE INDEX ... ON partition_scheme_name (partition_column_name)`: crea un índice con particiones basado en un esquema de particiones y una columna de particiones específicos.
- Vistas del sistema para la partición de metadatos:

  Se han añadido las siguientes vistas del sistema para proporcionar metadatos relacionados con la partición:

  - `sys.destination_data_spaces`
  - `sys.partitions`
  - `sys.partition_functions`
  - `sys.partition_parameters`
  - `sys.partition_range_values`
  - `sys.partition_schemes`

Limitaciones y soluciones provisionales

Babelfish aún no admite las siguientes capacidades de particionamiento de SQL Server:

- `ALTER PARTITION FUNCTION` y `ALTER PARTITION SCHEME`.

  > **Note**
  >
  > Babelfish no es compatible con las operaciones de división y fusión. Defina todas las particiones en las funciones de partición durante la creación, ya que no podrá agregar ni eliminar particiones posteriormente.

- Columnas calculadas como columnas de particionamiento.
- Utilidad de `INSERT BULK` y `BCP` para tablas particionadas.
- Opción de límite `LEFT` para funciones de partición.
- Tipo de datos `SQL_VARIANT` para las funciones de partición.
- `TRUNCATE TABLE ... WITH PARTITION`.
- `ALTER TABLE ... SWITCH PARTITION`.
- Índices con particiones no alineados, como, por ejemplo, el esquema de particiones y la columna de particiones que difieren de la tabla con particiones.

- La migración de DMS desde el origen de Babelfish solo se admite para tareas de carga completa en tablas con particiones.
- Uso de la intercalación en la función de partición.
- Uso de una columna de particionamiento con una intercalación distinta de la intercalación predeterminada de la base de datos.
- Babelfish no admite estas opciones de sintaxis, pero ofrece soluciones alternativas:
  - Uso del esquema de particiones con restricciones o índices en la instrucción CREATE TABLE.
  - ALTER TABLE... ADD CONSTRAINT... ON partition_scheme_name (partition_column_name).

> **Note**
>
> Puede configurar la escotilla de escape `babelfishpg_tsql.escape_hatch_storage_on_partition` para omitirlo. Esto permitirá que el analizador omita la opción de esquema de particiones utilizada con las restricciones o los índices, y el backend creará restricciones o índices individuales para cada partición.

Solución de problemas de Babelfish

A continuación, puede encontrar ideas de solución de problemas y soluciones para algunos problemas del clúster de base de datos de Babelfish.

Temas

- Error de conexión

Error de conexión

Entre las causas frecuentes de errores de conexión a un nuevo clúster de bases de datos de Aurora con Babelfish se incluyen las siguientes:

- El grupo de seguridad no permite el acceso: si tiene problemas para conectarse a una instancia de Babelfish, asegúrese de haber agregado su dirección IP al grupo de seguridad predeterminado de Amazon EC2. Puede usar https://checkip.amazonaws.com/ para determinar su dirección IP y, a continuación, agregarla a la regla de entrada para el puerto de TDS y el puerto de PostgreSQL. Para obtener más información, consulte Agregar reglas a un grupo de seguridad en la Guía del usuario de Amazon EC2.
- Configuraciones SSL incorrectas: si el parámetro `rds.force_ssl` está activado (establecido en 1) en Aurora PostgreSQL, entonces los clientes deben conectarse a Babelfish a través de SSL. Si el cliente no está configurado correctamente, verá un mensaje de error como el siguiente:

  ```
  Cannot connect to your-Babelfish-DB-cluster, 1433
  ---------------------
  ADDITIONAL INFORMATION:
  no pg_hba_conf entry for host "256.256.256.256", user "your-user-name",
  "database babelfish_db", SSL off (Microsoft SQL Server, Error: 33557097)
  ...
  ```

  Este error indica un posible problema de configuración SSL entre el cliente local y el clúster de bases de datos de Babelfish, y que el clúster requiere que los clientes utilicen SSL (`rds.force_ssl`se establece en 1). Para obtener más información acerca de la configuración de SSL, consulte Uso de SSL con una instancia de base de datos PostgreSQL en la guía del usuario de Amazon RDS.

  Si utiliza SQL Server Management Studio (SSMS) para conectarse a Babelfish y ve este error, puede elegir las opciones Cifrar conexión y Certificado de servidor de confianza en el panel

Propiedades de conexión e intentarlo de nuevo. Esta configuración gestiona el requisito de conexión SSL para SSMS.

Para obtener más información acerca de cómo solucionar problemas de conexión de Aurora, consulte No puede conectarse a la instancia de base de datos de Amazon RDS.

Desactivación de Babelfish

Cuando ya no necesite Babelfish, puede desactivar la funcionalidad de Babelfish.

Tenga en cuenta algunas consideraciones:

- En algunos casos, puede desactivar Babelfish antes de finalizar una migración a Aurora PostgreSQL. Si lo hace y su DDL depende de los tipos de datos de SQL Server o utiliza cualquier funcionalidad de T-SQL en el código, se producirá un error en el código.
- Si tras aprovisionar una instancia de Babelfish desactiva la extensión de Babelfish, no podrá aprovisionar la misma base de datos de nuevo en el mismo clúster.

Para desactivar Babelfish, modifique el grupo de parámetros y establezca `rds.babelfish_status` en `OFF`. Para poder seguir utilizando los tipos de datos de SQL Server con Babelfish desactivado, establezca `rds.babelfish_status` en `datatypesonly`.

Si desactiva Babelfish en el grupo de parámetros, todos los clústeres que utilizan ese grupo de parámetros pierden la funcionalidad de Babelfish.

Para obtener más información sobre cómo modificar los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora. Para obtener información acerca de los parámetros específicos de Babelfish, consulte Configuración del grupo de parámetros del clúster de base de datos para Babelfish.

Administración de las actualizaciones de versiones de Babelfish para Aurora PostgreSQL

Babelfish es una opción disponible para la versión 13.4 de Aurora PostgreSQL, así como también para versiones posteriores. Las actualizaciones de Babelfish están disponibles con ciertas versiones nuevas del motor de base de datos de Aurora PostgreSQL. Para obtener más información, consulte las Notas de la versión de Aurora PostgreSQL.

> **Note**
>
> Los clústeres de base de datos de Babelfish que funcionan con cualquier versión de Aurora PostgreSQL 13 no pueden actualizarse a Aurora PostgreSQL 14.3, 14.4 y 14.5. Además, Babelfish no permite actualizaciones directas desde 13.x a 15.x. Primero debe actualizar el clúster de base de datos 13.x a la versión 14.6 o una versión posterior y, a continuación, a la versión 15.x.

Para obtener una lista de las funcionalidades admitidas en las diferentes versiones de Babelfish, consulte Funcionalidades compatibles con Babelfish por versión.

Para ver una lista de las funciones no admitidas actualmente, consulte Funcionalidades no compatibles con Babelfish.

Para obtener una lista de las versiones de Aurora PostgreSQL compatibles con Babelfish, consulte la Región de AWS con el comando describe-db-engine-versions de la AWS CLI, tal como se muestra a continuación.

Para Linux, macOS o Unix:

```
$ aws rds describe-db-engine-versions --region us-east-1 \
    --engine aurora-postgresql \
    --query '*[]|[?SupportsBabelfish==`true`].[EngineVersion]' \
    --output text
```

Para obtener más información, consulte describe-db-engine-versions en la Referencia de los comandos de AWS CLI.

En los siguientes temas, puede aprender a identificar la versión de Babelfish que se ejecuta en su clúster de base de datos de Aurora PostgreSQL y a actualizarla a una nueva versión.

Contenido

- Identificación de la versión de Babelfish
- Actualización del clúster de base de datos a una nueva versión
  - Actualización de Babelfish a una nueva versión secundaria
  - Actualización de Babelfish a una nueva versión principal
    - Antes de actualizar Babelfish a una nueva versión principal
    - Realización de una actualización de la versión principal
    - Después de actualizar a una nueva versión principal
    - Ejemplo: Actualización del clúster de base de datos de Babelfish a una versión principal
- Uso del parámetro de versión del producto de Babelfish
  - Configuración del parámetro de versión del producto de Babelfish
  - Consultas y parámetros afectados
  - Interfaz con el parámetro babelfishpg_tsql.version

Identificación de la versión de Babelfish

Puede consultar Babelfish para encontrar detalles sobre la versión de Babelfish, la versión de Aurora PostgreSQL y la versión compatible de Microsoft SQL Server. Puede utilizar el puerto de TDS o de PostgreSQL.

- To use the TDS port to query for version information
- To use the PostgreSQL port to query for version information

Para utilizar el puerto TDS para consultar la información de versión

1. Use `sqlcmd` o `ssms` para conectarse al punto de conexión del clúster de base de datos de Babelfish.

   ```
   sqlcmd -S bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
       login-id -P password -d db_name
   ```

2. Para identificar la versión de Babelfish, ejecute la siguiente consulta:

   ```
   1> SELECT CAST(serverproperty('babelfishversion') AS VARCHAR)
   2> GO
   ```

La consulta devuelve resultados similares a los siguientes:

```
serverproperty
------------------------------
3.4.0

(1 rows affected)
```

3. Para identificar la versión del clúster de bases de datos de Aurora PostgreSQL, ejecute la siguiente consulta:

```
1> SELECT aurora_version() AS aurora_version
2> GO
```

La consulta devuelve resultados similares a los siguientes:

```
aurora_version


-----------------------------------------------
15.5.0

(1 rows affected)
```

4. Para identificar la versión compatible de Microsoft SQL Server, ejecute la siguiente consulta:

```
1> SELECT @@VERSION AS version
2> GO
```

La consulta devuelve resultados similares a los siguientes:

```
Babelfish for Aurora PostgreSQL with SQL Server Compatibility - 12.0.2000.8
Dec 7 2023 09:43:06
Copyright (c) Amazon Web Services
PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)

(1 rows affected)
```

Como ejemplo que muestra una pequeña diferencia entre Babelfish y Microsoft SQL Server, puede ejecutar la siguiente consulta. En Babelfish, la consulta devuelve 1, mientras que en Microsoft SQL Server, la consulta devuelveNULL.

```
SELECT CAST(serverproperty('babelfish') AS VARCHAR) AS runs_on_babelfish
```

También puede utilizar el puerto de PostgreSQL para obtener información sobre la versión, como se muestra en el siguiente procedimiento.

Para utilizar el puerto de PostgreSQL para consultar la información de versión

1. Use `psql` o `pgAdmin` para conectarse al punto de conexión del clúster de base de datos de Babelfish.

   ```
   psql host=bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com
        port=5432 dbname=babelfish_db user=sa
   ```

2. Active la característica extendida (`\x`) de `psql` para obtener una salida más legible.

   ```
   babelfish_db=> \x
   babelfish_db=> SELECT
   babelfish_db=> aurora_version() AS aurora_version,
   babelfish_db=> version() AS postgresql_version,
   babelfish_db=> sys.version() AS Babelfish_compatibility,
   babelfish_db=> sys.SERVERPROPERTY('BabelfishVersion') AS Babelfish_Version;
   ```

   La consulta devuelve una salida similar a la siguiente:

   ```
   -[ RECORD 1 ]-----------
   +-------------------------------------------------------------------------------------------
   aurora_version          | 15.5.0
   postgresql_version      | PostgreSQL 15.5 on x86_64-pc-linux-gnu, compiled by
    x86_64-pc-linux-gnu-gcc (GCC) 9.5.0, 64-bit
   babelfish_compatibility | Babelfish for Aurora Postgres with SQL Server
    Compatibility - 12.0.2000.8                    +
                           | Dec 7 2023 09:43:06
                                              +
                           | Copyright (c) Amazon Web Services
                                               +
                           | PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)
   ```

```
babelfish_version       | 3.4.0
```

Actualización del clúster de base de datos a una nueva versión

Las nuevas versiones de Babelfish están disponibles con algunas versiones nuevas del motor de base de datos Aurora PostgreSQL después de la versión 13.4. Cada nueva versión de Babelfish tiene su propio número de versión. Al igual que con Aurora PostgreSQL, Babelfish usa el esquema de nomenclatura *principal*.*secundaria*.*revisión* para las versiones. Por ejemplo, la primera versión de Babelfish, la versión 1.0.0, estuvo disponible como parte de Aurora PostgreSQL 13.4.0.

Babelfish no requiere un proceso de instalación independiente. Como se explica en Creación de un clúster de base de datos de Babelfish para Aurora PostgreSQL, Turn on Babelfish (Activar Babelfish) es una opción que se elige al crear un clúster de base de datos de Aurora PostgreSQL.

Del mismo modo, no puede actualizar Babelfish independientemente del clúster de base de datos de Aurora que es compatible. Para actualizar un clúster de base de datos de Babelfish para Aurora PostgreSQL existente a una nueva versión de Babelfish, actualice el clúster de base de datos de Aurora PostgreSQL a una nueva versión que sea compatible con la versión de Babelfish que desee utilizar. El procedimiento que siga para la actualización depende de la versión de Aurora PostgreSQL que sea compatible con la implementación de Babelfish, tal y como se indica a continuación.

Actualizaciones de la versión principal

Debe actualizar las siguientes versiones de Aurora PostgreSQL a Aurora PostgreSQL 14.6 o una versión posterior antes de actualizar a Aurora PostgreSQL 15.2.

- Aurora PostgreSQL 13.8 y todas las versiones posteriores
- Aurora PostgreSQL 13.7.1 y todas las versiones secundarias posteriores
- Aurora PostgreSQL 13.6.4 y todas las versiones secundarias posteriores

Puede actualizar Aurora PostgreSQL 14.6 y versiones posteriores a Aurora PostgreSQL 15.2 y versiones posteriores.

La actualización de un clúster de base de datos de Aurora PostgreSQL a una nueva versión principal implica varias tareas preliminares. Para obtener más información, consulte Actualización a una versión principal. Para actualizar correctamente su clúster de base de datos de Babelfish para Aurora PostgreSQL, debe crear un grupo de parámetros de clúster de base de datos personalizado para la nueva versión de Aurora PostgreSQL. Este nuevo grupo de parámetros debe contener la misma configuración de parámetros de Babelfish que la del clúster que está

actualizando. Para obtener más información y ver una tabla de las principales fuentes y destinos de las actualizaciones de las versiones principales, consulte Actualización de Babelfish a una nueva versión principal.

Actualizaciones y revisiones de versión secundarias

Las versiones secundarias y las revisiones no requieren la creación de un nuevo grupo de parámetros de clúster de base de datos para la actualización. Las versiones secundarias y las revisiones pueden utilizar el proceso de actualización de versiones secundarias, tanto automático como manual. Para obtener más información y ver una tabla de las fuentes y destinos de las versiones, consulte Actualización de Babelfish a una nueva versión secundaria.

> **Note**
>
> Antes de realizar una actualización principal o secundaria, aplique todas las tareas de mantenimiento pendientes a su clúster de Babelfish para Aurora PostgreSQL.

Temas

- Actualización de Babelfish a una nueva versión secundaria
- Actualización de Babelfish a una nueva versión principal

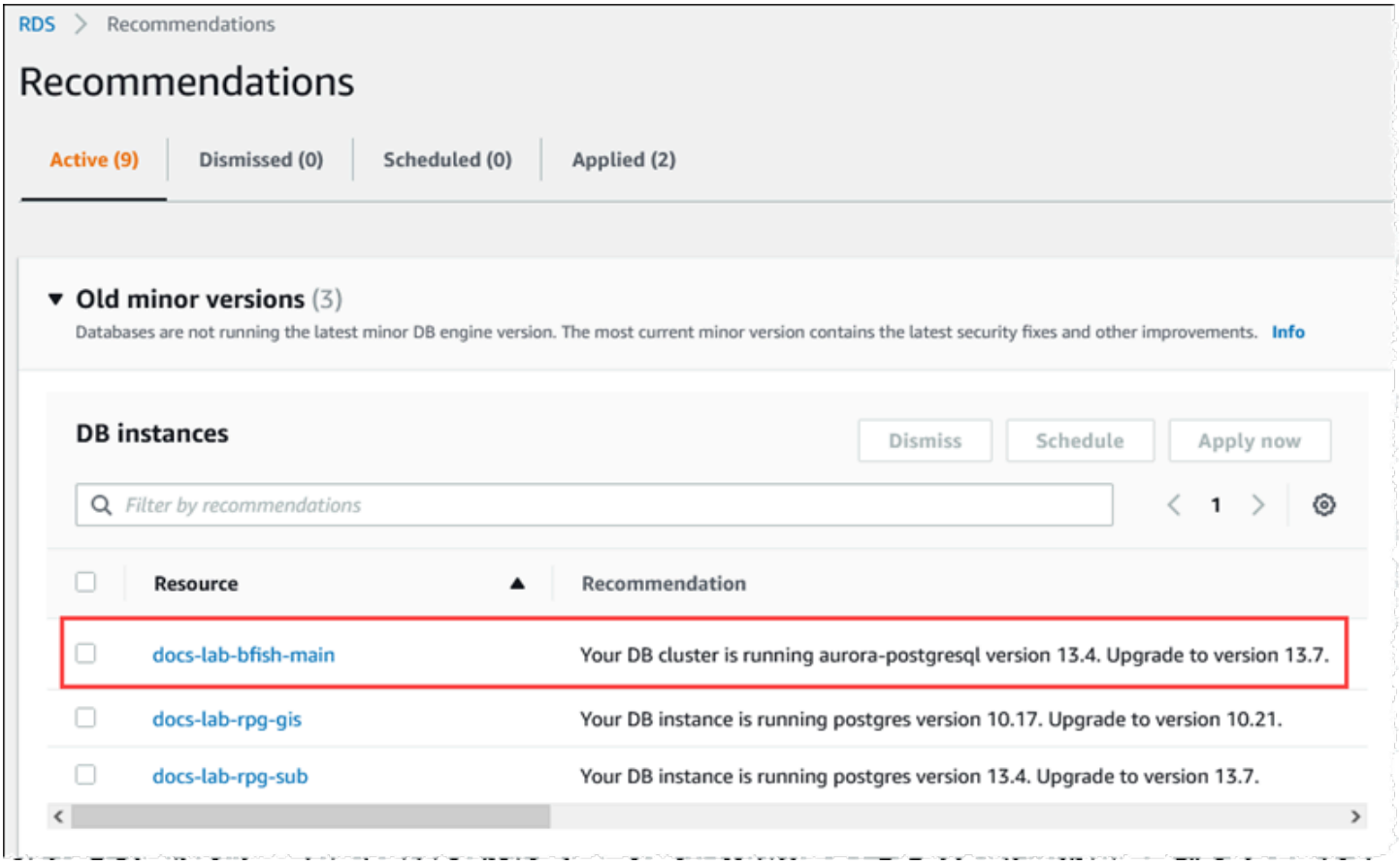
Actualización de Babelfish a una nueva versión secundaria

Una actualización menor de la versión tiene como objetivo mantener la compatibilidad con versiones anteriores. Sin embargo, en algunos casos, las correcciones de seguridad críticas o las correcciones de errores importantes pueden no ser totalmente compatibles con versiones anteriores. Una versión de revisión incluye correcciones importantes que se añaden a una versión secundaria después de su lanzamiento. Por ejemplo, la etiqueta de versión de la primera versión de Aurora PostgreSQL 13.4 era Aurora PostgreSQL 13.4.0. Hasta la fecha, se han publicado varias revisiones para esa versión secundaria, incluidas Aurora PostgreSQL 13.4.1, 13.4.2 y 13.4.4. Encontrará las revisiones disponibles para cada versión de Aurora PostgreSQL en la lista de versiones de revisiones que aparece en la parte superior de las notas de la versión de Aurora PostgreSQL. Para ver un ejemplo, consulte PostgreSQL 14.3 en las notas de la versión de Aurora PostgreSQL.

Si su clúster de base de datos de Aurora PostgreSQL está configurado con la opción Auto minor version upgrade (Actualización automática de versiones secundarias), su clúster de base de datos de Babelfish para Aurora PostgreSQL se actualiza automáticamente durante el período de

mantenimiento del clúster. Para obtener más información acerca de la actualización automática de versiones secundarias (AmVU) y cómo utilizarla, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora. Si su clúster no utiliza AmVU, puede actualizar manualmente su clúster de base de datos de Babelfish para Aurora PostgreSQL a las nuevas versiones secundarias, ya sea respondiendo a las tareas de mantenimiento o modificando el clúster para utilizar la nueva versión.

Al elegir una versión de Aurora PostgreSQL para instalar y al ver un clúster de base de datos de Aurora PostgreSQL existente en la Consola de administración de AWS, la versión muestra solo los dígitos *principal*.*secundaria*. Por ejemplo, la siguiente imagen de la consola de un clúster de base de datos de Babelfish para Aurora PostgreSQL existente con Aurora PostgreSQL 13.4 recomienda actualizar el clúster a la versión 13.7, una nueva versión secundaria de Aurora PostgreSQL.



Para obtener todos los detalles de la versión, incluido el nivel de *revisión*, puede consultar el clúster de base de datos de Aurora PostgreSQL mediante la función `aurora_version` de Aurora PostgreSQL. Para obtener más información, consulte aurora_version en la Referencia de las funciones de Aurora PostgreSQL. Encontrará un ejemplo del uso de la función en el procedimiento

To use the PostgreSQL port to query for version information en Identificación de la versión de Babelfish.

La siguiente tabla muestra las versiones de Aurora PostgreSQL y Babelfish y las versiones de destino disponibles que pueden admitir el proceso de actualización de la versión secundaria.

Versiones de origen actuales	Destinos de actualización más recientes
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
17.4 (5.1)	17.5 (5.2)
16.8 (4.5)	16.9 (4.6)
16.6 (4.4.0)	16.9 (4.6), 16.8 (4.5.0)
16.4 (4.3.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0)
16.3 (4.2.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
16.2 (4.1.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
16.1 (4.0.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.12 (3.9.0)	15.13 (3.10)
15.10 (3.8.0)	15.13 (3.10), 15.12 (3.9.0)
15.8 (3.7.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0)
15.7 (3.6.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
15.6 (3.5.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
15.5 (3.4.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)

Versiones de origen actuales	Destinos de actualización más recientes
15.4 (3.3.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
15.12 (3.9.0), 15.3 (3.2.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
15.2 (3.1.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.17 (2.12.0)	14.18 (2.13.0)
14.15 (2.11.0)	14.18 (2.13.0), 14.17 (2.12.0)
14.13 (2.10.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0)
14.12 (2.9.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0)
14.11 (2.8.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0)
14.10 (2.7.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0)
14.9 (2.6.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0)
14.8 (2.5.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0)
14.7 (2.4.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0)

Versiones de origen actuales	Destinos de actualización más recientes
14.6 (2.3.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0)
14.5 (2.2.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0), 14.6 (2.3.0)
14.3 (2.1.0)	14.18 (2.13.0), 14.6 (2.3.0)
13.8 (1.4.0)	13.9 (1.5.0)
13.7 (1.3.0)	13.9 (1.5.0), 13.8 (1.4.0)

Actualización de Babelfish a una nueva versión principal

Para actualizar una versión principal, primero debe actualizar su clúster de base de datos de Babelfish para Aurora PostgreSQL a una versión que sea compatible con la actualización de la versión principal. Para ello, aplique actualizaciones de revisiones o actualizaciones de versiones secundarias a su clúster de base de datos. Para obtener más información, consulte Actualización de Babelfish a una nueva versión secundaria.

La siguiente tabla muestra la versión de Aurora PostgreSQL y la versión de Babelfish que pueden ser compatibles con una actualización de la versión principal.

Versiones de origen actuales	Destinos de actualización más recientes
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
16.9 (4.6.0)	17.5 (5.2.0)
16.8 (4.5.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.6 (4.4.0)	17.5 (5.2.0), 17.4 (5.1.0)

Versiones de origen actuales	Destinos de actualización más recientes
16.4 (4.3.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.3 (4.2.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.2 (4.1.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.1 (4.0.0)	17.5 (5.2.0), 17.4 (5.1.0)
15.13 (3.10)	17.5 (5.2.0) 16.9 (4.6.0)
15.12 (3.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0)
15.10 (3.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0)
15.8 (3.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
15.7 (3.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
15.6 (3.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.5 (3.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)

Versiones de origen actuales	Destinos de actualización más recientes
15.4 (3.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.3 (3.2.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.2 (3.1.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
14.18 (2.13.0)	17.5 (5.2.0) 16.9 (4.6.0) 15.13 (3.10.0)
14.17 (2.12.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0) 15.13 (3.10.0), 15.12 (3.9.0)
14.15 (2.11.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0)
14.13 (2.10.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)

Versiones de origen actuales	Destinos de actualización más recientes
14.12 (2.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
14.11 (2.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
14.10 (2.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
14.9 (2.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)

Versiones de origen actuales	Destinos de actualización más recientes
14.8 (2.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.7 (2.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
14.6 (2.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
13.9 (1.5.0)	14.6 (2.3.0)
13.8 (1.4.0)	14.6 (2.3.0)
13.7 (1.3.0)	14.6 (2.3.0)

Antes de actualizar Babelfish a una nueva versión principal

Una actualización puede implicar breves interrupciones. Por este motivo, le recomendamos que realice o programe actualizaciones durante el período de mantenimiento o durante otros períodos de poca utilización.

Antes de realizar una actualización de la versión principal

1. Identifique la versión de Babelfish de su clúster de base de datos Aurora PostgreSQL existente mediante los comandos que se describen en Identificación de la versión de Babelfish. PostgreSQL gestiona la información de las versiones de Aurora PostgreSQL y Babelfish, así que siga los pasos detallados en el procedimiento To use the PostgreSQL port to query for version information para obtener los detalles.
2. Compruebe si su versión es compatible con la actualización de la versión principal. Para obtener la lista de versiones que son compatibles con la característica de actualización de versiones principales, consulte Actualización de Babelfish a una nueva versión secundaria y realice las tareas previas a la actualización necesarias.

   Por ejemplo, si su versión de Babelfish se ejecuta en un clúster de base de datos de Aurora PostgreSQL 13.5 y desea actualizar a Aurora PostgreSQL 15.2, aplique primero todas las versiones secundarias y revisiones para actualizar el clúster a Aurora PostgreSQL 14.6 o versiones posteriores. Cuando el clúster esté en la versión 14.6 o una versión posterior, continúe con el proceso de actualización de la versión principal.
3. Cree una instantánea manual de tu clúster de base de datos de Babelfish actual como copia de seguridad. La copia de seguridad le permite restaurar el clúster a su versión de Aurora PostgreSQL y Babelfish y restaurar todos los datos al estado anterior a la actualización. Para obtener más información, consulte Creación de una instantánea de clúster de base de datos. Asegúrese de conservar su grupo de parámetros de clúster de base de datos personalizado existente para volver a usarlo si decide restaurar este clúster a su estado previo a la actualización. Para obtener más información, consulte Restauración de una instantánea de clúster de base de datos y Consideraciones relativas al grupo de parámetros.
4. Prepare un grupo de parámetros de clúster de base de datos personalizado para la versión de base de datos de Aurora PostgreSQL de destino. Duplique los ajustes de los parámetros de Babelfish de su clúster de base de datos de Babelfish para Aurora PostgreSQL actual. Para obtener una lista de todos los parámetros de Babelfish, consulte Configuración del grupo de parámetros del clúster de base de datos para Babelfish. Para una actualización de versión importante, los siguientes parámetros requieren la misma configuración que el clúster de base de datos de origen. Para que la actualización se realice correctamente, todos los ajustes deben ser los mismos.

   - rds.babelfish_status
   - babelfishpg_tds.tds_default_numeric_precision

- babelfishpg_tds.tds_default_numeric_scale
- babelfishpg_tsql.database_name
- babelfishpg_tsql.default_locale
- babelfishpg_tsql.migration_mode
- babelfishpg_tsql.server_collation_name

> **Warning**
>
> Si los ajustes de los parámetros de Babelfish del grupo de parámetros del clúster de base de datos personalizado para la nueva versión de Aurora PostgreSQL no coincide con los valores de los parámetros del clúster que está actualizando, la operación `ModifyDBCluster` fallará. Aparece un mensaje de error `InvalidParameterCombination` en la Consola de administración de AWS o en el resultado del comando `modify-db-cluster` de la AWS CLI.

5. Utilice la Consola de administración de AWS o la AWS CLI para crear el grupo de parámetros personalizado del clúster de base de datos. Elija la familia de Aurora PostgreSQL aplicable a la versión de Aurora PostgreSQL que desea actualizar.

> **Tip**
>
> Los grupos de parámetros se administran a nivel de Región de AWS. Cuando trabaje con AWS CLI, puede configurarlos con una región predeterminada en lugar de especificarlos en el comando `--region`. Para obtener más información sobre la AWS CLI, consulte la sección Configuración rápida de la Guía de usuario de la AWS Command Line Interface.

Realización de una actualización de la versión principal

1. Actualice el clúster de base de datos de Aurora PostgreSQL a una nueva versión principal. Para obtener más información, consulte Actualización del motor de Aurora PostgreSQL a una nueva versión principal.
2. Reinicie la instancia de escritor del clúster para que la configuración de los parámetros se aplique.

Después de actualizar a una nueva versión principal

Tras una actualización de la versión principal a una nueva versión de Aurora PostgreSQL, el valor de `IDENTITY` de las tablas con una columna `IDENTITY` puede ser mayor (+32) que el valor anterior a la actualización. El resultado es que, cuando se inserta la siguiente fila en dichas tablas, el valor de la columna de identidad generado salta al número +32 y comienza la secuencia desde allí. Esta condición no afectará negativamente a las funciones del clúster de base de datos de Babelfish. Sin embargo, si lo desea, puede restablecer el objeto de secuencia en función del valor máximo de la columna. Para ello, conéctese al puerto T-SQL de la instancia de escritor de Babelfish mediante `sqlcmd` u otro cliente de SQL Server. Para obtener más información, consulte Uso de un cliente de SQL Server para conectarse al clúster de su base de datos.

```
sqlcmd -S bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
     sa -P ******** -d dbname
```

Cuando esté conectado, utilice el siguiente comando SQL para generar instrucciones que pueda utilizar para iniciar el objeto de secuencia asociado. Este comando SQL funciona tanto para configuraciones de Babelfish de una sola base de datos como de múltiples bases de datos. Para obtener más información acerca de estos dos modelos de implementación, consulte Uso de Babelfish con una base de datos única o varias bases de datos.

```
DECLARE @schema_prefix NVARCHAR(200) = ''
IF current_setting('babelfishpg_tsql.migration_mode') = 'multi-db'
    SET @schema_prefix = db_name() + '_'
SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
 schema_name(tables.schema_id)
    + '.' + tables.name + ''', ''' + columns.name + '''),(select max(' + columns.name +
 ')
    FROM ' + schema_name(tables.schema_id) + '.' + tables.name + '));
    'FROM sys.tables tables JOIN sys.columns
    columns ON tables.object_id = columns.object_id
    WHERE columns.is_identity = 1
GO
```

La consulta genera una serie de instrucciones SELECT que puede ejecutar luego para restablecer el valor máximo de IDENTITY y cerrar cualquier brecha. A continuación, se muestra el resultado cuando se utiliza la base de datos SQL Server de ejemplo, Northwind, que se ejecuta en un clúster de Babelfish.

```
-----------------------------------------------------
```

```
SELECT setval(pg_get_serial_sequence('northwind_dbo.categories', 'categoryid'),(select
 max(categoryid)
    FROM dbo.categories));

SELECT setval(pg_get_serial_sequence('northwind_dbo.orders', 'orderid'),(select
 max(orderid)
    FROM dbo.orders));


SELECT setval(pg_get_serial_sequence('northwind_dbo.products', 'productid'),(select
 max(productid)
    FROM dbo.products));


SELECT setval(pg_get_serial_sequence('northwind_dbo.shippers', 'shipperid'),(select
 max(shipperid)
    FROM dbo.shippers));


SELECT setval(pg_get_serial_sequence('northwind_dbo.suppliers', 'supplierid'),(select
 max(supplierid)
    FROM dbo.suppliers));


(5 rows affected)
```

Ejecute las instrucciones una por una para restablecer los valores de la secuencia.

Ejemplo: Actualización del clúster de base de datos de Babelfish a una versión principal

En este ejemplo, puede encontrar la serie de comandos de AWS CLI para explicar cómo se actualiza un clúster de base de datos de Aurora PostgreSQL 13.6.4 que ejecuta la versión 1.2.2 de Babelfish a Aurora PostgreSQL 14.6. En primer lugar, cree un grupo de parámetros de clúster de base de datos personalizado para Aurora PostgreSQL 14. A continuación, modifique los valores de los parámetros para que coincidan con los de su origen de Aurora PostgreSQL versión 13. Por último, realice la actualización modificando el clúster de origen. Para obtener más información, consulte Configuración del grupo de parámetros del clúster de base de datos para Babelfish. En ese tema, también encontrará información sobre el uso de la Consola de administración de AWS para realizar la actualización.

Utilice el comando de la CLI create-db-clúster-parameter-group ara crear el grupo de parámetros del clúster de base de datos para la nueva versión.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
    --db-parameter-group-family aurora-postgresql14 \
    --description 'New custom parameter group for upgrade to new major version' \
    --region us-west-1
```

Al ejecutar este comando, el grupo de parámetros del clúster de base de datos personalizado se crea en la Región de AWS. Se muestra una salida similar a la siguiente.

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14",
        "DBParameterGroupFamily": "aurora-postgresql14",
        "Description": "New custom parameter group for upgrade to new major version",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-west-1:111122223333:cluster-
pg:docs-lab-babelfish-apg-14"
    }
}
```

Para obtener más información, consulte Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Utilice el comando de la CLI modify-db-clúster-parameter-group para modificar la configuración de forma que coincida con el clúster de origen.

Para Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name docs-lab-
babelfish-apg-14 ^
  --parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_precision,ParameterValue=38,ApplyMethod=pen
reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_scale,ParameterValue=8,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.database_name,ParameterValue=babelfish_db,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.default_locale,ParameterValue=en-
US,ApplyMethod=pending-reboot" ^
```

```
 "ParameterName=babelfishpg_tsql.migration_mode,ParameterValue=single-
db,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.server_collation_name,ParameterValue=sql_latin1_general_cp1_ci
reboot"
```

La respuesta tiene un aspecto similar a la siguiente.

```
{
    "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14"
}
```

Utilice el comando de la CLI modify-db-clúster para modificar el clúster y utilizar la nueva versión y el nuevo grupo de parámetros del clúster de base de datos personalizado. También especifica el argumento `--allow-major-version-upgrade`, como se muestra en el siguiente ejemplo.

```
aws rds modify-db-cluster \
--db-cluster-identifier docs-lab-bfish-apg-14 \
--engine-version 14.6 \
--db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
--allow-major-version-upgrade \
--region us-west-1 \
--apply-immediately
```

Utilice el comando de la CLI reboot-db-instance para reiniciar la instancia de escritor del clúster, de modo que la configuración de los parámetros se aplique.

```
aws rds reboot-db-instance \
--db-instance-identifier docs-lab-bfish-apg-14-instance-1\
--region us-west-1
```

Uso del parámetro de versión del producto de Babelfish

A partir de las versiones 2.4.0 y 3.1.0 de Babelfish, se ha introducido un nuevo parámetro de Grand Unified Configuration (GUC) denominado `babelfishpg_tds.product_version`. Este parámetro permite establecer el número de versión del producto de SQL Server como la salida de Babelfish.

El parámetro es una cadena de identificador de versión de 4 partes y cada parte debe estar separada por “.”.

Sintaxis

```
Major.Minor.Build.Revision
```

- Versión principal: un número entre 11 y 16.
- Versión secundaria: un número entre 0 y 255.
- Versión de compilación: un número entre 0 y 65535.
- Revisión: 0 y cualquier número positivo.

Configuración del parámetro de versión del producto de Babelfish

Debe utilizar el grupo de parámetros del clúster para configurar el parámetro babelfishpg_tds.product_version en la consola. Para obtener más información sobre cómo modificar un parámetro de clúster de base de datos, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Si configura el parámetro de versión del producto en un valor no válido, el cambio no se aplicará. Aunque es posible que la consola le muestre el nuevo valor, el parámetro conserva el valor anterior. Consulte el archivo de registro del motor para obtener más detalles sobre los mensajes de error.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name mydbparametergroup \
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
--db-cluster-parameter-group-name mydbparametergroup ^
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Consultas y parámetros afectados

Consulta/parámetro	Resultado	Tiempo que tarda en aplicarse
SELECT @@VERSION	Devuelve la versión de SQL Server definida por el usuario (valor de babelfishpg_tsql.version = predeterminado)	De inmediato
SELECT SERVERPROPERTY('ProductVersion')	Devuelve la versión de SQL Server definida por el usuario	De inmediato
SELECT SERVERPROPERTY('ProductMajorVersion')	Devuelve la versión principal de la versión de SQL Server definida por el usuario	De inmediato
Tokens VERSION en el mensaje de respuesta de PRELOGIN	El servidor devuelve mensajes PRELOGIN con la versión de SQL Server definida por el usuario	Se aplica cuando un usuario crea una nueva sesión
SQLServerVersion en LoginAck cuando se usa JDBC	DatabaseMetaData.getDatabaseProductVersion() devuelve la versión de SQL Server definida por el usuario	Se aplica cuando un usuario crea una nueva sesión

Interfaz con el parámetro babelfishpg_tsql.version

Puede configurar la salida de @@VERSION mediante los parámetros babelfishpg_tsql.version y babelfishpg_tds.product_version. En los siguientes ejemplos, se muestra cómo interactúan estos dos parámetros.

- Cuando el parámetro babelfishpg_tsql.version es “default” y babelfishpg_tds.product_version es 15.0.2000.8.
  - Salida de @@version: 15.0.2000.8.
- Cuando el parámetro babelfishpg_tsql.version está establecido en 13.0.2000.8 y el parámetro babelfishpg_tds.product_version es 15.0.2000.8.

- Salida de @@version: 13.0.2000.8.

Referencia de Babelfish para Aurora PostgreSQL

Temas

- Funcionalidades compatibles con Babelfish por versión
- Funcionalidades no compatibles con Babelfish
- Uso de los procedimientos de Babelfish para Aurora PostgreSQL

Funcionalidades compatibles con Babelfish por versión

En la siguiente tabla, encontrará una lista de todas las funcionalidades compatibles en las distintas versiones de Babelfish. Para ver una lista de las funciones no admitidas, consulte Funcionalidades no compatibles con Babelfish. Para obtener información sobre diversas versiones de Babelfish, consulte las Notas de la versión de Aurora PostgreSQL.

Características de Aurora y PostgreSQL

En la siguiente tabla, encontrará las características de Aurora y PostgreSQL compatibles en diferentes versiones de Babelfish.

Características de Aurora y PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Servicios de Aurora ML	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Autenticación de bases de datos con Kerberos mediante Directory Service	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
Volcado y restauración	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Extensión pg_stat_statement																																

Características de Aurora y PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
pgvector	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Aplicación de parches sin tiempo de inactividad (ZDP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No

Funcionalidad T-SQL o sintaxis

En la siguiente tabla, encontrará la funcionalidad de T-SQL o sintaxis compatible con diferentes versiones de Babelfish.

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Referencias de nombres de objetos en cuatro partes para instrucciones SELECT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
Sintaxis ALTER AUTHORIZATION para cambiar el propietario de la base de datos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER DATABASE *<db_name>* MODIFY NAME = *<new_db_name>*	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
ALTER FUNCTION	S	S	S	S	S	N	N	N	N	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER PROCEDURE	S	S	S	S	S	S	S	N	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
ALTER USER...WITH LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER VIEW	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Palabra clave AS en CREATE FUNCTION	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Cláusula AT TIME ZONE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Instancia de Babelfish como servidor vinculado	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	No
Operadores de comparación !< y !>	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CREATE INDEX... ON partition _scheme_name (partition_column_ name)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE en lugar de desencadenadores (DML) en las vistas de SQL Server	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE PARTITION FUNCTION	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE PARTITION SCHEME	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREAR ROL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
CREATE TABLE... ON partition _scheme_name (partition_column_ name)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
CREATE OR ALTER VIEW	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
Creación de índices únicos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Ejecución de procedimientos entre bases de datos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
Referencias entre bases de datos SELECT, INSERT, UPDATE, DELETE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Parámetros escritos con el cursor solo para los parámetros de entrada (no para los de salida)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Migración de datos mediante la utilidad de cliente bcp	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Tipos de datos TIMESTAMP, ROWVERSION (para obtener información sobre el uso, consulte Características con implementación limitada)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
Palabra clave DEFAULT en las llamadas a procedimientos y funciones almacenados.	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DBCC CHECKIDENT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP DATABASE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DROP IF EXISTS (para objetos SCHEMA, DATABASE y USER)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Índice DROP INDEX ON schema.table	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP INDEX schema.table.index	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP PARTITION FUNCTION	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP PARTITION SCHEME	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
DROP ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
ENABLE/DISABLE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
FULL TEXT SEARCH	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Búsqueda de texto completo con cláusula CONTAINS	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Tipos de datos espaciales de geometría y geografía	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
GRANT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Permiso GRANT [,...n] ON SCHEMA	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
GUC babelfish pg_tds.product_version	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	No
Identificadores con carácter de punto inicial	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Desencadenadores INSTEAD OF en tablas	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Desencadenadores INSTEAD-OF en vistas	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
KILL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
$PARTITION.partition_function_name(partition_column_value)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
PIVOT (uso en la definición de vista de 4.4.0 y 3.8.0, expresión de tabla común de 3.4.0, unión de 3.4.0)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
REVOKE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Permiso REVOKE [,...n] ON SCHEMA	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Cláusulas SELECT... OFFSET...FETCH	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SELECT FOR JSON AUTO	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SET BABELFISH_SHOWPLAN_ALL ON (y OFF)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SET BABELFISH_STATISTICS PROFILE ON (OFF)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SET CONTEXT_INFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
SET LOCK_TIMEOUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SET NO_BROWSETABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET número de filas	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
SET SHOWPLAN_ALL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET STATISTICS IO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Sintaxis SET TRANSACTION ISOLATION LEVEL	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SSMS: conexión con el cuadro de diálogo de conexión del explorador de objetos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SSMS: migración de datos con el asistente de importación y exportación	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SSMS: compatibilidad parcial del explorador de objetos	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
STDEV	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
STDEVP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	No
Los desencadenadores con varias acciones de DML pueden hacer referencia a tablas de transición	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Sugerencias de T-SQL (métodos de unión, uso de índices, MAXDOP).	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
La sintaxis de corchetes de T-SQL con el predicado LIKE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	No
UNPIVOT	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Valores de cadena sin comillas en llamadas a procedimientos almacenados y valores predeterminados	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
VAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	No

Funcionalidad T-SQL o sintaxis	5.5.4	5.4.4	4.4.4	4.4.3	3.3.3	3.3.2	2.2.2	2.2.1.0
VARP	Sí	Sí	Sí	Sí	Sí	Sí	No	No

Funciones de T-SQL integradas

En la siguiente tabla, encontrará las funciones de T-SQL integradas compatibles con diferentes versiones de Babelfish.

Funciones de T-SQL integradas	5.5.4	5.4.4	4.4.4	4.4.3	3.3.3	3.3.2	2.2.2	2.2.1.0
APP_NAME	Sí	Sí	Sí	Sí	Sí	Sí	No	No
ATN2	Sí	Sí	Sí	Sí	Sí	Sí	No	No
CHARINDEX	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
CHOOSE	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
COL_LENGTH	Sí	Sí	Sí	Sí	Sí	Sí	No	No
COL_NAME	Sí	Sí	Sí	Sí	Sí	Sí	No	No
COLUMNS_UPDATED								

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
COLUMNPROPERTY (CharMaxLen, solo AllowsNull)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
CONCAT_WS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
CONTEXT_INFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No		
CURSOR_STATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
DATABASE_PRINCIPAL_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No		
DATEADD	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No			
DATEDIFF	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No			
DATEDIFF_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No			
DATEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí		

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
DATENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATEPART	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATETIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí		
DATETIME2FROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No		
DATETIMEOFFSETFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATETRUNC	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	N	N	N	N	N	N	No		
DATE_BUCKET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	N	N	N	N	N	N	No		
EOMONTH	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	N	N	N	N	N	N	No		
EXECUTE AS CALLER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
fn_listextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No		

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
FOR JSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
FULLTEXTS ERVICEPROPERTY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
HAS_DBACCESS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
HAS_PERMS _BY_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
HOST_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
HOST_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
IDENTITY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
IS_MEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
IS_ROLEMEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
IS_SRVROL EMEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
ISJSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
JSON_MODIFY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
JSON_QUERY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
JSON_VALUE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
NEXT VALUE FOR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
OBJECT_DE FINITION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
OBJECT_SC HEMA_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
OPENJSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
OPENQUERY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
ORIGINAL_LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
PARSENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
PATINDEX	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
ROWCOUNT_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
SCHEMA_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SESSION_C ONTEXT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
SESSION_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SID_BINARY (siempre devuelve NULL)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SMALLDATE TIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	S	N	N	N	N	No
SQUARE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
STR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
STRING_AGG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	N	N	N	N	N	N	N	N	No
STRING_SPLIT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SUSER_SID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SUSER_SNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
SWITCHOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SYSTEM_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
TIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
TODATETIMEOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
TO_CHAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
TRIGGER_NESTLEVEL (solo sin argumentos)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Funciones de T-SQL integradas	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2 1.0
TRY_CONVERT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
TYPE_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
TYPE_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
UPDATE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Catálogos INFORMATION_SCHEMA de T-SQL

En la siguiente tabla, encontrará los catálogos INFORMATION_SCHEMA de T-SQL compatibles con diferentes versiones de Babelfish.

Catálogos INFORMATION_SCHEMA de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
CHECK_CONSTRAINTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
COLUMN_DOMAIN_USAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
COLUMNS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Catálogos INFORMATION_SCHEMA de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.0 1.0
CONSTRAINT_COLUMN_USAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
DOMAINS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
KEY_COLUMN_USAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ROUTINES	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
TABLES	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
TABLE_CONSTRAINTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
VIEWS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Variables @@ definidas por el sistema de T-SQL

En la siguiente tabla, encontrará las variables @@ definidas por el sistema de T-SQL compatibles con diferentes versiones de Babelfish.

Variables @@ definidas por el sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
@@CURSOR_ ROWS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@DATEFIRST	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@DBTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@ERROR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@ERROR=213	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@FETCH_S TATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@IDENTITY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@LANGUAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@LOCK_TI MEOUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@MAX_CON NECTIONS																																

Variables @@ definidas por el sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@MAX_PRECISION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@MICROSOFTVERSION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@NESTLEVEL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@PROCID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@ROWCOUNT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@SERVERNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@SERVICENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@SPID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
@@TRANCOUNT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Variables @@ definidas por el sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
@@VERSION (observe la diferencia de formato descrita en [Diferencias de T-SQL en Babelfish](#)).	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Procedimientos almacenados del sistema de T-SQL

En la siguiente tabla, encontrará los procedimientos almacenados del sistema de T-SQL compatibles con diferentes versiones de Babelfish.

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
sp_addextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sp_addlinkedserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_addlinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_addrole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_addrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
sp_babelfish_volatility	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_column_privileges	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_columns_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_columns_managed	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursor	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursor_list	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursorclose	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursorexecute	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_cursorfetch	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursoropen	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursoroption	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursorprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursorprepexec	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_cursorunprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_databases	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_datatype_info	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_datatype_info_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
sp_describe_cursor	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_describe_first_result_set	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_describe_undeclared_parameters	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_dropextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sp_droplinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_droprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_droprolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_dropserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_enum_oledb_providers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_execute	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_execute_postgresql(CREATE, ALTER, DROP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No		
sp_executesql	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_fkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_getapplock	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_helpdb	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_helpdbfixedrole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
sp_helplinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
sp_helprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_helprolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_helpsrvrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
sp_helpuser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_linkedservers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	No
sp_oledb_ro_usrname	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_pkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_prefix	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
sp_prepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_procedure_params_100_managed	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_releaseapplock	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_rename	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_renamedb	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sp_rename (para el tipo de objeto 'DATABASE')	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sp_reset_connection	S	S	S	S	S	N	N	N	N	S	S	S	N	N	N	N	N	N	N	S	S	S	N	N	N	N	N	N	N	N	N	No
sp_serveroption(connect_timeout option)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	No
sp_set_session_context	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_special_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_sproc_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
sp_sproc_columns_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_statistics	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_statistics_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_stored_procedures	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_table_privileges	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_tablecollations_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_tables	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sp_testlinkedserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sp_unprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Procedimientos almacenados del sistema de T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
sp_updateextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sp_who	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
xp_qv	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Propiedades de T-SQL compatibles con la función del sistema CONNECTIONPROPERTY

En la siguiente tabla, encontrará propiedades de T-SQL compatibles con la función del sistema CONNECTIONPROPERTY y con diferentes versiones de Babelfish.

Propiedades de T-SQL compatibles con la función del sistema CONNECTIONPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
auth_scheme	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
client_net_address	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
local_net_address	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Propiedades de T-SQL compatibles con la función del sistema CONNECTIONPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
local_tcp_port	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
net_transport	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
protocol_type	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
physical_net_transport	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Propiedades de T-SQL compatibles con la función del sistema OBJECTPROPERTY

En la siguiente tabla, encontrará propiedades de T-SQL compatibles con la función del sistema OBJECTPROPERTY y con diferentes versiones de Babelfish.

Propiedades de T-SQL compatibles con la función del sistema OBJECTPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
IsInlineFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Propiedades de T-SQL compatibles con la función del sistema OBJECTPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
IsScalarFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
IsTableFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Propiedades de T-SQL compatibles con la función SERVERPROPERTY

En la siguiente tabla, encontrará propiedades de T-SQL compatibles con la función SERVERPROPERTY y con diferentes versiones de Babelfish.

Propiedades de T-SQL compatibles con la función SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Babelfish	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Collation (Intercalación)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
CollationID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Edición																																

Propiedades de T-SQL compatibles con la función SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
EditionID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
EngineEdition	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
InstanceName	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
está instalado Advanced Analytics	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
IsBigDataCluster	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
está instalado el texto completo	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Solo seguridad integrada	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
IsLocalDB	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Propiedades de T-SQL compatibles con la función SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
IsPolyBaseInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
IsSingleUser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Compatible con ISXTP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Japanese_CI_AI	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Japanese_CI_AS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Japanese_CS_AS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
LicenseType	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
MachineName	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
ProductLevel	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No

Propiedades de T-SQL compatibles con la función SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
Versión principal del producto	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
Versión menor del producto	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
ProductUpdateLevel	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
Versión del producto	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
ServerName (Nombre de servidor)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Vistas de SQL Server compatibles con Babelfish

En la siguiente tabla, encontrará vistas de SQL Server compatibles con diferentes versiones de Babelfish.

Vistas de SQL Server compatibles con Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
information_schema.key_column_usage	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No

Vistas de SQL Server compatibles con Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	20 1.0
information_schema .routines	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
information_schema .schemata	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
information_schema .sequences	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sys.all_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.all_objects	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.all_parameters	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sys.all_sql_modules	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.all_views	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Vistas de SQL Server compatibles con Babelfish	5.5	5.4	4.4	4.4	4.4	4.4	4.4	4.4	4.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.1	1.0
sys.configurations	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.data_spaces	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.database_files	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.database_mirroring	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.database_principals	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.database_role_members	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.databases	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.destination_data_spaces	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.dm_exec_connections	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Vistas de SQL Server compatibles con Babelfish	5.5.4.4.4.4.4.4.4.3.3.3.3.3.3.3.3.3.2.2.2.2.2.2.2.2.2.2.2.2.2.2	1.0
sys.dm_exec_sessions	S	Sí
sys.dm_hadr_database_replica_states	S	Sí
sys.dm_os_host_info	S	Sí
sys.dm_os_sys_info	S	No
sys.endpoints	S	Sí
sys.extended_properties	S	No
sys.indexes	S	Sí
sys.partitions	S	No
sys.partition_functions	S	No

Vistas de SQL Server compatibles con Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	20 1.0
sys.partition_parameters	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_range_values	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_schemes	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.schemas	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.server_principals	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.server_role_members	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.sql_modules	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.sysconfigures	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.syscurconfigs	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí

Vistas de SQL Server compatibles con Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
sys.syslogins	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sys.sysprocesses	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.sysusers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sys.table_types	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.tables	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sys.types	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sys.xml_schema_collections	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
syslanguages	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sí
sysobjects.crdate	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No

Funcionalidades no compatibles con Babelfish

En las siguientes listas y tablas puede encontrar la funcionalidad que actualmente no es compatible con Babelfish. Las actualizaciones de Babelfish se incluyen en las versiones de Aurora PostgreSQL. Para obtener más información, consulte las Notas de la versión de Aurora PostgreSQL.

Temas

- Funcionalidad no compatible actualmente
- Configuraciones no admitidas
- Comandos que no son compatibles
- Nombres de columnas o atributos que no son compatibles
- Tipos de datos que no son compatibles
- Tipos de objetos que no son compatibles
- Funciones que no son compatibles
- Sintaxis no compatible

Funcionalidad no compatible actualmente

En la tabla puede encontrar información acerca de ciertas funcionalidades no admitidas actualmente.

Funcionalidad o sintaxis	Descripción
Módulos de ensamblaje y rutinas SQL Common Language Runtime (CLR)	No se admite la funcionalidad relacionada con los módulos de ensamblado y las rutinas de CLR.
Atributos de columna	ROWGUIDCOL, SPARSE, FILESTREAM y MASKED no se admiten.
Bases de datos contenidas	No se admiten las bases de datos contenidas con inicios de sesión autenticados a nivel de base de datos en lugar de servidor.
DDL entre bases de datos	Aún no se admite la realización de instrucciones DDL que hagan referencia a objetos de varias bases de datos o que operen en ellos.

Funcionalidad o sintaxis	Descripción
Cursores (actualizables)	No se admiten cursores actualizables.
Cursores (globales)	No se admiten cursores GLOBALES.
Cursor (comportamientos de recuperación)	No se admiten los siguientes comportamientos de recuperación del cursor: FETCH PRIOR, FIRST, LAST, ABSOLUTE y RELATIVE
Parámetros de salida de tipo cursor	Las variables y parámetros de tipo cursor no son compatibles con los parámetros de salida (se produce un error).
Opciones del cursor	SCROLL, KEYSET, DYNAMIC, FAST_FORWARD, SCROLL_LOCKS, OPTIMISTIC, TYPE_WARNING y FOR UPDATE
Cifrado de datos	No se admite el cifrado de datos.
Aplicaciones de nivel de datos (DAC, por sus siglas en inglés)	No se admiten las operaciones de importación o exportación de aplicaciones de nivel de datos (DAC) con archivos de paquetes DAC (.dacpac) o de copia de seguridad de DAC (.bacpac).
Comandos de DBCC	No se admiten los comandos de consola de base de datos (DBCC) de Microsoft SQL Server. DBCC CHECKIDENT se admite en Babelfish 3.4.0 y versiones posteriores.
DROP IF EXISTS	Esta sintaxis no se admite para objetos USER y SCHEMA. Es compatible con los objetos TABLE, VIEW, PROCEDURE, FUNCTION y DATABASE.
Cifrado	Las funciones e instrucciones integradas no admiten el cifrado.
Conexiones ENCRYPT_CLIENT_CERT	No se admiten las conexiones de certificado de cliente.
Instrucción EXECUTE AS	No se admite esta instrucción.
Cláusula EXECUTE AS SELF	Esta cláusula no se admite en funciones, procedimientos o desencadenadores.

Funcionalidad o sintaxis	Descripción
Cláusula EXECUTE AS USER	Esta cláusula no se admite en funciones, procedimientos o desencadenadores.
Restricciones de clave externa que hacen referencia al nombre de la base de datos	No se admiten las restricciones de clave externa que referenci an el nombre de la base de datos.
FORMAT	No se admiten los tipos definidos por el usuario.
Declaraciones de funciones con más de 100 parámetros	No se admiten declaraciones de funciones que contengan más de 100 parámetros.
Llamadas a funciones que incluyen DEFAULT como valor de parámetro	DEFAULT no es un valor de parámetro admitido para una llamada a una función. DEFAULT como valor de parámetro para una llamada de función es compatible para versiones de Babelfish a partir de la versión 3.4.0.
Funciones definidas externamente	Las funciones externas, incluidas las funciones SQL CLR, no son compatibles.
Tablas temporales globales (tablas con nombres que comienzan por ##)	No se admiten las tablas temporales globales.
Funcionalidad de grafo	No se admiten todas las funcionalidades de grafo de SQL.
Procedimientos almacenados generalmente extendidos	No se admiten los procedimientos almacenados del sistema que proporcionan una interfaz desde una instancia de SQL Server a programas externos para diversas actividades de mantenimiento. Esto incluye `xp_cmdshell` y otros procedimi entos almacenados del sistema. Para obtener más información, consulte General Extended stored procedures.
Identificadores (variables o parámetros) con varios caracteres @ iniciales	No se admiten los identificadores que comienzan por más de un `@` inicial.

Funcionalidad o sintaxis	Descripción
Identificadores, nombres de tabla o columna que contienen los caracteres de @ o]]	No se admiten los nombres de tablas o columnas que contienen un signo de @ ni corchetes.
Índices insertados	No se admiten los índices insertados.
Invocación de un procedimiento cuyo nombre se encuentra en una variable	No se admite el uso de una variable como nombre de procedimiento.
Vistas materializadas	No se admiten las vistas materializadas.
Cláusula NOT FOR REPLICATION	Esta sintaxis se acepta e ignora.
Funciones de escape ODBC	No se admiten las funciones de escape de ODBC.
Llamadas a procedimientos que incluyen DEFAULT como valor de parámetro	DEFAULT no es un valor de parámetro compatible. DEFAULT como valor de parámetro para una llamada de función es compatible para versiones de Babelfish a partir de la versión 3.4.0.
Declaraciones de procedimiento con más de 100 parámetros	No se admiten declaraciones con más de 100 parámetros.
Procedimientos definidos externamente	No se admiten los procedimientos definidos externamente, incluidos los procedimientos SQL CLR.
Control de versiones de procedimientos	No se admite el control de versiones de procedimientos.
Procedimientos WITH RECOMPILE	No se admite WITH RECOMPILE (cuando se usa junto con las instrucciones DECLARE y EXECUTE).

Funcionalidad o sintaxis	Descripción
Referencias de objetos remoto	No se admite la ejecución de procedimientos almacenados en servidores enlazados de Babelfish. Los nombres de objetos de cuatro partes solo sirven para leer y no para modificar la tabla remota. Un UPDATE puede hacer referencia a una tabla remota de la cláusula FROM sin modificarla. Para obtener más información, consulte Babelfish admite servidores enlazados.
Seguridad de nivel básico	No se admite la seguridad de nivel de fila con CREATE SECURITY POLICY y funciones de valor de tabla integradas.
Funcionalidad de Service Broker	No se admite la funcionalidad de Service Broker.
PROPIEDAD SESSION	Propiedades no admitidas: ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL y NUMERIC_ROUNDABORT
SET LANGUAGE	Esta sintaxis no se admite con ningún valor que no sea `english` o `us_english` .
SP_CONFIGURE	No se admite este procedimiento almacenado del sistema.
Palabra clave `SPARSE` de SQL	La palabra clave SPARSE se acepta e ignora.
Sintaxis del constructor de valores de tabla (cláusula FROM)	La sintaxis no admitida corresponde a una tabla derivada construida con la cláusula FROM.
Tablas temporales	No se permite usar tablas temporales.
Los procedimientos temporales no se eliminan automáticamente	No se admite esta funcionalidad.
Desencadenadores definidos externamente	No se admiten estos desencadenadores, incluido SQL Common Language Runtime (CLR).

Funcionalidad o sintaxis	Descripción
Sin cláusula SCHEMABIN DING	No se admite la creación de una vista sin SCHEMABIN DING, pero se crea como si se hubiera especificado WITH SCHEMABINDING. El uso de SCHEMABINDING al crear funciones, procedimientos y disparadores se ignora de forma silenciosa.

Configuraciones no admitidas

No se admiten las siguientes acciones:

- SET ANSI_NULL_DFLT_OFF ON
- SET ANSI_NULL_DFLT_ON OFF
- SET ANSI_PADDING OFF
- SET ANSI_WARNINGS OFF
- SET ARITHABORT OFF
- SET ARITHIGNORE ON
- SET CURSOR_CLOSE_ON_COMMIT ON
- SET NUMERIC_ROUNDABORT ON
- SET PARSEONLY ON (el comando no funciona según lo previsto)
- SET FMTONLY ON (el comando no funciona según lo previsto, solo suprime la ejecución de las instrucciones SELECT, pero no de otras).

Comandos que no son compatibles

Algunas funcionalidades de los siguientes comandos no son compatibles:

- ADD SIGNATURE
- ALTER DATABASE, ALTER DATABASE SET
- BACKUP/RESTORE DATABASE/LOG
- BACPAC y DACPAC FILES RESTORE
- CREATE, ALTER, DROP AUTHORIZATION. ALTER AUTHORIZATION se admite para objetos de base de datos.

- CREATE, ALTER, DROP AVAILABILITY GROUP
- CREATE, ALTER, DROP BROKER PRIORITY
- CREATE, ALTER, DROP COLUMN ENCRYPTION KEY
- CREATE, ALTER, DROP DATABASE ENCRYPTION KEY
- CREATE, ALTER, DROP, BACKUP CERTIFICATE
- CREATE AGGREGATE
- CREATE CONTRACT
- CHECKPOINT

Nombres de columnas o atributos que no son compatibles

No se admiten los siguientes nombres de columna:

- $IDENTITY
- $ROWGUID
- IDENTITYCOL

Tipos de datos que no son compatibles

Los tipos de datos siguientes son compatibles:

- HIERARCHYID

Tipos de objetos que no son compatibles

Los siguientes tipos de objetos no son compatibles:

- COLUMN MASTER KEY
- CREATE, ALTER EXTERNAL DATA SOURCE
- CREATE, ALTER, DROP DATABASE AUDIT SPECIFICATION
- CREATE, ALTER, DROP EXTERNAL LIBRARY
- CREATE, ALTER, DROP SERVER AUDIT
- CREATE, ALTER, DROP SERVER AUDIT SPECIFICATION
- CREATE, ALTER, DROP, OPEN/CLOSE SYMMETRIC KEY

- CREATE, DROP DEFAULT
- CREDENTIAL
- CRYPTOGRAPHIC PROVIDER
- DIAGNOSTIC SESSION
- Vistas indexadas
- SERVICE MASTER KEY
- SYNONYM

Funciones que no son compatibles

La siguientes funciones integradas no son compatibles:

Funciones de agregación

- APPROX_COUNT_DISTINCT
- CHECKSUM_AGG
- GROUPING_ID
- STRING_AGG con la cláusula WITHIN GROUP

Funciones criptográficas

- Función CERTENCODED
- Función CERTID
- Función CERTPROPERTY

Funciones de metadatos

- COLUMNPROPERTY
- TYPEPROPERTY
- Función SERVERPROPERTY: no se admiten las siguientes propiedades:
  - BuildClrVersion
  - ComparisonStyle
  - ComputerNamePhysicalNetBIOS

- HadrManagerStatus
- InstanceDefaultDataPath
- InstanceDefaultLogPath
- Isclústered
- IsHadrEnabled
- LCID
- NumLicenses
- ProcessID
- ProductBuild
- ProductBuildType
- ProductUpdateReference
- ResourceLastUpdateDateTime
- ResourceVersion
- ServerName (Nombre de servidor)
- SqlCharSet
- SqlCharSetName
- SqlSortOrder
- SqlSortOrderName
- FilestreamShareName
- FilestreamConfiguredLevel
- FilestreamEffectiveLevel

Security functions

- CERTPRIVATEKEY
- LOGINPROPERTY

Declaraciones, operadores y otras funciones

- Función EVENTDATA
- GET_TRANSMISSION_STATUS
- OPENXML

Sintaxis no compatible

La siguiente sintaxis no es compatible:

- ALTER DATABASE
- ALTER DATABASE SCOPED CONFIGURATION
- ALTER DATABASE SCOPED CREDENTIAL
- ALTER DATABASE SET HADR
- ALTER INDEX
- ALTER PARTITION FUNCTION
- ALTER PARTITION SCHEME
- ALTER SCHEMA
- ALTER SERVER CONFIGURATION
- Cláusula ALTER SERVICE, BACKUP/RESTORE SERVICE MASTER KEY
- BEGIN CONVERSATION TIMER
- BEGIN DISTRIBUTED TRANSACTION
- BEGIN DIALOG CONVERSATION
- BULK INSERT
- CREATE COLUMNSTORE INDEX
- CREATE EXTERNAL FILE FORMAT
- CREATE EXTERNAL TABLE
- CREATE, ALTER, DROP APPLICATION ROLE
- CREATE, ALTER, DROP ASSEMBLY
- CREATE, ALTER, DROP ASYMMETRIC KEY
- CREATE, ALTER, DROP CREDENTIAL
- CREATE, ALTER, DROP CRYPTOGRAPHIC PROVIDER
- CREATE, ALTER, DROP ENDPOINT
- CREATE, ALTER, DROP EVENT SESSION
- CREATE, ALTER, DROP EXTERNAL LANGUAGE
- CREATE, ALTER, DROP EXTERNAL RESOURCE POOL
- CREATE, ALTER, DROP FULLTEXT CATALOG
- CREATE, ALTER, DROP FULLTEXT INDEX

- CREATE, ALTER, DROP FULLTEXT STOPLIST
- CREATE, ALTER, DROP MESSAGE TYPE
- CREATE, ALTER, DROP, OPEN/CLOSE, BACKUP/RESTORE MASTER KEY
- CREATE, ALTER, DROP QUEUE
- CREATE, ALTER, DROP RESOURCE GOVERNOR
- CREATE, ALTER, DROP RESOURCE POOL
- CREATE, ALTER, DROP ROUTE
- CREATE, ALTER, DROP SEARCH PROPERTY LIST
- CREATE, ALTER, DROP SECURITY POLICY
- CREATE, ALTER, DROP SELECTIVE XML INDEX clause
- CREATE, ALTER, DROP SERVICE
- CREATE, ALTER, DROP SPATIAL INDEX
- CREATE, ALTER, DROP TYPE
- CREATE, ALTER, DROP XML INDEX
- CREATE, ALTER, DROP XML SCHEMA COLLECTION
- CREATE/DROP RULE
- CREATE, DROP WORKLOAD CLASSIFIER
- CREATE, ALTER, DROP WORKLOAD GROUP
- ALTER TRIGGER
- CREATE TABLE... Cláusula GRANT
- CREATE TABLE... Cláusula IDENTITY
- CREATE USER: esta sintaxis no puede usarse. La instrucción CREATE USER de PostgreSQL no crea un usuario equivalente a la sintaxis CREATE USER de SQL Server. Para obtener más información, consulte Diferencias de T-SQL en Babelfish.
- DENY
- END, MOVE CONVERSATION
- EXECUTE with AS LOGIN or AT option
- GET CONVERSATION GROUP
- GROUP BY ALL clause
- GROUP BY CUBE clause
- GROUP BY ROLLUP clause

- INSERT... DEFAULT VALUES
- MERGE
- READTEXT
- REVERT
- SELECT TOP x PERCENT WHERE x <> 100
- SELECT TOP... WITH TIES
- SELECT... FOR BROWSE
- SELECT... FOR XML AUTO
- SELECT... FOR XML EXPLICIT
- SELECT... FOR XML PATH
- SEND
- SET DATEFORMAT
- SET DEADLOCK_PRIORITY
- SET FMTONLY
- SET FORCEPLAN
- SET NUMERIC_ROUNDABORT ON
- SET OFFSETS
- SET REMOTE_PROC_TRANSACTIONS
- SET SHOWPLAN_TEXT
- SET SHOWPLAN_XML
- SET STATISTICS
- SET STATISTICS PROFILE
- SET STATISTICS TIME
- SET STATISTICS XML
- SHUTDOWN statement
- UPDATE STATISTICS
- UPDATETEXT
- Using EXECUTE to call a SQL function
- VIEW... CHECK OPTION clause
- VIEW... VIEW_METADATA clause

- WAITFOR DELAY
- WAITFOR TIME
- WAITFOR, RECEIVE
- WITH XMLNAMESPACES construct
- WRITETEXT
- XPATH expressions

Uso de los procedimientos de Babelfish para Aurora PostgreSQL

Información general

Puede utilizar el siguiente procedimiento para las instancias de base de datos de Amazon RDS que ejecuten Babelfish para Aurora PostgreSQL para obtener un mejor rendimiento de las consultas:

- sp_babelfish_volatility
- sp_execute_postgresql

sp_babelfish_volatility

La volatilidad de las funciones de PostgreSQL ayuda al optimizador a ejecutar mejor las consultas, lo que, cuando se usa en partes de determinadas cláusulas, tiene un impacto significativo en el rendimiento de las consultas.

Sintaxis

```
sp_babelfish_volatility 'function_name', 'volatility'
```

Argumentos

function_name (opcional)

Puede especificar el valor de este argumento con un nombre de dos partes como `schema_name.function_name` o solo el `function_name`. Si especifica solo el `function_name`, el nombre del esquema será el esquema predeterminado para el usuario actual.

volatility (optional)

Los valores de volatilidad válidos de PostgreSQL son `stable`, `volatile` o `immutable`. Para obtener más información, consulte https://www.postgresql.org/docs/current/xfunc-volatility.html

> **Note**
>
> Cuando `sp_babelfish_volatility` se llama con un `function_name` con múltiples definiciones, arrojará un error.

Conjuntos de resultados

Si no se mencionan los parámetros, el conjunto de resultados se muestra en las siguientes columnas: `schemaname`, `functionname`, `volatility`.

Notas de uso

La volatilidad de las funciones de PostgreSQL ayuda al optimizador a ejecutar mejor las consultas, lo que, cuando se usa en partes de determinadas cláusulas, tiene un impacto significativo en el rendimiento de las consultas.

Ejemplos

En los siguientes ejemplos se muestra cómo crear funciones simples y, posteriormente, se explica cómo utilizar `sp_babelfish_volatility` en estas funciones mediante diferentes métodos.

```
1> create function f1() returns int as begin return 0 end
2> go
```

```
1> create schema test_schema
2> go
```

```
1> create function test_schema.f1() returns int as begin return 0 end
2> go
```

El siguiente ejemplo muestra la volatilidad de las funciones:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           volatile
test_schema f1           volatile
```

El siguiente ejemplo muestra cómo cambiar la volatilidad de las funciones:

```
1> exec sp_babelfish_volatility 'f1','stable'
2> go
1> exec sp_babelfish_volatility 'test_schema.f1','immutable'
```

```
2> go
```

Al especificar solo el function_name, se muestran el nombre del esquema, el nombre de la función y la volatilidad de esa función. El siguiente ejemplo muestra la volatilidad de las funciones después de cambiar los valores:

```
1> exec sp_babelfish_volatility 'test_schema.f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
test_schema f1           immutable
```

```
1> exec sp_babelfish_volatility 'f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
```

Si no especifica ningún argumento, se muestra una lista de funciones (nombre del esquema, nombre de la función, volatilidad de las funciones) presentes en la base de datos actual:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
test_schema f1           immutable
```

sp_execute_postgresql

Puede ejecutar instrucciones PostgreSQL desde el punto de conexión de T-SQL. Así se simplifican las aplicaciones, ya que no es necesario salir del puerto T-SQL para ejecutar las instrucciones.

Sintaxis

```
sp_execute_postgresql [ @stmt = ] statement
```

Argumentos

Instrucción [@stmt]

El argumento es de tipo varchar. Este argumento acepta instrucciones del dialecto PG.

> **Note**
>
> Como argumento solo puede especificarse una instrucción del dialecto PG; de lo contrario, se generará el error siguiente:

```
1>exec sp_execute_postgresql 'create extension pg_stat_statements; drop extension
 pg_stat_statements'
2>go
```

```
Msg 33557097, Level 16, State 1, Server BABELFISH, Line 1
expected 1 statement but got 2 statements after parsing
```

Notas de uso

CREATE EXTENSION

Crea y carga una nueva extensión en la base de datos actual.

```
1>EXEC sp_execute_postgresql 'create extension [ IF NOT EXISTS ] <extension name>
 [ WITH ] [SCHEMA schema_name] [VERSION version]';
2>go
```

En el ejemplo siguiente se muestra cómo crear una extensión:

```
1>EXEC sp_execute_postgresql 'create extension pg_stat_statements with schema sys
 version "1.10"';
2>go
```

Utilice el comando siguiente para obtener acceso a objetos de extensión:

```
1>select * from pg_stat_statements;
2>go
```

> **Note**
>
> Si el nombre del esquema no se proporciona de forma explícita cuando se crea la extensión, las extensiones se instalan de forma predeterminada en el esquema público. Debe proporcionar el calificador de esquema para tener acceso a los objetos de la extensión, como se indica a continuación:

```
1>select * from [public].pg_stat_statements;
2>go
```

Extensiones compatibles

Las siguientes extensiones disponibles con Aurora PostgreSQL funcionan con Babelfish.

- `pg_stat_statements`
- `tds_fdw`
- `fuzzystrmatch`

Limitaciones

- Los usuarios deben tener el rol sysadmin en T-SQL y rds_superuser en postgres para instalar las extensiones.
- Las extensiones no se pueden instalar en los esquemas creados por el usuario ni tampoco en los esquemas dbo y guest de las bases de datos master, tempdb y msdb.
- La opción CASCADE no es compatible.

ALTER EXTENSION

Puede actualizar a una nueva versión de la extensión utilizando ALTER EXTENSION.

```
1>EXEC sp_execute_postgresql 'alter extension <extension name> UPDATE TO
 <new_version>';
2>go
```

Limitaciones

- La versión de la extensión solamente se puede actualizar con la instrucción ALTER EXTENSION. No se admiten otras operaciones.

DROP EXTENSION

Elimina la extensión especificada. También puede usar las opciones `if exists` o `restrict` para eliminar la extensión.

```
1>EXEC sp_execute_postgresql 'drop extension <extension name>';
2>go
```

Limitaciones

- La opción CASCADE no es compatible.

Rendimiento y escalado para Amazon Aurora PostgreSQL

La siguiente sección trata sobre la administración del rendimiento y el escalado de un clúster de bases de datos de Amazon Aurora PostgreSQL. También incluye información sobre otras tareas de mantenimiento.

Temas

- Escalado de las instancias de base de datos Aurora PostgreSQL
- Número máximo de conexiones a una instancia de base de datos Aurora PostgreSQL
- Límites de almacenamiento temporal de Aurora PostgreSQL
- Páginas enormes para Aurora PostgreSQL
- Pruebas de Amazon Aurora PostgreSQL mediante consultas de inserción de errores
- Visualización del estado del volumen para un clúster de bases de datos de Aurora PostgreSQL
- Especificación del disco RAM para stats_temp_directory

- Administración de archivos temporales con PostgreSQL

Escalado de las instancias de base de datos Aurora PostgreSQL

Puede escalar las instancias de base de datos Aurora PostgreSQL de dos formas, mediante el escalado de instancia y mediante el escalado de lectura. Para obtener más información acerca del escalado de lectura, consulte Escalado de lectura.

Para escalar el clúster de bases de datos de Aurora PostgreSQL, modifique la clase de instancia de base de datos para cada instancia de base de datos del clúster de bases de datos. Aurora PostgreSQL admite varias clases de instancia de base de datos optimizadas para Aurora. No utilice las clases de instancia db.t2 o db.t3 con clústeres de Aurora que tengan más de 40 terabytes (TB).

> **Note**
>
> Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más detalles sobre las clases de instancia T, consulte Tipos de clase de instancia de base de datos.

El escalado no es instantáneo. Puede llevar 15 minutos o más completar el cambio a una clase de instancia de base de datos diferente. Si utiliza este enfoque para modificar la clase de instancia de base de datos, aplique el cambio durante el siguiente periodo de mantenimiento programado (en lugar de hacerlo de inmediato) para evitar afectar a los usuarios.

Como alternativa a modificar la clase de instancia de base de datos directamente, puede minimizar el tiempo de inactividad por medio de las características de alta disponibilidad de Amazon Aurora. En primer lugar, agregue una réplica de Aurora al clúster. Cuando cree la réplica, elija el tamaño de clase de la instancia de base de datos que desea utilizar para el clúster. Cuando la réplica de Aurora se sincroniza con el clúster, se realiza la conmutación por error a la réplica recién agregada. Para obtener más información, consulte Réplicas de Aurora y Conmutación por error rápida con Amazon Aurora PostgreSQL.

Para ver especificaciones detalladas de las clases de instancia de base de datos admitidas en Aurora PostgreSQL, consulte Motores de base de datos compatibles para clases de instancia de base de datos.

Número máximo de conexiones a una instancia de base de datos Aurora PostgreSQL

Un clúster de base de datos de Aurora PostgreSQL asigna recursos en función de la clase de instancia de base de datos y la memoria disponible. Cada conexión al clúster de base de datos consume cantidades incrementales de estos recursos, como memoria y CPU. La memoria consumida por conexión varía según el tipo de consulta, el recuento y si se utilizan tablas temporales. Incluso una conexión inactiva consume memoria y CPU. Esto se debe a que, cuando las consultas se ejecutan en una conexión, se asigna más memoria a cada consulta y esta no se libera por completo, ni siquiera cuando el procesamiento se detiene. Por lo tanto, le recomendamos que se asegure de que sus aplicaciones no mantienen las conexiones inactivas, ya que malgastan recursos y afectan negativamente al rendimiento. Para obtener más información, consulte Resources consumed by idle PostgreSQL connections.

El número máximo de conexiones permitidas por una instancia de base de datos de Aurora PostgreSQL está determinado por el valor del parámetro `max_connections` especificado en el grupo de parámetros para esa instancia de base de datos. La configuración ideal para el parámetro `max_connections` es una que admita todas las conexiones de clientes que su aplicación necesita, sin un exceso de conexiones no usadas, más al menos 3 conexiones más para soportar la automatización de AWS. Antes de modificar la configuración del parámetro `max_connections`, le recomendamos que considere las siguientes acciones:

- Si el valor de `max_connections` es muy bajo, la instancia de base de datos de Aurora PostgreSQL podría no tener suficientes conexiones disponibles cuando los clientes intenten conectarse. Si esto sucede, intenta conectarse mediante mensajes de generación de errores de `psql` como los siguientes:

  ```
  psql: FATAL: remaining connection slots are reserved for non-replication superuser
   connections
  ```

- Si el valor de `max_connections` excede el número de conexiones que realmente se necesitan, las conexiones no utilizadas pueden hacer que el rendimiento se degrade.

El valor predeterminado de `max_connections` se obtiene de la siguiente función `LEAST` de Aurora PostgreSQL:

`LEAST({DBInstanceClassMemory/9531392},5000)`.

Si desea cambiar el valor de `max_connections`, necesita crear un grupo de parámetros de clúster de base de datos personalizado para cambiarlo valor allí. Después de aplicar el grupo de parámetros de base de datos personalizado al clúster, asegúrese de reiniciar la instancia principal para que se aplique el nuevo valor. Para obtener más información, consulte Parámetros de Amazon Aurora PostgreSQL. y Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

> **Tip**
>
> Si sus aplicaciones abren y cierran conexiones con frecuencia, o mantienen abierto un gran número de conexiones de larga duración, le recomendamos que utilice Amazon RDS Proxy. El RDS Proxy es un proxy de base de datos totalmente administrado y de alta disponibilidad que utiliza agrupación de conexiones para compartir conexiones de base de datos de forma segura y eficiente. Para obtener más información acerca de RDS Proxy, consulteAmazon RDS Proxy para Aurora.

Para obtener más información acerca de cómo las instancias de Aurora Serverless v2 manejan este parámetro, consulte Número máximo de conexiones para Aurora Serverless v2.

Límites de almacenamiento temporal de Aurora PostgreSQL

Aurora PostgreSQL almacena tablas e índices en el subsistema de almacenamiento de Aurora. Aurora PostgreSQL utiliza almacenamiento temporal independiente para archivos temporales no persistentes. Esto incluye archivos que se utilizan para fines tales como ordenar conjuntos de datos grandes durante el procesamiento de consultas o para operaciones de creación de índices. Para obtener más información, consulte el artículo How can I troubleshoot local storage issues in Aurora PostgreSQL-Compatible instances? (¿Cómo puedo solucionar problemas de almacenamiento local en instancias compatibles con Aurora PostgreSQL?).

Estos volúmenes de almacenamiento local están respaldados por Amazon Elastic Block Store y pueden ampliarse utilizando una clase de instancia de base de datos mayor. Para obtener más información acerca del almacenamiento, consulte Almacenamiento de Amazon Aurora. También puede aumentar el almacenamiento local de objetos temporales mediante un tipo de instancia habilitado para NVMe y objetos temporales habilitados para las lecturas optimizadas de Aurora. Para obtener más información, consulte Mejora del rendimiento de las consultas de Aurora PostgreSQL con lecturas optimizadas de Aurora .

> **Note**
>
> Puede ver los eventos `storage-optimization` al escalar las instancias de base de datos, por ejemplo, de db.r5.2xlarge a db.r5.4xlarge.

En la siguiente tabla se muestra la cantidad máxima de almacenamiento temporal disponible para cada clase de instancia de base de datos de Aurora PostgreSQL. Para obtener más información sobre la compatibilidad de la clase de instancia de la base de datos con Aurora, consulte Clases de instancia de base de datos de Amazon Aurora.

Clase de instancia de base de datos	Almacenamiento temporal máximo disponible (GiB)
db.x2g.16xlarge	1829
db.x2g.12xlarge	1606
db.x2g.8xlarge	1071
db.x2g.4xlarge	535
db.x2g.2xlarge	268
db.x2g.xlarge	134
db.x2g.large	67
db.r8g.48xlarge	3072
db.r8g.24xlarge	1536
db.r8g.16xlarge	998
db.r8g.12xlarge	749
db.r8g.8xlarge	499
db.r8g.4xlarge	250

Clase de instancia de base de datos	Almacenamiento temporal máximo disponible (GiB)
db.r8g.2xlarge	125
db.r8g.xlarge	63
db.r8g.large	31
db.r7g.16xlarge	1008
db.r7g.12xlarge	756
db.r7g.8xlarge	504
db.r7g.4xlarge	252
db.r7g.2xlarge	126
db.r7g.xlarge	63
db.r7g.large	32
db.r7i.48xlarge	3072
db.r7i.24xlarge	1500
db.r7i.16xlarge	1008
db.r7i.12xlarge	748
db.r7i.8xlarge	504
db.r7i.4xlarge	249
db.r7i.2xlarge	124
db.r7i.xlarge	62
db.r7i.large	31
db.r6g.16xlarge	1008

Clase de instancia de base de datos	Almacenamiento temporal máximo disponible (GiB)
db.r6g.12xlarge	756
db.r6g.8xlarge	504
db.r6g.4xlarge	252
db.r6g.2xlarge	126
db.r6g.xlarge	63
db.r6g.large	32
db.r6i.32xlarge	1829
db.r6i.24xlarge	1500
db.r6i.16xlarge	1008
db.r6g.12xlarge	748
db.r6i.8xlarge	504
db.r6i.4xlarge	249
db.r6i.2xlarge	124
db.r6i.xlarge	62
db.r6i.large	31
db.r5.24xlarge	1500
db.r5.16xlarge	1008
db.r5.12xlarge	748
db.r5.8xlarge	504
db.r5.4xlarge	249

Clase de instancia de base de datos	Almacenamiento temporal máximo disponible (GiB)
db.r5.2xlarge	124
db.r5.xlarge	62
db.r5.large	31
db.r4.16xlarge	960
db.r4.8xlarge	480
db.r4.4xlarge	240
db.r4.2xlarge	120
db.r4.xlarge	60
db.r4.large	30
db.t4g.large	16,5
db.t4g.medium	8,13
db.t3.large	16
db.t3.medium	7.5

> **Note**
>
> Los tipos de instancias compatibles con NVMe pueden aumentar el espacio temporal disponible hasta el tamaño total de NVMe. Para obtener más información, consulte Mejora del rendimiento de las consultas de Aurora PostgreSQL con lecturas optimizadas de Aurora .

Puede supervisar el almacenamiento temporal disponible para una instancia de base de datos con la métrica de CloudWatch `FreeLocalStorage`, que se describe en Métricas de Amazon CloudWatch para Amazon Aurora. (Esto no se aplica a Aurora Serverless v2.)

Para algunas cargas de trabajo, puede reducir la cantidad de almacenamiento temporal asignando más memoria a los procesos que están realizando la operación. Para aumentar la memoria disponible de una operación, se aumentan los valores de los parámetros work_mem o maintenance_work_mem de PostgreSQL.

Páginas enormes para Aurora PostgreSQL

Las páginas enormes son una característica de administración de la memoria que reduce la sobrecarga cuando una instancia de base de datos trabaja con grandes fragmentos contiguos de memoria, como la utilizada por los búferes compartidos. Esta característica de PostgreSQL es compatible con todas las versiones de Aurora PostgreSQL disponibles actualmente.

El parámetro `Huge_pages` se activa de forma predeterminada para todas las clases de instancia de base de datos que no sean las siguientes: t3.medium, db.t3.large, db.t4g.medium y db.t4g.large. No puede cambiar el valor del parámetro `huge_pages` ni desactivar esta característica en las clases de instancias compatibles de Aurora PostgreSQL.

En las instancias de base de datos de Aurora PostgreSQL que no admiten la característica de memoria de páginas enormes, el uso de memoria de procesos específicos podría aumentar sin los correspondientes cambios en la carga de trabajo.

El sistema asigna segmentos de memoria compartida, como la caché del búfer, durante el inicio del servidor. Cuando no hay páginas de memoria enormes disponibles, el sistema no carga estas asignaciones al proceso postmaster. En su lugar, incluye la memoria en el proceso que accedió por primera vez a cada página de 4 KB del segmento de memoria compartida.

> **Note**
>
> Las conexiones activas comparten la memoria asignada según sea necesario, independientemente de cómo se realice el seguimiento del uso de la memoria compartida en los procesos.

Pruebas de Amazon Aurora PostgreSQL mediante consultas de inserción de errores

Puede probar la tolerancia a errores de su clúster de bases de datos de Aurora PostgreSQL usando consultas de inserción de errores. Las consultas de inyección de errores se emiten como comandos

SQL a una instancia Amazon Aurora. Las consultas de inyección de errores le permiten bloquear la instancia para poder probar la conmutación por error y la recuperación. También puede simular un fallo de Aurora Replica, un fallo de disco y una congestión del disco. Las consultas de inyección de errores son compatibles con todas las versiones disponibles de Aurora PostgreSQL, de la siguiente manera.

- Aurora PostgreSQL versiones 12, 13, 14 y posteriores
- Aurora PostgreSQL versión 11.7 y posteriores
- Aurora PostgreSQL versión 10.11 y posteriores

Temas

- Prueba de un bloqueo de instancia
- Prueba de un error de una réplica de Aurora
- Prueba de un error de disco
- Prueba de congestión del disco

Cuando una consulta de inserción de errores especifica un bloqueo, fuerza un bloqueo de la instancia de base de datos de Aurora PostgreSQL. Las otras consultas de inserción de errores producen simulaciones de eventos de error, pero no hacen que el evento ocurra. Cuando se envía una consulta de inserción de errores, se especifica también la cantidad de tiempo que debe durar la simulación del evento de error.

Puede enviar una consulta de inserción de errores a una de las instancias de réplica de Aurora conectándose al punto de enlace de la réplica de Aurora. Para obtener más información, consulte Conexiones de puntos de conexión de Amazon Aurora.

Prueba de un bloqueo de instancia

Puede forzar el bloqueo de una instancia de Aurora PostgreSQL mediante la función de consulta de inserción de errores `aurora_inject_crash()`.

En esta consulta de inserción de errores no se produce una conmutación por error. Si desea probar una conmutación por error, puede elegir la acción de instancia Failover (Conmutación por error) para el clúster de bases de datos en la consola de RDS o usar el comando failover-db-cluster de la AWS CLI o la operación FailoverDBCluster de la API de RDS.

Sintaxis

```
SELECT aurora_inject_crash ('instance' | 'dispatcher' | 'node');
```

Opciones

La consulta de inserción de errores toma uno de los siguientes tipos de bloqueos. El tipo de bloqueo no distingue entre mayúsculas y minúsculas:

'instance'

Se simula un bloqueo de la base de datos compatible con PostgreSQL para la instancia de Amazon Aurora.

“despachador”

Se simula un bloqueo del distribuidor de la instancia principal del clúster de bases de datos de Aurora. El distribuidor escribe actualizaciones en el volumen de clúster de un clúster de bases de datos Amazon Aurora.

'node'

Se simula un bloqueo de la base de datos compatible con PostgreSQL y del distribuidor para la instancia de Amazon Aurora.

Prueba de un error de una réplica de Aurora

Puede simular el error de una réplica de Aurora mediante la función de consulta de inserción de errores `aurora_inject_replica_failure()`.

Un error de réplica de Aurora bloquea la reproducción en la réplica de Aurora o en todas las réplicas de Aurora del clúster de bases de datos en el porcentaje especificado para el intervalo de tiempo especificado. Cuando se complete el intervalo de tiempo, las réplicas de Aurora afectadas se sincronizan automáticamente con la instancia principal.

Sintaxis

```
SELECT aurora_inject_replica_failure(
   percentage_of_failure,
   time_interval,
   'replica_name'
);
```

Opciones

Esta consulta de inserción de errores toma los siguientes parámetros:

percentage_of_failure

El porcentaje de reproducción que se debe bloquear durante el evento de error. Puede ser un valor doble entre 0 y 100. Si especifica 0, no se bloquea la reproducción. Si especifica 100, se bloquea toda la reproducción.

time_interval

Cantidad de tiempo para simular el error de la réplica de Aurora. El intervalo es en segundos. Por ejemplo, si el valor es 20, la simulación se ejecuta durante 20 segundos.

> **Note**
>
> Debe tener cuidado al especificar el intervalo de tiempo del evento de error de la réplica de Aurora. Si especifica un intervalo demasiado largo y la instancia de escritor escribe una gran cantidad de datos durante el evento de error, su clúster de bases de datos de Aurora podría entender que la réplica de Aurora se ha bloqueado y reemplazarla.

replica_name

La réplica de Aurora en la que se inserta la simulación de errores. Especifique el nombre de una réplica de Aurora para simular un error de la réplica de Aurora única. Especifique una cadena vacía para simular errores para todas las réplicas de Aurora en el clúster de bases de datos.

Para identificar nombres de réplica, consulte la columna `server_id` de la función `aurora_replica_status()`. Por ejemplo:

```
postgres=> SELECT server_id FROM aurora_replica_status();
```

Prueba de un error de disco

Puede simular un error de disco para un clúster de bases de datos de Aurora PostgreSQL mediante la función de consulta de inserción de errores `aurora_inject_disk_failure()`.

Durante la simulación de un error de disco, el clúster de bases de datos de Aurora PostgreSQL marca de forma aleatoria los segmentos de disco como defectuosos. Las solicitudes que lleguen a esos segmentos se bloquean mientras dure la simulación.

Sintaxis

```
SELECT aurora_inject_disk_failure(
   percentage_of_failure,
   index,
   is_disk,
   time_interval
);
```

Opciones

Esta consulta de inserción de errores toma los siguientes parámetros:

percentage_of_failure

El porcentaje del disco que se debe marcar como defectuoso durante el evento de error. Puede ser un valor doble entre 0 y 100. Si se especifica 0, ninguna parte del disco se marca como defectuosa. Si se especifica 100, todo el disco se marca como defectuoso.

índice

Un bloque lógico de datos específico en el que se simula el evento de error. Si se sobrepasa el intervalo de datos de nodos de almacenamiento o bloques lógicos disponibles, aparece un error que indica el valor máximo del índice que se puede especificar. Para evitar este error, consulte Visualización del estado del volumen para un clúster de bases de datos de Aurora PostgreSQL.

is_disk

Indica si el error de inserción se produce en un bloque lógico o en un nodo de almacenamiento. Especificar «true» significa que los errores de inserción son de un bloque lógico. Especificar «false» significa que los errores de inyección son en un nodo de almacenamiento.

time_interval

La cantidad de tiempo para simular el error de disco. El intervalo es en segundos. Por ejemplo, si el valor es 20, la simulación se ejecuta durante 20 segundos.

Prueba de congestión del disco

Puede simular una congestión de disco para un clúster de bases de datos de Aurora PostgreSQL mediante la función de consulta de inserción de errores `aurora_inject_disk_congestion()`.

Durante la simulación de congestión del disco, el clúster de bases de datos de Aurora PostgreSQL marca de forma aleatoria los segmentos de disco como congestionados. Las solicitudes que lleguen a esos segmentos se retrasan entre el mínimo especificado y el tiempo de demora máximo mientras dure la simulación.

Sintaxis

```
SELECT aurora_inject_disk_congestion(
   percentage_of_failure,
   index,
   is_disk,
   time_interval,
   minimum,
   maximum
);
```

Opciones

Esta consulta de inserción de errores toma los siguientes parámetros:

percentage_of_failure

El porcentaje del disco que se debe marcar como congestionado durante el evento de error. Este es un valor doble entre 0 y 100. Si se especifica 0, ninguna parte del disco se marca como congestionada. Si se especifica 100, todo el disco se marca como congestionado.

índice

Un bloque lógico de datos o nodo de almacenamiento específico en el que se simula el evento de error.

Si se sobrepasa el intervalo de nodos de almacenamiento o bloques lógicos de datos disponibles, aparece un error que indica el valor máximo del índice que se puede especificar. Para evitar este error, consulte Visualización del estado del volumen para un clúster de bases de datos de Aurora PostgreSQL.

is_disk

Indica si el error de inserción se produce en un bloque lógico o en un nodo de almacenamiento. Especificar «true» significa que los errores de inserción son de un bloque lógico. Especificar «false» significa que los errores de inyección son en un nodo de almacenamiento.

time_interval

La cantidad de tiempo para simular la congestión del disco. El intervalo es en segundos. Por ejemplo, si el valor es 20, la simulación se ejecuta durante 20 segundos.

mínimo, máximo

La cantidad mínima y máxima de demora de la congestión en milisegundos. Los valores válidos varían entre 0,0 y 100,0 milisegundos. Los segmentos de disco marcados como congestionados se retrasan por una cantidad de tiempo aleatoria dentro del rango mínimo y máximo mientras dure la simulación. El valor máximo debe ser mayor que el valor mínimo.

Visualización del estado del volumen para un clúster de bases de datos de Aurora PostgreSQL

En Amazon Aurora, un volumen de clúster de bases de datos se compone de un conjunto de bloques lógicos. Cada uno de esos bloques representa 10 gigabytes de almacenamiento asignado. Estos bloques se denominan grupos de protección.

Los datos de cada grupo de protección se replican en seis dispositivos de almacenamiento físicos denominados nodos de almacenamiento. Estos nodos de almacenamiento se distribuyen entre tres zonas de disponibilidad (AZ) en la región en la que reside el clúster de bases de datos. A su vez, cada nodo de almacenamiento contiene uno o varios bloques lógicos de datos para el volumen del clúster de bases de datos. Para obtener más información acerca de los grupos de protección y los nodos de almacenamiento, consulte Introducing the Aurora Storage Engine en el Blog de base de datos de AWS. Para obtener más información sobre volúmenes de clúster de Aurora en general, consulte Almacenamiento de Amazon Aurora.

Utilice la función `aurora_show_volume_status()` para devolver las siguientes variables de estado del servidor:

- `Disks`: el número total de bloques lógicos de datos para el volumen del clúster de bases de datos.
- `Nodes` — el número total de nodos de almacenamiento para el volumen del clúster de bases de datos.

Puede utilizar la función `aurora_show_volume_status()` para evitar un error al usar la función de inserción de errores `aurora_inject_disk_failure()`. La función de inserción de errores `aurora_inject_disk_failure()` simula el error de un nodo de almacenamiento completo o de un único bloque lógico de datos dentro de un nodo de almacenamiento. En la función, especifique el valor del índice de un bloque lógico de datos o nodo de almacenamiento concreto. Sin embargo, si especifica un valor del índice mayor que el número de bloques lógicos de datos o los nodos de almacenamiento utilizados por el volumen de clúster de bases de datos, la instrucción devuelve un error. Para obtener más información acerca de las consultas de inserción de errores, vea Pruebas de Amazon Aurora PostgreSQL mediante consultas de inserción de errores.

> **Note**
>
> La función `aurora_show_volume_status()` está disponible para la versión 10.11 de Aurora PostgreSQL. Para obtener más información acerca de las versiones de Aurora PostgreSQL, consulte Versiones de Amazon Aurora PostgreSQL y versiones del motor.

Sintaxis

```
SELECT * FROM aurora_show_volume_status();
```

Ejemplo

```
customer_database=> SELECT * FROM aurora_show_volume_status();
 disks | nodes
-------+-------
    96 |    45
```

Especificación del disco RAM para stats_temp_directory

Puede utilizar el parámetro de Aurora PostgreSQL `rds.pg_stat_ramdisk_size` para especificar la memoria del sistema asignada a un disco RAM para almacenar `stats_temp_directory` de PostgreSQL. El parámetro del disco RAM solo está disponible en la versión 14 e inferiores de Aurora PostgreSQL.

Para algunas cargas de trabajo, configurar este parámetro puede mejorar el rendimiento y reducir los requisitos de E/S. Para obtener más información sobre el `stats_temp_directory`, consulte Run-time Statistics (Estadísticas de tiempo de ejecución) en la documentación de PostgreSQL. A partir

de la versión 15 de PostgreSQL, la comunidad de PostgreSQL pasó a utilizar la memoria compartida dinámica. Por lo tanto, no es necesario configurar `stats_temp_directory`.

Para habilitar un disco RAM para `stats_temp_directory`, defina el parámetro `rds.pg_stat_ramdisk_size` en un valor distinto de cero en el grupo de parámetros de clúster de bases de datos que utilice su clúster de bases de datos. Este parámetro indica MB, por lo que debe utilizar un valor entero. Las expresiones, fórmulas y funciones no son válidas para el parámetro `rds.pg_stat_ramdisk_size`. Asegúrese de reiniciar el clúster de bases de datos para que el cambio surta efecto. Para obtener información acerca de cómo configurar los parámetros, consulte Grupos de parámetros para Amazon Aurora. Para obtener más información acerca de la conexión al clúster de base de datos, consulte Reinicio de un clúster de base de datos de Amazon Aurora o de una instancia de base de datos de Amazon Aurora.

Por ejemplo, el comando AWS CLI siguiente establece el parámetro del disco RAM en 256 MB.

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name db-cl-pg-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

Después de reiniciar el clúster de bases de datos, ejecute el siguiente comando para ver el estado de `stats_temp_directory`:

```
postgres=> SHOW stats_temp_directory;
```

El comando debe devolver lo siguiente:

```
stats_temp_directory
---------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

Administración de archivos temporales con PostgreSQL

En PostgreSQL, una consulta compleja puede realizar varias operaciones de ordenación y hash al mismo tiempo, y cada una de ellas utiliza memoria de la instancia para almacenar los resultados hasta el valor especificado en el parámetro `work_mem`. Cuando la memoria de la instancia no es suficiente, se crean archivos temporales para almacenar los resultados. Se escriben en el disco para completar la ejecución de la consulta. Posteriormente, una vez finalizada la

consulta, estos archivos se eliminan automáticamente. En En Aurora PostgreSQL, estos archivos comparten el almacenamiento local con otros archivos de registro. Puede supervisar el espacio de almacenamiento local de su clúster de base de datos de Aurora PostgreSQL observando la métrica de Amazon CloudWatch para `FreeLocalStorage`. Para obtener más información, consulte Troubleshoot local storage issues (Solución de problemas del almacenamiento local).

Recomendamos utilizar clústeres de lecturas optimizadas para Aurora para las cargas de trabajo que implican múltiples consultas simultáneas que aumentan el uso de archivos temporales. Estos clústeres utilizan almacenamiento por bloques local basado en unidades de estado sólido (SSD) de memoria rápida no volátil (NVMe) para colocar los archivos temporales. Para obtener más información, consulte Mejora del rendimiento de las consultas de Aurora PostgreSQL con lecturas optimizadas de Aurora .

Puede utilizar los siguientes parámetros y funciones para administrar los archivos temporales de la instancia.

- **`temp_file_limit`**: este parámetro cancela cualquier consulta que supere el tamaño de temp_files en KB. Este límite evita que cualquier consulta se ejecute de forma indefinida y consuma espacio en disco con archivos temporales. Puede calcular el valor utilizando los resultados del parámetro `log_temp_files`. Como práctica recomendada, examine el comportamiento de la carga de trabajo y establezca el límite de acuerdo con la estimación. En el siguiente ejemplo, se cancela una consulta cuando se supera el límite.

  ```
  postgres=>select * from pgbench_accounts, pg_class, big_table;
  ```

  ```
  ERROR: temporary file size exceeds temp_file_limit (64kB)
  ```

- **`log_temp_files`**: este parámetro envía mensajes a postgresql.log cuando se eliminan los archivos temporales de una sesión. Este parámetro produce registros después de que la consulta se complete correctamente. Por lo tanto, puede que no ayude a solucionar problemas de consultas activas y de larga ejecución.

  El ejemplo siguiente muestra que, cuando la consulta se completa correctamente, las entradas se registran en el archivo postgresql.log y se limpian los archivos temporales.

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir**: esta función que está disponible desde RDS para PostgreSQL 13 y versiones posteriores proporciona visibilidad sobre el uso actual de los archivos temporales. La consulta completada no aparece en los resultados de la función. En el siguiente ejemplo, puede ver los resultados de esta función.

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;

query
-------------------------------------------------------------------------------------
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

El nombre del archivo incluye el ID de procesamiento (PID) de la sesión que generó el archivo temporal. Una consulta más avanzada, como en el ejemplo siguiente, realiza una suma de los archivos temporales de cada PID.

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements**: si activa el parámetro pg_stat_statements, puede ver el uso medio de archivos temporales por llamada. Puede identificar el query_id de la consulta y usarlo para examinar el uso de archivos temporales, como se muestra en el siguiente ejemplo.

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
```

```
----------------------+---------------------------+-------+------------------------
+---------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                 239226 |
                      388678
(1 row)
```

- **Performance Insights**: en el panel de Información sobre el rendimiento, puede ver el uso temporal de los archivos activando las métricas temp_bytes y temp_files. A continuación, puede ver la media de estas dos métricas y cómo se corresponden con la carga de trabajo de la consulta. La vista de Información sobre el rendimiento no muestra específicamente las consultas que generan los archivos temporales. Sin embargo, al combinar Información sobre el rendimiento con la consulta que se muestra para `pg_ls_tmpdir`, puede solucionar problemas, realizar análisis y determinar los cambios en la carga de trabajo de la consulta.

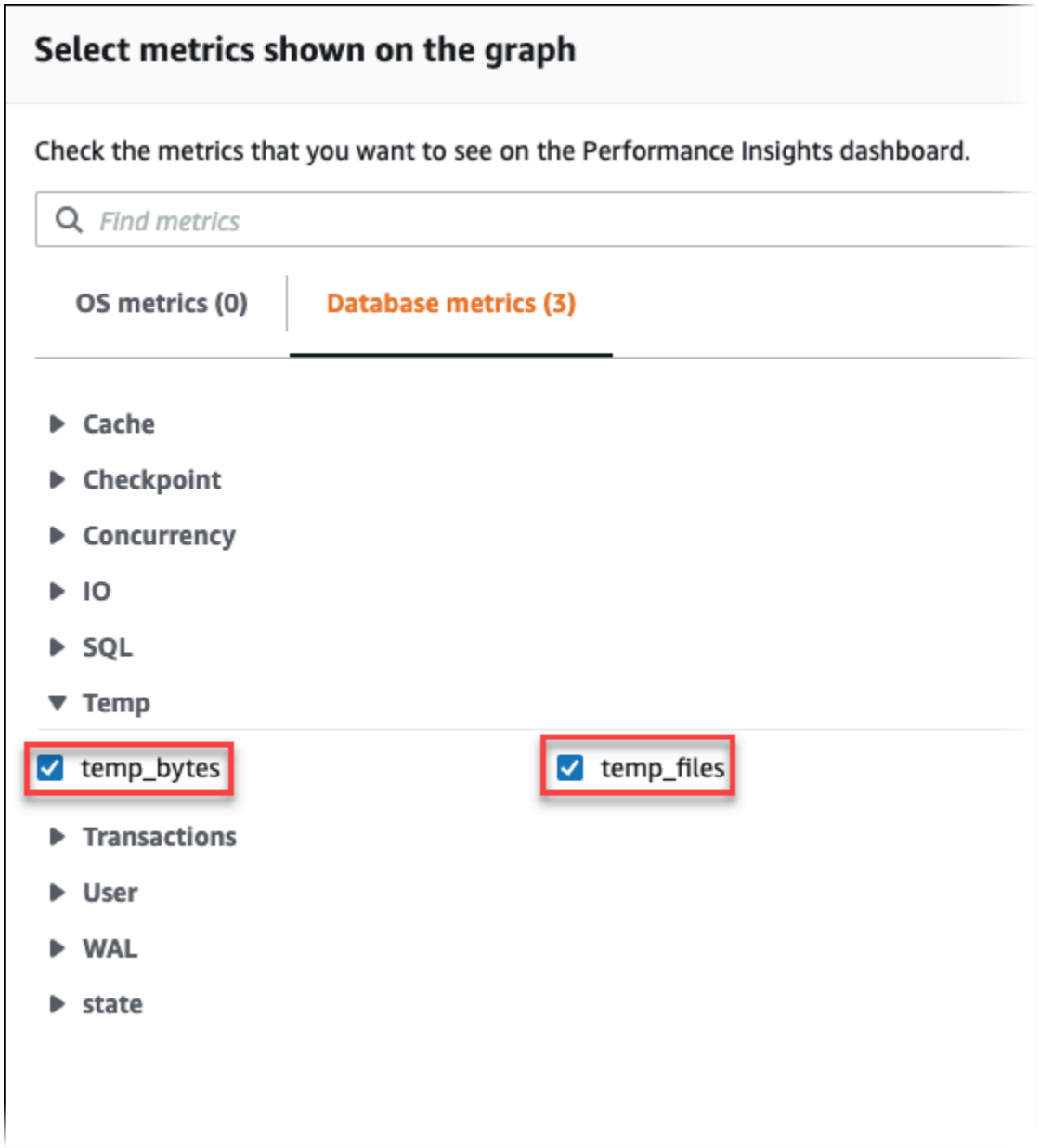
  Para obtener más información sobre cómo analizar métricas y consultas con Información de rendimiento, consulte Análisis de métricas mediante el panel de Información sobre rendimiento.

  Para ver un ejemplo sobre la visualización del uso de archivos temporales con Información de rendimiento, consulte Visualización del uso de archivos temporales con Información de rendimiento

Visualización del uso de archivos temporales con Información de rendimiento

Puede usar Información de rendimiento para consultar el uso de archivos temporales activando las métricas temp_bytes y temp_files. En Información de rendimiento, la vista no muestra las consultas específicas que generan archivos temporales; sin embargo, si combina Información de rendimiento con la consulta mostrada para `pg_ls_tmpdir`, puede solucionar problemas, realizar análisis y determinar cuáles son los cambios necesarios en la carga de trabajo de consultas.

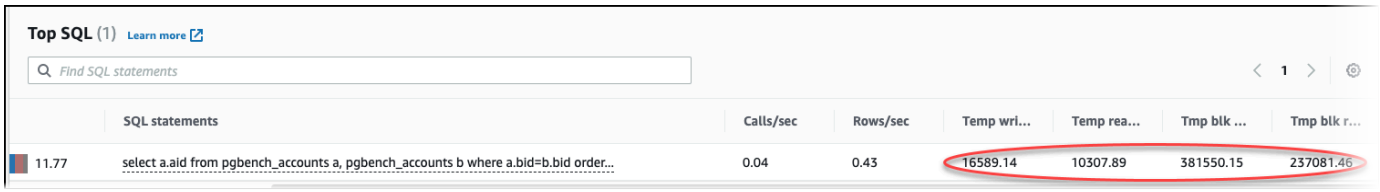
1. En el panel de Información sobre el rendimiento, elija Administrar métricas.
2. Elija las Métricas de la base de datos y seleccione las métricas temp_bytes y temp_files como se muestra en la siguiente captura de pantalla.



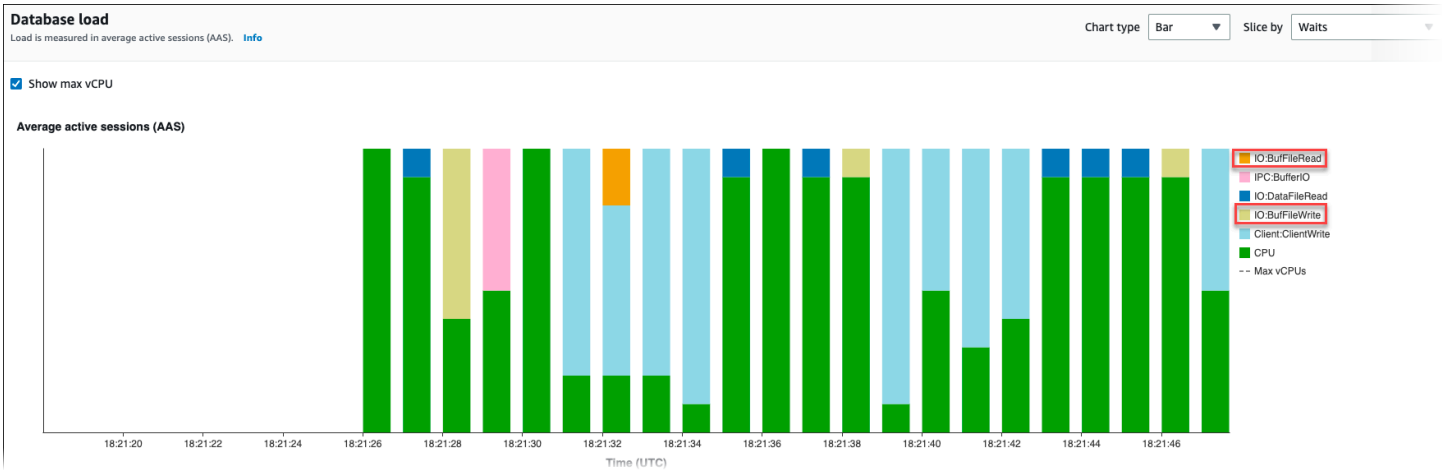
3. En la pestaña Principales SQL, seleccione el icono Preferencias.
4. En la ventana Preferencias, active las siguientes estadísticas para que aparezcan en la pestaña Principales SQL y seleccione Continuar.

   - Escrituras temporales por segundo
   - Lecturas temporales por segundo
   - Escritura temporal en bloque por llamada
   - Lectura temporal en bloque por llamada

5. El archivo temporal se divide cuando se combina con la consulta mostrada para `pg_ls_tmpdir`, como se observa en el siguiente ejemplo.



Los eventos `IO:BufFileRead` y `IO:BufFileWrite` se producen cuando las consultas principales de la carga de trabajo crean archivos temporales a menudo. Puede utilizar la Información de rendimiento para identificar las principales consultas pendientes en `IO:BufFileRead` e `IO:BufFileWrite` mediante la revisión del promedio de sesiones activas (AAS) en las secciones de carga de base de datos y SQL principales.



Para obtener más información sobre cómo analizar las consultas principales y cargar mediante eventos de espera con Información de Rendimiento, consulte Información general sobre la pestaña Top SQL (SQL principal). Debe identificar y ajustar las consultas que provocan el aumento del uso de archivos temporales y los eventos de espera relacionados. Para obtener más información sobre estos eventos de espera y su corrección, consulte IO:BufFileRead e IO:BufFileWrite.

> **Note**
>
> El parámetro `work_mem` controla cuándo se agota la memoria de la operación de ordenación y los resultados se escriben en archivos temporales. Se recomienda no cambiar la configuración de este parámetro por encima del valor predeterminado, ya que haría que cada sesión de base de datos consumiera más memoria. Además, una sola sesión que realiza combinaciones y ordenaciones complejas puede realizar operaciones paralelas en las que cada operación consume memoria.

Como práctica recomendada, cuando tenga un informe de gran tamaño con múltiples combinaciones y ordenaciones, defina este parámetro en el nivel de sesión mediante el comando `SET work_mem`. Por tanto, el cambio solo se aplica a la sesión actual y no cambia el valor globalmente.

Ajuste con eventos de espera de Aurora PostgreSQL

Los eventos de espera son una importante herramienta de ajuste para Aurora PostgreSQL. Si puede averiguar por qué las sesiones esperan recursos y qué están haciendo, podrá reducir mejor los cuellos de botella. Puede utilizar la información de esta sección para encontrar las posibles causas y acciones correctivas. Antes de profundizar en esta sección, le recomendamos encarecidamente que comprenda los conceptos básicos de Aurora, especialmente los siguientes temas:

- Almacenamiento de Amazon Aurora
- Administración del rendimiento y el escalado para clústeres de base de datos Aurora

Important

Los eventos de espera en esta sección son específicos de Aurora PostgreSQL. Utilice la información de esta sección solo para ajustar Amazon Aurora, no RDS for PostgreSQL. Algunos eventos de espera en esta sección no tienen análogos en las versiones de código abierto de estos motores de base de datos. Otros eventos de espera tienen los mismos nombres que los eventos en los motores de código abierto, pero se comportan de forma diferente. Por ejemplo, el almacenamiento de Amazon Aurora funciona de forma diferente al almacenamiento de código abierto, por lo que los eventos de espera relacionados con el almacenamiento indican condiciones de recursos diferentes.

Temas

- Conceptos esenciales para el ajuste de Aurora PostgreSQL
- Eventos de espera de Aurora PostgreSQL
- Client:ClientRead
- Client:ClientWrite
- CPU

- IO:BufFileRead y IO:BufFileWrite
- IO:DataFileRead
- IO:XactSync
- IPC:DamRecordTxAck
- Eventos de espera IPC:parallel
- IPC:ProcArrayGroupUpdate
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:buffer_content (BufferContent)
- LWLock:buffer_mapping
- LWLock:BufferIO (IPC:BufferIO)
- LWLock:lock_manager
- LWLock:MultiXact
- LWLock:pg_stat_statements
- Timeout:PgSleep

Conceptos esenciales para el ajuste de Aurora PostgreSQL

Antes de ajustar la base de datos Aurora PostgreSQL, asegúrese de saber qué son los eventos de espera y por qué se producen. También revise la arquitectura básica de memoria y disco de Aurora PostgreSQL. Para ver un diagrama de arquitectura útil, consulte el wikibook de PostgreSQL.

Temas

- Eventos de espera de Aurora PostgreSQL
- Memoria de Aurora PostgreSQL
- Procesos de Aurora PostgreSQL

Eventos de espera de Aurora PostgreSQL

Un evento de espera indica un recurso por el cual una sesión está en espera. Por ejemplo, el evento de espera `Client:ClientRead` ocurre cuando Aurora PostgreSQL espera recibir datos del cliente. Los recursos típicos por los que espera una sesión son los siguientes:

- Acceso de subproceso único a un búfer, por ejemplo, cuando una sesión intenta modificar un búfer
- Una fila bloqueada actualmente por otra sesión
- Lectura de un archivo de datos
- Escritura de un archivo de registro

Por ejemplo, para satisfacer una consulta, la sesión podría hacer un escaneo de tabla completo. Si los datos ya no están en la memoria, la sesión espera a que se complete la E/S del disco. Cuando los búferes se leen en la memoria, es posible que la sesión tenga que esperar porque otras sesiones tienen acceso a los mismos búferes. La base de datos registra las esperas mediante un evento de espera predefinido. Estos eventos se agrupan en categorías.

Un evento de espera no muestra por sí solo un problema de rendimiento. Por ejemplo, si los datos solicitados no están en memoria, es necesario leer los datos del disco. Si una sesión bloquea una fila para una actualización, otra sesión espera a que se desbloquee la fila para poder actualizarla. Una confirmación requiere un tiempo de espera para que se complete la escritura en un archivo de registro. Las esperas forman parte del funcionamiento normal de una base de datos.

Un gran número de eventos de espera suele mostrar un problema de rendimiento. En estos casos, se pueden utilizar los datos de los eventos de espera para determinar en qué se gastan las sesiones. Por ejemplo, si un informe que normalmente se ejecuta en minutos ahora se ejecuta durante horas, puede identificar los eventos de espera que más contribuyen al tiempo total de espera. Si puede determinar las causas de los principales eventos de espera, a veces puede hacer cambios que mejoren el rendimiento. Por ejemplo, si la sesión se encuentra a la espera de una fila que ha sido bloqueada por otra sesión, puede terminar la sesión de bloqueo.

Memoria de Aurora PostgreSQL

La memoria de Aurora PostgreSQL se divide en compartida y local.

Temas

- Memoria compartida en Aurora PostgreSQL

- Memoria local en Aurora PostgreSQL

Memoria compartida en Aurora PostgreSQL

Aurora PostgreSQL asigna memoria compartida cuando se inicia la instancia. La memoria compartida se divide en múltiples subáreas. A continuación se describen las más importantes.

Temas

- Búferes compartidos
- Búferes de registro de escritura anticipada (WAL)

Búferes compartidos

El grupo de búferes compartidos es un área de memoria de Aurora PostgreSQL que contiene todas las páginas que están o han sido utilizadas por las conexiones de la aplicación. Una página es la versión de memoria de un bloque de disco. El grupo de búferes compartidos almacena en caché los bloques de datos leídos desde el disco. El grupo reduce la necesidad de volver a leer los datos del disco, lo que hace que la base de datos funcione de forma más eficiente.

Cada tabla e índice se almacena como una matriz de páginas de tamaño fijo. Cada bloque contiene varias tuplas, que corresponden a filas. Una tupla se puede almacenar en cualquier página.

El grupo de búferes compartidos tiene memoria finita. Si una nueva solicitud requiere una página que no está en la memoria, y no hay más memoria, Aurora PostgreSQL desaloja una página que se utiliza con menos frecuencia para satisfacer la solicitud. La política de expulsión se implementa mediante un algoritmo de barrido de reloj.

El parámetro `shared_buffers` determina la cantidad de memoria que el servidor dedica al almacenamiento en caché de los datos.

Búferes de registro de escritura anticipada (WAL)

Un búfer del registro de escritura anticipada (WAL) contiene datos de transacciones que Aurora PostgreSQL escribe posteriormente en el almacenamiento persistente. Con el mecanismo WAL, Aurora PostgreSQL puede hacer lo siguiente:

- Recuperar datos después de un error
- Reducir la E/S del disco al evitar las escrituras frecuentes en el disco

Cuando un cliente cambia los datos, Aurora PostgreSQL escribe los cambios en el búfer WAL. Cuando el cliente emite un `COMMIT`, el proceso de escritura WAL escribe los datos de la transacción en el archivo WAL.

El parámetro `wal_level` determina la cantidad de información que se escribe en el WAL.

Memoria local en Aurora PostgreSQL

Cada proceso de backend asigna memoria local para el procesamiento de consultas.

Temas

- Área de memoria de trabajo
- Área de memoria de trabajo de mantenimiento
- Área de búfer temporal

Área de memoria de trabajo

El área de memoria de trabajo contiene datos temporales para las consultas que ejecutan ordenaciones y hashes. Por ejemplo, una consulta con una cláusula `ORDER BY` ejecuta una ordenación. Las consultas utilizan tablas hash en uniones hash y agregaciones.

El parámetro `work_mem` indica la cantidad de memoria que se utilizará en las operaciones internas de ordenación y en las tablas hash antes de escribir en los archivos temporales del disco. El valor predeterminado es 4 MB. Se pueden ejecutar varias sesiones simultáneamente, y cada sesión puede ejecutar operaciones de mantenimiento en paralelo. Por esta razón, la memoria de trabajo total utilizada puede ser múltiplo del parámetro `work_mem`.

Área de memoria de trabajo de mantenimiento

El área de memoria de trabajo de mantenimiento almacena en caché los datos de las operaciones de mantenimiento. Estas operaciones incluyen el vaciado, creación de un índice y adición de claves externas.

El parámetro `maintenance_work_mem` especifica la cantidad máxima de memoria que se utilizará para las operaciones de mantenimiento. El valor predeterminado es 64 MB. Una sesión de base de datos solo puede ejecutar una operación de mantenimiento a la vez.

Área de búfer temporal

El área de búfer temporal almacena en caché las tablas temporales de cada sesión de la base de datos.

Cada sesión asigna búferes temporales según sea necesario hasta el límite que se especifique. Cuando finaliza la sesión, el servidor borra los búferes.

El parámetro `temp_buffers` establece el número máximo de búferes temporales utilizados por cada sesión. Antes del primer uso de las tablas temporales dentro de una sesión, puede cambiar el valor de `temp_buffers`.

Procesos de Aurora PostgreSQL

Aurora PostgreSQL utiliza varios procesos.

Temas

- Proceso de administrador de correos
- Procesos de backend
- Procesos en segundo plano

Proceso de administrador de correos

El proceso de administrador de correos es el primer proceso que se ejecuta cuando se inicia Aurora PostgreSQL. El proceso de administrador de correos tiene las siguientes funciones principales:

- Bifurcar y monitorear los procesos en segundo plano
- Recibir solicitudes de autenticación de los procesos cliente, y autenticarlos antes de permitir que la base de datos atienda las solicitudes

Procesos de backend

Si el administrador de correos autentica una solicitud de cliente, el administrador de correos bifurca un nuevo proceso de backend, también llamado proceso postgres. Un proceso cliente se conecta exactamente a un proceso backend. El proceso cliente y el proceso backend se comunican directamente sin la intervención del proceso de administrador de correos.

Procesos en segundo plano

El proceso de administrador de correos bifurca varios procesos que ejecutan distintas tareas de backend. Algunas de las más importantes son las siguientes:

- Escritor de WAL

  Aurora PostgreSQL escribe datos en el búfer WAL (registro de escritura anticipada) en los archivos de registro. El principio del registro por adelantado es que la base de datos no puede escribir los cambios en los archivos de datos hasta que la base de datos escriba los registros que describen esos cambios en el disco. El mecanismo de WAL reduce la E/S del disco y permite a Aurora PostgreSQL utilizar los registros para recuperar la base de datos en caso de error.

- Escritor en segundo plano

  Este proceso escribe de forma periódica las páginas sucias (modificadas) desde los búferes de memoria a los archivos de datos. Una página se vuelve sucia cuando un proceso de backend la modifica en la memoria.

- Daemon de autovacuum

  El daemon consta de lo siguiente:

  - El iniciador de autovacuum
  - Los procesos de trabajo de autovacuum

  Cuando autovacuum está activado busca las tablas en las que se ha insertado, actualizado o eliminado un número elevado de tuplas. El daemon tiene las siguientes responsabilidades:

  - Recuperar o reutilizar el espacio de disco ocupado por las filas actualizadas o eliminadas
  - Actualizar las estadísticas utilizadas por el planificador
  - Proteger contra la pérdida de datos antiguos debido al reinicio del ID de transacción

  La característica de autovacuum automatiza la ejecución de los comandos `VACUUM` y `ANALYZE`. `VACUUM` tiene las siguientes variantes: estándar y completo. El vacío estándar se ejecuta en paralelo con otras operaciones de la base de datos. `VACUUM FULL` requiere un bloqueo exclusivo sobre la tabla en la que se trabaja. Por lo tanto, no puede ejecutarse en paralelo con operaciones que acceden a la misma tabla. `VACUUM` crea una cantidad considerable de tráfico de E/S, lo que puede causar un bajo rendimiento para otras sesiones activas.

Eventos de espera de Aurora PostgreSQL

La siguiente tabla enumera los eventos de espera de Aurora PostgreSQL que suelen indicar problemas de rendimiento, y resume las causas más comunes y acciones correctivas. Los siguientes eventos de espera son un subconjunto de la lista de Eventos de espera de Amazon Aurora PostgreSQL.

Evento de espera	Definición
Client:ClientRead	Este evento se produce cuando Aurora PostgreSQL espera recibir datos del cliente.
Client:ClientWrite	Este evento se produce cuando Aurora PostgreSQL espera escribir datos en el cliente.
CPU	Este evento ocurre cuando un subproceso está activo en la CPU o espera por la CPU.
IO:BufFileRead y IO:BufFileWrite	Estos eventos ocurren cuando Aurora PostgreSQL crea archivos temporales.
IO:DataFileRead	Este evento ocurre cuando una conexión espera en un proceso backend para leer una página requerida desde el almacenamiento porque la página no está disponible en la memoria compartida.
IO:XactSync	Este evento ocurre cuando la base de datos espera que el subsistema de almacenamiento de Aurora acepte la confirmación de una transacción regular, o la confirmación o restauración de una transacción preparada.
IPC:DamRecordTxAck	Este evento ocurre cuando Aurora PostgreSQL en una sesión que utiliza flujos de actividad de la base de datos genera un evento de transmisión de actividad, y luego espera que ese evento se vuelva permanente.

Evento de espera	Definición
Lock:advisory	Este evento ocurre cuando una aplicación PostgreSQL utiliza un bloqueo para coordinar la actividad en varias sesiones.
Lock:extend	Este evento ocurre cuando un proceso backend espera bloquear una relación para ampliarla mientras otro proceso tiene un bloqueo en esa relación para el mismo propósito.
Lock:Relation	Este evento ocurre cuando una consulta espera adquirir un bloqueo en una tabla o vista que está actualmente bloqueada por otra transacción.
Lock:transactionid	Este evento ocurre cuando una transacción espera un bloqueo a nivel de fila.
Lock:tuple	Este evento ocurre cuando un proceso de backend espera adquirir un bloqueo en una tupla.
LWLock:buffer_content (BufferContent)	Este evento ocurre cuando una sesión espera leer o escribir una página de datos en la memoria mientras otra sesión bloquea esa página para la escritura.
LWLock:buffer_mapping	Este evento se produce cuando una sesión espera asociar un bloque de datos con un búfer en el grupo de búferes compartidos.
LWLock:BufferIO (IPC:BufferIO)	Este evento ocurre cuando Aurora PostgreSQL o RDS for PostgreSQL espera que otros procesos terminen sus operaciones de entrada/salida (E/S) cuando intentan acceder a una página de forma simultánea.

Evento de espera	Definición
LWLock:lock_manager	Este evento ocurre cuando el motor de Aurora PostgreSQL mantiene el área de memoria del bloqueo compartido para asignar, verificar y desasignar un bloqueo cuando no es posible un bloqueo de ruta rápida.
LWLock:MultiXact	Este tipo de evento se produce cuando Aurora PostgreSQL mantiene una sesión abierta para completar varias transacciones que involucran la misma fila de una tabla. El evento de espera indica qué aspecto del procesamiento de múltiples transacciones está generando el evento de espera, es decir, LWLock:MultiXactOffsetSLRU, LWLock:MultiXactOffsetBuffer, LWLock:MultiXactMemberSLRU o LWLock:MultiXactMemberBuffer.
Timeout:PgSleep	Este evento ocurre cuando un proceso del servidor llamó a la función `pg_sleep` y espera que el tiempo de espera expire.

Client:ClientRead

El evento `Client:ClientRead` ocurre cuando Aurora PostgreSQL espera recibir datos del cliente.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera se admite para Aurora PostgreSQL versión 10 y posterior.

Context

Un clúster de base de datos de Aurora PostgreSQL espera recibir datos del cliente. El clúster de la base de datos de Aurora PostgreSQL tiene que recibir los datos del cliente antes de poder enviar más datos al cliente. El tiempo que el clúster espera antes de recibir los datos del cliente es un evento `Client:ClientRead`.

Causas probables del aumento de las esperas

Las causas más comunes para que el evento `Client:ClientRead` aparezca en el máximo de esperas son las siguientes:

Aumento de la latencia de la red

Puede haber un aumento de la latencia de la red entre el clúster de la base de datos PostgreSQL de Aurora y el cliente. Una mayor latencia de red aumenta el tiempo necesario para que el clúster de la base de datos reciba los datos del cliente.

Aumento de la carga en el cliente

Puede haber presión de la CPU o saturación de la red en el cliente. Un aumento de la carga en el cliente puede retrasar la transmisión de datos desde el cliente al clúster de la base de datos de Aurora PostgreSQL.

Excesivos viajes de ida y vuelta de la red

Un gran número de viajes de ida y vuelta de la red entre el clúster de la base de datos de Aurora PostgreSQL y el cliente puede retrasar la transmisión de datos del cliente al clúster de la base de datos de Aurora PostgreSQL.

Operación de copia grande

Durante una operación de copia, los datos se transfieren desde el sistema de archivos del cliente al clúster de la base de datos de Aurora PostgreSQL. El envío de una gran cantidad de datos al clúster de la base de datos puede retrasar la transmisión de datos del cliente al clúster de la base de datos.

Conexión de cliente inactivo

Una conexión a una instancia de base de datos de Aurora PostgreSQL está inactiva en el estado de transacción y espera a que un cliente envíe más datos o emita un comando. Este estado puede resultar en un aumento de eventos `Client:ClientRead`.

PgBouncer se utiliza para la agrupación de conexiones

PgBouncer tiene un ajuste de configuración de red de bajo nivel llamado `pkt_buf`, que se establece en 4.096 de forma predeterminada. Si la carga de trabajo envía paquetes de consulta de más de 4096 bytes a través de PgBouncer, recomendamos aumentar la configuración de `pkt_buf` a 8192. Si la nueva configuración no disminuye el número de eventos `Client:ClientRead`, recomendamos aumentar la configuración de `pkt_buf` a valores mayores, como 16 384 o 32 768. Si el texto de la consulta es grande, el ajuste más grande puede ser particularmente útil.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Coloque los clientes en la misma zona de disponibilidad y subred VPC que el clúster
- Escale el cliente
- Utilizar las instancias de generación actual
- Aumentar el ancho de banda de la red
- Monitorear los máximos de rendimiento de la red
- Monitorear las transacciones en el estado “inactivo en la transacción”

Coloque los clientes en la misma zona de disponibilidad y subred VPC que el clúster

Para reducir la latencia de la red y aumentar el rendimiento de la red, coloque los clientes en la misma zona de disponibilidad y subred de nube virtual privada (VPC) que el clúster de la base de datos de Aurora PostgreSQL. Asegúrese de que los clientes estén lo más cerca geográficamente posible del clúster de la base de datos.

Escale el cliente

Con Amazon CloudWatch u otras métricas del anfitrión, determine si su cliente está actualmente limitado por la CPU o el ancho de banda de la red, o ambos. Si el cliente está restringido, escale su cliente en forma adecuada.

Utilizar las instancias de generación actual

En algunos casos, es posible que no utilice una clase de instancia de base de datos que admita tramas gigantes. Si ejecuta la aplicación en Amazon EC2, considere la posibilidad de utilizar una instancia de generación actual para el cliente. Además, configure la unidad de transmisión máxima (MTU) en el sistema operativo del cliente. Esta técnica podría reducir el número de viajes de ida y vuelta de la red y aumentar el rendimiento de la red. Para obtener más información, consulte Tramas gigantes (MTU 9001) en la Guía del usuario de Amazon EC2.

Para obtener información acerca de las clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora. Para determinar la clase de instancia de base de datos que equivale a un tipo de instancia de Amazon EC2, coloque `db.` antes del nombre del tipo de instancia de Amazon EC2. Por ejemplo, la instancia de Amazon EC2 `r5.8xlarge` equivale a la clase de instancia de base de datos `db.r5.8xlarge`.

Aumentar el ancho de banda de la red

Utilice las métricas de Amazon CloudWatch de `NetworkReceiveThroughput` y `NetworkTransmitThroughput` para monitorear el tráfico de red entrante y saliente en el clúster de la base de datos. Estas métricas pueden ayudarle a determinar si el ancho de banda de la red es suficiente para su carga de trabajo.

Si el ancho de banda de su red no es suficiente, auméntelo. Si el cliente de AWS o la instancia de base de datos alcanza los límites del ancho de banda de la red, la única forma de aumentar el ancho de banda es aumentar el tamaño de la instancia de base de datos.

Para obtener más información acerca de las métricas de CloudWatch, consulte Métricas de Amazon CloudWatch para Amazon Aurora.

Monitorear los máximos de rendimiento de la red

Si utiliza clientes de Amazon EC2, Amazon EC2 proporciona límites máximos para las métricas de rendimiento de la red, incluido el ancho de banda de red entrante y saliente agregado. También proporciona un seguimiento de la conexión para garantizar que los paquetes se devuelven como se espera y el acceso a los servicios de enlace local para servicios como el sistema de nombres de dominio (DNS). Para monitorear estos máximos, utilice un controlador de red mejorado actual y monitoree el rendimiento de la red para su cliente.

Para obtener más información, consulte Monitorear el rendimiento de la red de la instancia de Amazon EC2 en la Guía del usuario de Amazon EC2 y Monitorear el rendimiento de la red de la instancia de Amazon EC2 en la Guía del usuario de Amazon EC2.

Monitorear las transacciones en el estado “inactivo en la transacción”

Verifique si tiene un número creciente de conexiones `idle in transaction`. Para ello, monitoree la columna `state` en la tabla `pg_stat_activity`. Es posible que pueda identificar el origen de la conexión si ejecuta una consulta similar a la siguiente.

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
order by 3 desc
```

Client:ClientWrite

El evento `Client:ClientWrite` ocurre cuando Aurora PostgreSQL espera escribir datos en el cliente.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera se admite para Aurora PostgreSQL versión 10 y posterior.

Context

Un proceso cliente debe leer todos los datos recibidos de un clúster de la base de datos de Aurora PostgreSQL antes de que el clúster pueda enviar más datos. El tiempo que el clúster espera antes de enviar más datos al cliente es un evento `Client:ClientWrite`.

La reducción del rendimiento de la red entre el clúster de base de datos de Aurora PostgreSQL y el cliente puede causar este evento. La presión de la CPU y la saturación de la red en el cliente también pueden causar este evento. La presión de la CPU es cuando la CPU se utiliza por completo y hay tareas esperando por el tiempo de la CPU. La saturación de la red es cuando la red entre la base de datos y el cliente transporta más datos de los que puede manejar.

Causas probables del aumento de las esperas

Las causas más comunes para que el evento `Client:ClientWrite` aparezca en el máximo de esperas son las siguientes:

Aumento de la latencia de la red

Puede haber un aumento de la latencia de la red entre el clúster de la base de datos PostgreSQL de Aurora y el cliente. Una mayor latencia de la red aumenta el tiempo necesario para que el cliente reciba los datos.

Aumento de la carga en el cliente

Puede haber presión de la CPU o saturación de la red en el cliente. Un aumento de la carga en el cliente retrasa la recepción de los datos del clúster de la base de datos de Aurora PostgreSQL.

Gran volumen de datos enviados al cliente

El clúster de la base de datos Aurora PostgreSQL se encuentra enviando una gran cantidad de datos al cliente. Es posible que el cliente no pueda recibir los datos tan rápido como el clúster los envía. Actividades como una copia de una tabla grande pueden resultar en un aumento de eventos `Client:ClientWrite`.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Coloque los clientes en la misma zona de disponibilidad y subred VPC que el clúster
- Utilice las instancias de generación actual
- Reducir la cantidad de datos enviados al cliente
- Escale el cliente

Coloque los clientes en la misma zona de disponibilidad y subred VPC que el clúster

Para reducir la latencia de la red y aumentar el rendimiento de la red, coloque los clientes en la misma zona de disponibilidad y subred de nube virtual privada (VPC) que el clúster de la base de datos de Aurora PostgreSQL.

Utilice las instancias de generación actual

En algunos casos, es posible que no utilice una clase de instancia de base de datos que admita tramas gigantes. Si ejecuta la aplicación en Amazon EC2, considere la posibilidad de utilizar una instancia de generación actual para el cliente. Además, configure la unidad de transmisión máxima (MTU) en el sistema operativo del cliente. Esta técnica podría reducir el número de viajes de ida y vuelta de la red y aumentar el rendimiento de la red. Para obtener más información, consulte Tramas gigantes (MTU 9001) en la Guía del usuario de Amazon EC2.

Para obtener información acerca de las clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora. Para determinar la clase de instancia de base de datos que equivale a un tipo de instancia de Amazon EC2, coloque `db.` antes del nombre del tipo de instancia de Amazon EC2. Por ejemplo, la instancia de Amazon EC2 `r5.8xlarge` equivale a la clase de instancia de base de datos `db.r5.8xlarge`.

Reducir la cantidad de datos enviados al cliente

Cuando sea posible, ajuste la aplicación para reducir la cantidad de datos que el clúster de la base de datos de Aurora PostgreSQL envía al cliente. Hacer estos ajustes reduce la contención de la CPU y de la red en el cliente.

Escale el cliente

Con Amazon CloudWatch u otras métricas del anfitrión, determine si su cliente está actualmente limitado por la CPU o el ancho de banda de la red, o ambos. Si el cliente está restringido, escale su cliente en forma adecuada.

CPU

Este evento ocurre cuando un subproceso está activo en la CPU o espera por la CPU.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

La información del evento de espera es relevante para Aurora PostgreSQL versión 9.6 y posterior.

Context

La unidad de procesamiento central (CPU) es el componente de un ordenador que ejecuta instrucciones. Por ejemplo, las instrucciones de la CPU hacen operaciones aritméticas e intercambian datos en la memoria. Si una consulta aumenta el número de instrucciones que ejecuta a través del motor de base de datos, aumenta el tiempo de ejecución de la consulta. La programación de la CPU consiste en dar tiempo de CPU a un proceso. La programación es orquestada por el núcleo del sistema operativo.

Temas

- Cómo saber cuándo se produce esta espera
- Métrica de DBloadCPU
- Métrica os.cpuUtilization
- Causa probable de la programación de la CPU

Cómo saber cuándo se produce esta espera

Este evento de espera de CPU indica que un proceso del backend se encuentra activo en la CPU o en espera de la misma. Se sabrá que sucede cuando una consulta muestre la siguiente información:

- La columna `pg_stat_activity.state` tiene el valor `active`.
- Las columnas `wait_event_type` y `wait_event` en `pg_stat_activity` son `null`.

Para ver los procesos del backend que se encuentran en uso o en espera de CPU, ejecute la siguiente consulta.

```
SELECT *
FROM   pg_stat_activity
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

Métrica de DBloadCPU

La métrica de Información sobre rendimiento para la CPU es `DBLoadCPU`. El valor de `DBLoadCPU` puede diferir del valor de la métrica `CPUUtilization` de Amazon CloudWatch. Esta última métrica se recopila del hipervisor para una instancia de base de datos.

Métrica os.cpuUtilization

Las métricas del sistema operativo de Información sobre rendimiento proporcionan información detallada sobre la utilización de la CPU. Por ejemplo, puede mostrar las siguientes métricas:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Información sobre rendimiento informa del uso de la CPU por parte del motor de base de datos como `os.cpuUtilization.nice.avg`.

Causa probable de la programación de la CPU

Desde la perspectiva del sistema operativo, la CPU se encuentra activa cuando no está ejecutando el subproceso inactivo. La CPU está activa mientras ejecuta un cálculo, pero también se activa cuando espera en la E/S de la memoria. Este tipo de E/S domina la carga de trabajo típica de una base de datos.

Es probable que los procesos esperen a que se programe una CPU cuando se cumplen las siguientes condiciones:

- La métrica CloudWatch `CPUUtilization` está cerca del 100 por ciento.
- La carga media es mayor que el número de vCPU, lo que indica una carga pesada. Puede encontrar la métrica `loadAverageMinute` en la sección de métricas del sistema operativo en Información sobre rendimiento.

Causas probables del aumento de las esperas

Cuando el evento de espera de la CPU ocurre más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas pueden ser las siguientes.

Temas

- Causas probables de picos repentinos
- Causas probables de alta frecuencia prolongada
- Casos aislados

Causas probables de picos repentinos

Las causas más probables de picos repentinos son las siguientes:

- La aplicación abrió demasiadas conexiones simultáneas a la base de datos. Este escenario se conoce como “tormenta de conexiones”
- La carga de trabajo de la aplicación ha cambiado de alguna de las siguientes maneras:
  - Nuevas consultas
  - Un aumento del tamaño del conjunto de datos
  - Mantenimiento o creación de índices
  - Nuevas funciones
  - Nuevos operadores
  - Aumento de la ejecución de consultas en paralelo
- Los planes de ejecución de sus consultas han cambiado. En algunos casos, un cambio puede provocar un aumento de los búferes. Por ejemplo, la consulta utiliza ahora un escaneo secuencial cuando antes utilizaba un índice. En este caso, las consultas necesitan más CPU para lograr el mismo objetivo.

Causas probables de alta frecuencia prolongada

Las causas más probables de eventos que se repiten durante un periodo prolongado:

- Demasiados procesos de backend se ejecutan de forma simultánea en la CPU. Estos procesos pueden llegar a ser procesos de trabajo paralelos.
- Las consultas tienen un rendimiento subóptimo porque necesitan un gran número de búferes.

Casos aislados

Si ninguna de las causas probables resulta ser la causa real, es posible que se produzcan las siguientes situaciones:

- La CPU está intercambiando procesos de entrada y salida.
- El cambio de contexto de CPU ha aumentado.
- El código de Aurora PostgreSQL no tiene eventos de espera.

Acciones

Si el evento de espera de CPU domina la actividad de la base de datos, no indica necesariamente un problema de rendimiento. Responda a este evento solo cuando el rendimiento se deteriore.

Temas

- Investigue si la base de datos es la causa del aumento de la CPU
- Determine si el número de conexiones aumentó
- Responder a los cambios en la carga de trabajo

Investigue si la base de datos es la causa del aumento de la CPU

Examine la métrica `os.cpuUtilization.nice.avg` en Información sobre rendimiento. Si este valor es mucho menor que el uso de la CPU, los procesos ajenos a la base de datos son los que más contribuyen a la CPU.

Determine si el número de conexiones aumentó

Examine la métrica `DatabaseConnections` en Amazon CloudWatch. La acción a tomar depende de si el número aumentó o disminuyó durante el periodo de aumento de los eventos de espera de la CPU.

Las conexiones aumentaron

Si el número de conexiones aumentó, compare el número de procesos de backend que consumen CPU con el número de vCPU. Los siguientes escenarios son posibles:

- El número de procesos de backend que consumen CPU es menor que el número de vCPU.

  En este caso, el número de conexiones no es un problema. Sin embargo, puede intentar reducir la utilización de la CPU.
- El número de procesos de backend que consumen CPU es mayor que el número de vCPU.

  En este caso, considere las siguientes opciones:

- Disminuya el número de procesos backend conectados a la base de datos. Por ejemplo, implemente una solución de agrupación de conexiones, como el proxy RDS. Para obtener más información, consulte Amazon RDS Proxy para Aurora.
- Actualice el tamaño de su instancia para obtener un mayor número de vCPU.
- Redirija algunas cargas de trabajo de solo lectura a nodos lectores, si procede.

Las conexiones no aumentaron

Examine las métricas de `blks_hit` en Información sobre rendimiento. Busque una correlación entre el aumento de `blks_hit` y el uso de la CPU. Los siguientes escenarios son posibles:

- El uso de la CPU y `blks_hit` están correlacionados.

  En este caso, encuentre las principales instrucciones SQL que están relacionadas con el uso de la CPU y busque los cambios de plan. Puede utilizar cualquiera de las siguientes técnicas:

  - Explicar los planes manualmente y compararlos con el plan de ejecución esperado.
  - Buscar un aumento en los aciertos de bloque por segundo y en los aciertos de bloque local por segundo. En la sección Top SQL (SQL principal) del panel de Información sobre rendimiento, elija Preferences (Preferencias).
- El uso de la CPU y `blks_hit` no están correlacionados.

  En este caso, determine si se produce alguna de las siguientes situaciones:

  - La aplicación se conecta y desconecta con rapidez de la base de datos.

    Diagnostique este comportamiento mediante la activación de `log_connections` y `log_disconnections`, y luego analice los registros de PostgreSQL. Considere utilizar el analizador de registros `pgbadger`. Para obtener más información, consulte https://github.com/darold/pgbadger.
  - El sistema operativo está sobrecargado.

    En este caso, Información sobre rendimiento muestra que los procesos del backend consumen la CPU durante más tiempo del habitual. Busque pruebas en las métricas de Información sobre rendimiento `os.cpuUtilization` o en la métrica CloudWatch `CPUUtilization`. Si el sistema operativo está sobrecargado, consulte las métricas de Monitoreo mejorado para hacer un diagnóstico más profundo. Específicamente, observe la lista de procesos y el porcentaje de CPU que consume cada proceso.

- Las instrucciones SQL más importantes son las que consumen demasiada CPU.

  Examine las instrucciones que se relacionan con el uso de la CPU para ver si pueden utilizar menos CPU. Ejecute un comando `EXPLAIN`, y céntrese en los nodos del plan que tienen el mayor impacto. Considere utilizar un visualizador de planes de ejecución de PostgreSQL. Para probar esta herramienta, consulte http://explain.dalibo.com/.

Responder a los cambios en la carga de trabajo

Si la carga de trabajo cambió, busque los siguientes tipos de cambios:

Nuevas consultas

Verifique si las nuevas consultas son las esperadas. Si es así, asegúrese de que los planes de ejecución y el número de ejecuciones por segundo son los esperados.

Aumento del tamaño del conjunto de datos

Determine si la partición, si no se ha implementado todavía, podría ayudar. Esta estrategia podrá reducir el número de páginas que debe recuperar una consulta.

Mantenimiento o creación de índices

Verifique si el programa de mantenimiento es el previsto. Una práctica recomendada es programar las actividades de mantenimiento fuera de los picos de actividad.

Nuevas funciones

Verifique si estas funciones se comportan como se espera durante las pruebas. En concreto, verifique si el número de ejecuciones por segundo es el esperado.

Nuevos operadores

Verifique si su rendimiento es el esperado durante las pruebas.

Aumento de la ejecución de consultas paralelas

Determine si se ha producido alguna de las siguientes situaciones:

- Las relaciones o los índices implicados han crecido repentinamente en tamaño de modo que difieren significativamente de `min_parallel_table_scan_size` o `min_parallel_index_scan_size`.
- Se hicieron cambios recientes en `parallel_setup_cost` o `parallel_tuple_cost`.

- Se hicieron cambios recientes en `max_parallel_workers` o `max_parallel_workers_per_gather`.

IO:BufFileRead y IO:BufFileWrite

Los eventos `IO:BufFileRead` e `IO:BufFileWrite` ocurren cuando Aurora PostgreSQL crea archivos temporales. Cuando las operaciones requieren más memoria de la que los parámetros de la memoria de trabajo definen actualmente, escriben datos temporales en el almacenamiento persistente. Esta operación se llama a veces derramamiento en el disco. Para obtener más información sobre los archivos temporales y su uso, consulte Administración de archivos temporales con PostgreSQL.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Contexto

`IO:BufFileRead` e `IO:BufFileWrite` se relacionan con el área de memoria de trabajo y el área de memoria de trabajo de mantenimiento. Para más información sobre estas áreas de memoria local, consulte Área de memoria de trabajo y Área de memoria de trabajo de mantenimiento.

El valor predeterminado de `work_mem` es de 4 MB. Si una sesión ejecuta operaciones en paralelo, cada proceso de trabajo que maneja el paralelismo utiliza 4 MB de memoria. Por esta razón, configure `work_mem` con cuidado. Si el valor es demasiado alto, una base de datos con muchas sesiones puede consumir demasiada memoria. Si establece el valor demasiado bajo, Aurora PostgreSQL crea archivos temporales en el almacenamiento local. La E/S del disco para estos archivos temporales puede reducir el rendimiento.

Si se observa la siguiente secuencia de eventos, es posible que la base de datos genere archivos temporales:

1. Disminución repentina y brusca de la disponibilidad
2. Recuperación rápida del espacio libre

También puede observar un patrón de “motosierra”. Este patrón puede indicar que la base de datos crea archivos pequeños de forma constante.

Causas probables del aumento de las esperas

En general, estos eventos de espera son causados por operaciones que consumen más memoria de la que asignan los parámetros `work_mem` o `maintenance_work_mem`. Para compensar, las operaciones se escriben en archivos temporales. Las causas más comunes de los eventos `IO:BufFileRead` y `IO:BufFileWrite` son las siguientes:

Consultas que necesitan más memoria de la que existe en la zona de memoria de trabajo

Las consultas con las siguientes características utilizan el área de memoria de trabajo:

- Combinaciones hash
- `ORDER BY`Cláusula
- `GROUP BY`Cláusula
- `DISTINCT`
- Funciones de ventana
- `CREATE TABLE AS SELECT`
- Actualización de la vista materializada

Instrucciones que necesitan más memoria de la que existe en el área de memoria de trabajo de mantenimiento

Las siguientes instrucciones utilizan el área de memoria de trabajo de mantenimiento:

- `CREATE INDEX`
- `CLUSTER`

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Identifique el problema
- Examine sus consultas de unión (join)
- Examinar las consultas ORDER BY y GROUP BY
- Evite utilizar la operación DISTINCT
- Considere la posibilidad de utilizar funciones de ventana en lugar de funciones GROUP BY
- Investigar las vistas materializadas y las instrucciones CTA
- Utilizar pg_repack al crear índices
- Aumentar maintenance_work_mem al hacer un clúster de tablas
- Ajustar la memoria para evitar IO:BufFileRead e IO:BufFileWrite

Identifique el problema

Puede ver el uso de archivos temporales directamente en información de rendimiento. Para obtener más información, consulte Visualización del uso de archivos temporales con Información de rendimiento. Cuando la información de rendimiento está desactivada, es posible que observe un aumento de las operaciones `IO:BufFileRead` e `IO:BufFileWrite`. Para solucionar problemas, haga lo siguiente:

1. Examine la métrica `FreeLocalStorage` en Amazon CloudWatch.
2. Busque un patrón de “motosierra”, que es una serie de picos irregulares.

Un patrón de motosierra indica un consumo y liberación rápidos de almacenamiento, a menudo asociados con archivos temporales. Si observa este patrón, active Información sobre rendimiento. Al utilizar Información sobre rendimiento, puede identificar cuándo se producen los eventos de espera y qué consultas están asociadas a ellos. La solución depende de la consulta específica que causa los eventos.

O establezca el parámetro `log_temp_files`. Este parámetro registra todas las consultas que generan más de un umbral de KB de archivos temporales. Si el valor es `0`, Aurora PostgreSQL registra todos los archivos temporales. Si el valor es `1024`, Aurora PostgreSQL registra todas

las consultas que producen archivos temporales de más de 1 MB. Para más información sobre `log_temp_files`, consulte Error Reporting and Logging en la documentación de PostgreSQL.

Examine sus consultas de unión (join)

Es probable que su aplicación utilice uniones. Por ejemplo, la siguiente consulta une cuatro tablas.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

Una posible causa de los picos de uso de archivos temporales es un problema en la propia consulta. Por ejemplo, una cláusula rota podría no filtrar las uniones correctamente. Considere la segunda unión interna en el siguiente ejemplo.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = customer.id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

La consulta anterior une por error `customer.id` con `customer.id`, lo que genera un producto cartesiano entre cada cliente y cada pedido. Este tipo de unión accidental genera grandes archivos temporales. Según el tamaño de las tablas, una consulta cartesiana puede incluso llenar el almacenamiento. Es posible que la aplicación tenga uniones cartesianas cuando se den las siguientes condiciones:

- Se observan grandes y bruscas disminuciones en la disponibilidad del almacenamiento, seguidas de una rápida recuperación.

- No se crean índices.
- No se emiten instrucciones `CREATE TABLE FROM SELECT`.
- No se actualizan las vistas materializadas.

Para ver si las tablas se unen con las claves adecuadas, inspeccione las directivas de consulta y de asignación objeto-relacional. Tenga en cuenta que algunas consultas de la aplicación no se llaman todo el tiempo, y algunas consultas se generan de forma dinámica.

Examinar las consultas ORDER BY y GROUP BY

En algunos casos, una cláusula `ORDER BY` puede dar lugar a un exceso de archivos temporales. Tenga en cuenta estas directrices:

- Incluya las columnas en una cláusula `ORDER BY` solo cuando sea necesario ordenarlas. Esta directriz es especialmente importante para las consultas que devuelven miles de filas y especifican muchas columnas en la cláusula `ORDER BY`.
- Considere la posibilidad de crear índices para acelerar las cláusulas `ORDER BY` cuando coincidan con columnas que tengan el mismo orden ascendente o descendente. Los índices parciales son preferibles porque son más pequeños. Los índices más pequeños se leen y recorren más rápidamente.
- Si crea índices para columnas que pueden aceptar valores nulos, considere si quiere que los valores nulos se almacenen al final o al principio de los índices.

  Si es posible, reduzca el número de filas que hay que ordenar, mediante el filtrado del conjunto de resultados. Si utiliza instrucciones de la cláusula `WITH` o subconsultas, recuerde que una consulta interna genera un conjunto de resultados y lo pasa a la consulta externa. Cuantas más filas pueda filtrar una consulta, menos tendrá que ordenar esta última.
- Si no necesita obtener el conjunto de resultados completo, utilice la cláusula `LIMIT`. Por ejemplo, si solo quiere las cinco primeras filas, una consulta que utilice la cláusula `LIMIT` no sigue generando resultados. De este modo, la consulta requiere menos memoria y archivos temporales.

Una consulta que utiliza una cláusula `GROUP BY` también puede requerir archivos temporales. Las consultas `GROUP BY` resumen los valores con funciones como las siguientes:

- `COUNT`
- `AVG`

- MIN
- MAX
- SUM
- STDDEV

Para ajustar las consultas GROUP BY, siga las recomendaciones para las consultas ORDER BY.

Evite utilizar la operación DISTINCT

Si es posible, evite utilizar la operación DISTINCT para eliminar las filas duplicadas. Cuantas más filas innecesarias y duplicadas devuelva la consulta, más cara será la operación DISTINCT. Si es posible, agregue filtros en la cláusula WHERE, incluso si utiliza los mismos filtros para diferentes tablas. Filtrar la consulta y unirla correctamente mejora su rendimiento y reduce el uso de recursos. Además, evita que los informes y resultados sean incorrectos.

Si necesita utilizar DISTINCT para varias filas de una misma tabla, considere la posibilidad de crear un índice compuesto. Agrupar varias columnas en un índice puede mejorar el tiempo de evaluación de las filas distintas. Además, si utiliza Amazon Aurora PostgreSQL versión 10 o posterior, puede correlacionar estadísticas entre varias columnas con el comando CREATE STATISTICS.

Considere la posibilidad de utilizar funciones de ventana en lugar de funciones GROUP BY

Al utilizar GROUP BY, se modifica el conjunto de resultados y luego se recupera el resultado agregado. Con las funciones de ventana, se agregan los datos sin cambiar el conjunto de resultados. Una función de ventana utiliza la cláusula OVER para efectuar cálculos a través de los conjuntos definidos por la consulta, correlacionando una fila con otra. Puede utilizar todas las funciones GROUP BY en las funciones de ventana, pero también puede utilizar funciones como las siguientes:

- RANK
- ARRAY_AGG
- ROW_NUMBER
- LAG
- LEAD

Para minimizar el número de archivos temporales generados por una función de ventana, elimine las duplicaciones para el mismo conjunto de resultados cuando necesite dos agregaciones distintas. Analice la siguiente consulta.

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

Puede volver a escribir la consulta con la cláusula `WINDOW` de la siguiente manera.

```
SELECT sum(salary) OVER w as sum_salary
         , avg(salary) OVER w as_avg_salary
    FROM empsalary
  WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

De forma predeterminada, el planificador de ejecución de Aurora PostgreSQL consolida nodos similares para no duplicar operaciones. Sin embargo, al utilizar una declaración explícita para el bloque de la ventana, puede actualizar la consulta más fácilmente. También puede mejorar el rendimiento al evitar la duplicación.

Investigar las vistas materializadas y las instrucciones CTA

Cuando una vista materializada se actualiza, ejecuta una consulta. Esta consulta puede contener una operación como `GROUP BY`, `ORDER BY` o `DISTINCT`. Durante una actualización, es posible que observe un gran número de archivos temporales y los eventos de espera `IO:BufFileWrite` e `IO:BufFileRead`. Del mismo modo, cuando se crea una tabla basada en una instrucción `SELECT`, la instrucción `CREATE TABLE` ejecuta una consulta. Para reducir los archivos temporales necesarios, optimice la consulta.

Utilizar pg_repack al crear índices

Cuando se crea un índice, el motor ordena el conjunto de resultados. A medida que las tablas aumentan de tamaño y los valores de la columna indexada se diversifican, los archivos temporales requieren más espacio. En la mayoría de los casos, no se puede evitar la creación de archivos temporales para tablas grandes sin modificar el área de memoria de trabajo de mantenimiento. Para obtener más información, consulte Área de memoria de trabajo de mantenimiento.

Una posible solución para recrear un índice grande es utilizar la herramienta pg_repack. Para más información, consulte Reorganize tables in PostgreSQL databases with minimal locks en la documentación de pg_repack.

Aumentar maintenance_work_mem al hacer un clúster de tablas

El comando `CLUSTER` hace un clúster de la tabla especificada por table_name basado en un índice existente especificado por index_name. Aurora PostgreSQL recrea físicamente la tabla para que coincida con el orden de un índice determinado.

Cuando el almacenamiento magnético era frecuente, los clústeres eran comunes porque el rendimiento del almacenamiento era limitado. Ahora que el almacenamiento basado en SSD es común, los clústeres son menos populares. Sin embargo, si se hacen clústeres en las tablas, se puede aumentar ligeramente el rendimiento en función del tamaño de la tabla, índice, consulta, etc.

Si ejecuta el comando `CLUSTER` y observa los eventos de espera `IO:BufFileWrite` e `IO:BufFileRead`, ajuste `maintenance_work_mem`. Aumente el tamaño de la memoria a una cantidad bastante grande. Un valor alto significa que el motor puede utilizar más memoria para la operación de clusterización.

Ajustar la memoria para evitar IO:BufFileRead e IO:BufFileWrite

En algunas situaciones, es necesario ajustar la memoria. El objetivo es equilibrar los siguientes requisitos:

- El valor de `work_mem` (consulte Área de memoria de trabajo)
- La memoria restante después de descontar el valor `shared_buffers` (consultar Grupo de búferes)
- El máximo de conexiones abiertas y en uso, que está limitado por `max_connections`

Aumentar el tamaño del área de memoria de trabajo

En algunas situaciones, la única opción es aumentar la memoria que utiliza la sesión. Si las consultas están correctamente escritas y se utilizan las claves correctas para las uniones, considere aumentar el valor `work_mem`. Para obtener más información, consulte Área de memoria de trabajo.

Para saber cuántos archivos temporales genera una consulta, establezca `log_temp_files` en `0`. Si aumenta el valor de `work_mem` hasta el valor máximo identificado en los registros, evitará que la consulta genere archivos temporales. Sin embargo, `work_mem` establece el máximo por nodo del plan para cada conexión o proceso de trabajo paralelo. Si la base de datos tiene 5000 conexiones, y si cada una utiliza 256 MiB de memoria, el motor necesita 1.2 TiB de RAM. Esto significa que la instancia podría quedarse sin memoria.

Reservar suficiente memoria para el grupo de búferes compartidos

La base de datos utiliza áreas de memoria como el grupo de búferes compartidos, no solo el área de memoria de trabajo. Tenga en cuenta los requisitos de estas áreas de memoria adicionales antes de aumentar `work_mem`. Para más información sobre el grupo de búferes, consulte Grupo de búferes.

Por ejemplo, supongamos que su clase de instancia Aurora PostgreSQL es db.r5.2xlarge. Esta clase tiene 64 GiB de memoria. De forma predeterminada, el 75 por ciento de la memoria se reserva para el grupo de búferes compartidos. Después de restar la cantidad asignada al área de memoria compartida, quedan 16 384 MB. No asigne la memoria restante exclusivamente al área de memoria de trabajo porque el sistema operativo y el motor también necesitan memoria.

La memoria que puedes asignar a `work_mem` depende de la clase de instancia. Si utiliza una clase de instancia más grande, habrá más memoria disponible. Sin embargo, en el ejemplo anterior, no puedes usar más de 16 GiB. De lo contrario, la instancia dejará de estar disponible cuando se agote la memoria. Para recuperar la instancia del estado no disponible, los servicios de automatización de Aurora PostgreSQL se reinician automáticamente.

Administrar el número de conexiones

Supongamos que la instancia de su base de datos tiene 5 000 conexiones simultáneas. Cada conexión utiliza al menos 4 MiB de `work_mem`. El alto consumo de memoria de las conexiones puede degradar el rendimiento. Para ello, tiene las siguientes opciones:

- Actualizar a una clase de instancia mayor.
- Disminuir el número de conexiones simultáneas a la base de datos mediante el uso de un proxy de conexión o un grupo de conexiones.

En el caso de los proxies, considere Amazon RDS Proxy, pgBouncer o un grupo de conexiones acorde con su aplicación. Esta solución alivia la carga de la CPU. También reduce el riesgo cuando todas las conexiones requieren el área de memoria de trabajo. Cuando hay menos conexiones a la base de datos, puede aumentar el valor de `work_mem`. De esta manera, se reduce la ocurrencia de los eventos de espera `IO:BufFileRead` y `IO:BufFileWrite`. Además, las consultas que esperan el área de memoria de trabajo se aceleran de forma significativa.

IO:DataFileRead

El evento `IO:DataFileRead` ocurre cuando una conexión espera en un proceso backend para leer una página requerida del almacenamiento porque la página no está disponible en la memoria compartida.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Context

Todas las consultas y operaciones de manipulación de datos (DML) acceden a páginas en el grupo de búferes. Las instrucciones que pueden inducir lecturas incluyen `SELECT`, `UPDATE` y `DELETE`. Por ejemplo, un `UPDATE` puede leer páginas de tablas o índices. Si la página que se solicita o actualiza no está en el grupo de búferes compartidos, esta lectura puede provocar el evento `IO:DataFileRead`.

Dado que el grupo de búferes compartidos es finito, puede llenarse. En este caso, las solicitudes de páginas que no están en la memoria obligan a la base de datos a leer bloques del disco. Si el evento `IO:DataFileRead` se produce con frecuencia, es posible que el grupo de búferes compartidos sea demasiado pequeño para acomodar la carga de trabajo. Este problema se agudiza en las consultas `SELECT` que leen un gran número de filas que no caben en el grupo de búferes. Para más información sobre el grupo de búferes, consulte Grupo de búferes.

Causas probables del aumento de las esperas

Las causas más comunes del evento `IO:DataFileRead` son las siguientes:

Picos de conexión

Es posible que varias conexiones generen el mismo número de eventos de espera IO:DataFileRead. En este caso, puede producirse un pico (aumento repentino y grande) en los eventos `IO:DataFileRead`.

Instrucciones SELECT y DML que hacen escaneos secuenciales

Es posible que la aplicación realice una nueva operación. O una operación existente podría cambiar debido a un nuevo plan de ejecución. En estos casos, busque las tablas (particularmente las tablas grandes) que tengan un valor de `seq_scan` mayor. Encuéntrelas mediante la consulta de `pg_stat_user_tables`. Para rastrear las consultas que generan más operaciones de lectura, utilice la extensión `pg_stat_statements`.

CTAS y CREATE INDEX para grandes conjuntos de datos

Un CTAS es una instrucción `CREATE TABLE AS SELECT`. Si ejecuta un CTAS con un conjunto de datos grande como origen, o crea un índice en una tabla grande, puede producirse el evento `IO:DataFileRead`. Cuando se crea un índice, es posible que la base de datos tenga que leer todo el objeto mediante una exploración secuencial. Un CTAS genera lecturas `IO:DataFile` cuando las páginas no están en la memoria.

Varios procesos de trabajo de vacío que se ejecutan al mismo tiempo

Los procesos de trabajo de vacío pueden activarse de forma manual o automática. Se recomienda adoptar una estrategia de vacío agresiva. Sin embargo, cuando una tabla tiene muchas filas actualizadas o eliminadas, las esperas `IO:DataFileRead` aumentan. Una vez recuperado el espacio, el tiempo de vacío dedicado a `IO:DataFileRead` disminuye.

Ingesta de grandes cantidades de datos

Cuando la aplicación ingiere grandes cantidades de datos, las operaciones `ANALYZE` pueden producirse con mayor frecuencia. El proceso `ANALYZE` se puede activar mediante un desencadenador de autovacuum o invocarse de forma manual.

La operación `ANALYZE` lee un subconjunto de la tabla. El número de páginas que deben ser escaneadas se calcula al multiplicar 30 por el valor de `default_statistics_target`. Para obtener más información, consulte la documentación de PostgreSQL. El parámetro `default_statistics_target` acepta valores entre 1 y 10 000, siendo el valor por defecto 100.

Falta de recursos

Si el ancho de banda de la red de la instancia o la CPU se consumen, el evento `IO:DataFileRead` podría ocurrir con más frecuencia.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Verificar los filtros de predicado para las consultas que generan esperas
- Minimizar el efecto de las operaciones de mantenimiento
- Responder a un gran número de conexiones

Verificar los filtros de predicado para las consultas que generan esperas

Supongamos que identifica consultas específicas que generan eventos de espera `IO:DataFileRead`. Puede identificarlos con las siguientes técnicas:

- Información sobre rendimiento
- Vistas de catálogo como la que proporciona la extensión `pg_stat_statements`
- La vista de catálogo `pg_stat_all_tables`, si muestra de forma periódica un aumento del número de lecturas físicas
- La vista `pg_statio_all_tables`, si muestra que los contadores `_read` aumentan

Le recomendamos que determine qué filtros se utilizan en el predicado (cláusula `WHERE`) de estas consultas. Siga estas instrucciones:

- Ejecute el comando `EXPLAIN`. En la salida, identifique qué tipos de escaneos se utilizan. Un escaneo secuencial no indica necesariamente un problema. Las consultas que utilizan escaneos secuenciales producen naturalmente más eventos `IO:DataFileRead` en comparación con las consultas que utilizan filtros.

  Averigüe si la columna que aparece en la cláusula `WHERE` se encuentra en un índice. Si no es así, considere la posibilidad de crear un índice para esta columna. Este enfoque evita los escaneos secuenciales y reduce los eventos `IO:DataFileRead`. Si una consulta tiene filtros restrictivos y produce aún escaneos secuenciales, evalúe si se utilizan los índices adecuados.

- Verifique si la consulta tiene acceso a una tabla muy grande. En algunos casos, la partición de una tabla puede mejorar el rendimiento, ya que permite que la consulta solo lea las particiones necesarias.
- Examine la cardinalidad (número total de filas) de sus operaciones de unión. Tenga en cuenta lo restrictivos que son los valores que se pasan en los filtros de la cláusula `WHERE`. Si es posible, ajuste su consulta para reducir el número de filas que se pasan en cada paso del plan.

Minimizar el efecto de las operaciones de mantenimiento

Las operaciones de mantenimiento como `VACUUM` y `ANALYZE` son importantes. Le recomendamos que no las desactive ya que encontrará eventos de espera `IO:DataFileRead` relacionados con estas operaciones de mantenimiento. Los siguientes enfoques pueden minimizar el efecto de estas operaciones:

- Ejecute las operaciones de mantenimiento de forma manual durante las horas de menor actividad. Esta técnica evita que la base de datos alcance el umbral de las operaciones automáticas.
- Para tablas muy grandes, considere la posibilidad de particionar la tabla. Esta técnica reduce la sobrecarga de las operaciones de mantenimiento. La base de datos solo accede a las particiones que requieren mantenimiento.
- Cuando capture grandes cantidades de datos, considere la posibilidad de desactivar la característica de autoanálisis.

La característica de autovacuum se activa de forma automática para una tabla cuando la siguiente fórmula es verdadera.

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

La vista `pg_stat_user_tables` y el catálogo `pg_class` tienen varias filas. Una fila puede corresponder a una fila de la tabla. Esta fórmula asume que las `reltuples` son para una tabla específica. Los parámetros `autovacuum_vacuum_scale_factor` (0,20 de forma predeterminada) y `autovacuum_vacuum_threshold` (50 tuplas de forma predeterminada) se suelen establecer de forma global para toda la instancia. Sin embargo, puede establecer valores diferentes para una tabla específica.

Temas

- Busque tablas que consuman espacio de una forma innecesaria

- Buscar índices que consumen espacio innecesario
- Buscar tablas aptas para autovacuum

Busque tablas que consuman espacio de una forma innecesaria

Para buscar tablas que consumen más espacio del necesario, ejecute la siguiente consulta. Cuando esta consulta la ejecuta un rol de usuario de base de datos que no tiene el rol `rds_superuser`, devuelve información únicamente sobre las tablas para las que el rol de usuario tiene permisos de lectura. Esta consulta es compatible con la versión 12 y las versiones posteriores de PostgreSQL.

```
WITH report AS (
   SELECT   schemaname
           ,tblname
           ,n_dead_tup
           ,n_live_tup
           ,block_size*tblpages AS real_size
           ,(tblpages-est_tblpages)*block_size AS extra_size
           ,CASE WHEN tblpages - est_tblpages > 0
              THEN 100 * (tblpages - est_tblpages)/tblpages::float
              ELSE 0
            END AS extra_ratio, fillfactor, (tblpages-est_tblpages_ff)*block_size AS
 bloat_size
           ,CASE WHEN tblpages - est_tblpages_ff > 0
              THEN 100 * (tblpages - est_tblpages_ff)/tblpages::float
              ELSE 0
            END AS bloat_ratio
           ,is_na
    FROM (
          SELECT  ceil( reltuples / ( (block_size-page_hdr)/tpl_size ) ) +
 ceil( toasttuples / 4 ) AS est_tblpages
                 ,ceil( reltuples / ( (block_size-page_hdr)*fillfactor/
(tpl_size*100) ) ) + ceil( toasttuples / 4 ) AS est_tblpages_ff
                 ,tblpages
                 ,fillfactor
                 ,block_size
                 ,tblid
                 ,schemaname
                 ,tblname
                 ,n_dead_tup
                 ,n_live_tup
                 ,heappages
                 ,toastpages
```

```
                  ,is_na
             FROM (
                    SELECT ( 4 + tpl_hdr_size + tpl_data_size + (2*ma)
                               - CASE WHEN tpl_hdr_size%ma = 0 THEN ma ELSE
 tpl_hdr_size%ma END
                               - CASE WHEN ceil(tpl_data_size)::int%ma = 0 THEN ma ELSE
 ceil(tpl_data_size)::int%ma END
                           ) AS tpl_size
                           ,block_size - page_hdr AS size_per_block
                           ,(heappages + toastpages) AS tblpages
                           ,heappages
                           ,toastpages
                           ,reltuples
                           ,toasttuples
                           ,block_size
                           ,page_hdr
                           ,tblid
                           ,schemaname
                           ,tblname
                           ,fillfactor
                           ,is_na
                           ,n_dead_tup
                           ,n_live_tup
                          FROM (
                               SELECT  tbl.oid                       AS tblid
                                      ,ns.nspname                    AS schemaname
                                      ,tbl.relname                   AS tblname
                                      ,tbl.reltuples                 AS reltuples
                                      ,tbl.relpages                  AS heappages
                                      ,coalesce(toast.relpages, 0)   AS toastpages
                                      ,coalesce(toast.reltuples, 0)  AS toasttuples
                                      ,psat.n_dead_tup               AS n_dead_tup
                                      ,psat.n_live_tup               AS n_live_tup
                                      ,24                            AS page_hdr
                                      ,current_setting('block_size')::numeric AS
 block_size

 ,coalesce(substring( array_to_string(tbl.reloptions, ' ') FROM
 'fillfactor=([0-9]+)')::smallint, 100) AS fillfactor
                                      ,CASE WHEN version()~'mingw32' OR version()~'64-
bit|x86_64|ppc64|ia64|amd64' THEN 8 ELSE 4 END        AS ma
                                      ,23 + CASE WHEN MAX(coalesce(null_frac,0)) > 0
 THEN ( 7 + count(*) ) / 8 ELSE 0::int END              AS tpl_hdr_size
```

```
                                      ,sum( (1-coalesce(s.null_frac, 0)) *
 coalesce(s.avg_width, 1024) )                                AS tpl_data_size
                                      ,bool_or(att.atttypid =
 'pg_catalog.name'::regtype) OR count(att.attname) <> count(s.attname)        AS is_na
                                  FROM  pg_attribute       AS att
                                  JOIN  pg_class           AS tbl    ON (att.attrelid =
 tbl.oid)
                                  JOIN  pg_stat_all_tables AS psat   ON (tbl.oid =
 psat.relid)
                                  JOIN  pg_namespace       AS ns     ON (ns.oid =
 tbl.relnamespace)
                              LEFT JOIN  pg_stats          AS s      ON
 (s.schemaname=ns.nspname AND s.tablename = tbl.relname AND s.inherited=false AND
 s.attname=att.attname)
                              LEFT JOIN  pg_class          AS toast  ON
 (tbl.reltoastrelid = toast.oid)
                                 WHERE  att.attnum > 0
                                   AND  NOT att.attisdropped
                                   AND  tbl.relkind = 'r'
                                GROUP BY  tbl.oid, ns.nspname, tbl.relname,
 tbl.reltuples, tbl.relpages, toastpages, toasttuples, fillfactor, block_size, ma,
 n_dead_tup, n_live_tup
                                ORDER BY  schemaname, tblname
                            ) AS s
                  ) AS s2
       ) AS s3
 ORDER BY bloat_size DESC
)
  SELECT *
    FROM report
   WHERE bloat_ratio != 0
 -- AND schemaname = 'public'
 -- AND tblname = 'pgbench_accounts'
;

-- WHERE NOT is_na
--   AND tblpages*((pst).free_percent + (pst).dead_tuple_percent)::float4/100 >= 1
```

Puede comprobar si hay una sobrecarga de tablas e índices en su aplicación. Para obtener más información, consulte Diagnóstico de sobrecarga de tablas e índices.

Buscar índices que consumen espacio innecesario

Para buscar índices que consumen espacio innecesario, ejecute la siguiente consulta.

```
-- WARNING: run with a nonsuperuser role, the query inspects
-- only indexes on tables you have permissions to read.
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
    ) AS est_pages,
    coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
        + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
 MAXALIGN
            WHEN nulldatawidth = 0 THEN 0
            WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
```

```
            ELSE nulldatawidth::integer%maxalign
          END
      )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
      -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
    FROM (
      SELECT
        i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
AS table_oid,
        current_setting('block_size')::numeric AS bs, fillfactor,
        CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
          WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
THEN 8
          ELSE 4
        END AS maxalign,
        /* per page header, fixed size: 20 for 7.X, 24 for others */
        24 AS pagehdr,
        /* per page btree opaque data */
        16 AS pageopqdata,
        /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
        CASE WHEN max(coalesce(s.null_frac,0)) = 0
          THEN 2 -- IndexTupleData size
          ELSE 2 + (( 32 + 8 - 1 ) / 8)
          -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
index + 8 - 1 /8)
        END AS index_tuple_hdr_bm,
        /* data len: we remove null values save space using it fractionnal part from
stats */
        sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
nulldatawidth,
        max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
AS is_na
      FROM pg_attribute AS a
        JOIN (
          SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
            idx.reltuples, idx.relpages, idx.relam,
            indrelid, indexrelid, indkey::smallint[] AS attnum,
            coalesce(substring(
              array_to_string(idx.reloptions, ' ')
               from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
          FROM pg_index
            JOIN pg_class idx ON idx.oid=pg_index.indexrelid
            JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
            JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
          WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
```

```
        ) AS i ON a.attrelid = i.indexrelid
        JOIN pg_stats AS s ON s.schemaname = i.nspname
          AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
          -- stats from tbl
          OR  (s.tablename = i.idxname AND s.attname = a.attname))
          -- stats from functionnal cols
        JOIN pg_type AS t ON a.atttypid = t.oid
      WHERE a.attnum > 0
      GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
    ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

Buscar tablas aptas para autovacuum

Para buscar tablas que se puedan vaciar automáticamente, ejecute la siguiente consulta.

```
--This query shows tables that need vacuuming and are eligible candidates.
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
        coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
        as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
```

```
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC;
```

Responder a un gran número de conexiones

Cuando se monitorea Amazon CloudWatch, se puede encontrar que la métrica `DatabaseConnections` se dispara. Este aumento indica un mayor número de conexiones a su base de datos. Se recomienda el siguiente enfoque:

- Limite el número de conexiones que la aplicación puede abrir con cada instancia. Si la aplicación tiene una característica de grupo de conexiones integrada, establezca un número razonable de conexiones. Base el número en lo que las vCPU de su instancia puedan paralelizar de forma efectiva.

  Si su aplicación no utiliza una característica de grupo de conexiones, considere utilizar Amazon RDS Proxy o una alternativa. Este enfoque permite que su aplicación abra varias conexiones con el equilibrador de carga. El equilibrador puede entonces abrir un número restringido de conexiones con la base de datos. Como se ejecutan menos conexiones en paralelo, la instancia de base de datos hace menos cambios de contexto en el núcleo. Las consultas deberían progresar más rápido, lo que provocaría menos eventos de espera. Para obtener más información, consulte Amazon RDS Proxy para Aurora.

- Siempre que sea posible, aproveche los nodos de lectura para Aurora PostgreSQL y las réplicas de lectura para RDS for PostgreSQL. Cuando la aplicación ejecute una operación de solo lectura, envíe estas solicitudes al punto de conexión de solo lectura. Esta técnica reparte las peticiones de la aplicación entre todos los nodos de lectura, lo que reduce la presión de E/S en el nodo de escritura.
- Considere la posibilidad de escalar verticalmente su instancia de base de datos. Una clase de instancia de mayor capacidad proporciona más memoria, lo que le da a Aurora PostgreSQL un grupo de búferes compartidos más grande para mantener las páginas. El tamaño más grande también le da a la instancia de base de datos más vCPU para manejar las conexiones. Más vCPU son especialmente útiles cuando las operaciones que generan eventos de espera `IO:DataFileRead` se escriben.

IO:XactSync

El evento `IO:XactSync` ocurre cuando la base de datos espera que el subsistema de almacenamiento de Aurora acuse de recibo la confirmación de una transacción regular, o la confirmación o restauración de una transacción preparada. Una transacción preparada es parte del soporte de PostgreSQL para una confirmación en dos fases. Este evento también puede ocurrir cuando una consulta está esperando que se confirme otra transacción, en concreto en los casos en los que la confirmación automática está desactivada. En tales casos, puede parecer que las actualizaciones están esperando en XactSync aunque aún no se hayan confirmado.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Contexto

El evento `IO:XactSync` indica que la instancia pasa tiempo esperando que el subsistema de almacenamiento de Aurora confirme que los datos de la transacción fueron procesados.

Causas probables del aumento de las esperas

Cuando el evento `IO:XactSync` aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

Saturación de la red

El tráfico entre los clientes y la instancia de base de datos o el tráfico hacia el subsistema de almacenamiento podría ser demasiado pesado para el ancho de banda de la red.

Presión de la CPU

Una gran carga de trabajo podría evitar que el daemon de almacenamiento de Aurora obtenga suficiente tiempo de CPU.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Monitorear los recursos
- Escalar verticalmente la CPU
- Aumentar el ancho de banda de la red
- Reducir el número de confirmaciones

Monitorear los recursos

Para determinar la causa del aumento de los eventos `IO:XactSync`, verifique las siguientes métricas:

- `WriteThroughput` y `CommitThroughput`: los cambios en el rendimiento de escritura o rendimiento de confirmación pueden mostrar un aumento de la carga de trabajo.

- `WriteLatency` y `CommitLatency`: los cambios en la latencia de escritura o en la latencia de confirmación pueden mostrar que se solicita al subsistema de almacenamiento que realice más trabajo.
- `CPUUtilization`: si el uso de la CPU de la instancia está por encima del 90 por ciento, el daemon de almacenamiento de Aurora puede no recibir suficiente tiempo en la CPU. En este caso, el rendimiento de E/S se reduce.

Para obtener información acerca de estas métricas, consulte Métricas de nivel de instancia para Amazon Aurora.

Escalar verticalmente la CPU

Para solucionar los problemas de falta de CPU, considere la posibilidad de cambiar a un tipo de instancia con más capacidad de CPU. Para obtener información sobre la capacidad de la CPU para una clase de instancia de base de datos, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

Aumentar el ancho de banda de la red

Para determinar si la instancia se encuentra en los límites del ancho de banda de la red, verifique los siguientes eventos de espera:

- `IO:DataFileRead`, `IO:BufferRead`, `IO:BufferWrite` y `IO:XactWrite`: las consultas que utilizan grandes cantidades de E/S pueden generar más de estos eventos de espera.
- `Client:ClientRead` y `Client:ClientWrite`: las consultas con grandes cantidades de comunicación con el cliente pueden generar más de estos eventos de espera.

Si el ancho de banda de la red es un problema, considere cambiar a un tipo de instancia con más ancho de banda de red. Para obtener información sobre el rendimiento de la red para una clase de instancia de base de datos, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

Reducir el número de confirmaciones

Para reducir el número de confirmaciones, combine las instrucciones en bloques de transacciones.

IPC:DamRecordTxAck

El evento `IPC:DamRecordTxAck` ocurre cuando Aurora PostgreSQL en una sesión que utiliza transmisiones de actividad de la base de datos genera un evento de transmisión de actividad, y luego espera que ese evento se vuelva permanente.

Temas

- Versiones del motor relevantes
- Context
- Causas
- Acciones

Versiones del motor relevantes

Esta información de eventos de espera es relevante para todas las versiones de Aurora PostgreSQL 10.7 y posteriores a 10, 11.4 y posteriores a 11, y todas las versiones 12 y 13.

Context

En el modo síncrono, la durabilidad de los eventos de transmisión de actividad se ve favorecida por el rendimiento de la base de datos. Mientras se espera una escritura duradera del evento, la sesión bloquea otra actividad de la base de datos, lo que provoca el evento de espera `IPC:DamRecordTxAck`.

Causas

La causa más común para que el evento `IPC:DamRecordTxAck` aparezca en las esperas más altas es que la característica de Transmisiones de actividades de la base de datos (DAS) es una auditoría integral. Una mayor actividad SQL genera eventos de transmisiones de actividad que se deben registrar.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera:

- Reduzca el número de instrucciones SQL o desactive las transmisiones de actividad de la base de datos. De esta forma se reduce el número de eventos que requieren escrituras duraderas.

- Cambia al modo asíncrono. Esto ayuda a reducir la contención en el evento de espera `IPC:DamRecordTxAck`.

  Sin embargo, la característica DAS no puede garantizar la durabilidad de cada evento en modo asíncrono.

Eventos de espera IPC:parallel

Los siguientes `IPC:parallel wait events` indican que una sesión está esperando una comunicación entre procesos relacionada con operaciones de ejecución de consultas en paralelo.

- `IPC:BgWorkerStartup`: un proceso está esperando a que un proceso de trabajo paralelo complete la secuencia de inicio. Esto sucede al inicializar los procesos de trabajo para la ejecución de consultas en paralelo.
- `IPC:BgWorkerShutdown`: un proceso está esperando a que un proceso de trabajo paralelo complete la secuencia de apagado. Esto sucede durante la fase de limpieza de la ejecución de consultas en paralelo.
- `IPC:ExecuteGather`: un proceso está esperando recibir datos de procesos de trabajo paralelos durante la ejecución de consultas. Esto ocurre cuando el proceso líder necesita recopilar resultados de los procesos de trabajo.
- `IPC:ParallelFinish`: un proceso está esperando a que los procesos de trabajo paralelos finalicen la ejecución y comuniquen los resultados finales. Esto sucede durante la fase de finalización de la ejecución de consultas en paralelo.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Contexto

La ejecución de consultas en paralelo en PostgreSQL implica que varios procesos trabajan juntos para procesar una sola consulta. Cuando se determina que una consulta es adecuada para la paralelización, un proceso líder coordina uno o varios procesos de trabajo paralelos en función de la configuración del parámetro `max_parallel_workers_per_gather`. El proceso líder divide el trabajo entre los procesos de trabajo, cada proceso de trabajo procesa la parte de los datos de forma independiente y los resultados se recopilan de nuevo en el proceso líder.

> **Note**
>
> Cada proceso de trabajo paralelo funciona como un proceso independiente con requisitos de recursos similares a los de una sesión de usuario completa. Esto significa que una consulta en paralelo con cuatro procesos de trabajo puede consumir hasta cinco veces más recursos (CPU, memoria y ancho de banda de E/S) que una consulta que no sea en paralelo, ya que tanto el proceso líder como cada proceso de trabajo mantienen las asignaciones de recursos propias. Por ejemplo, la configuración como `work_mem` se aplica individualmente a cada proceso de trabajo, lo que puede multiplicar el uso total de memoria en todos los procesos.

La arquitectura de consulta en paralelo consta de tres componentes principales:

- Proceso líder: el proceso principal que inicia la operación paralela, divide la carga de trabajo y coordina los procesos de trabajo.
- Procesos de trabajo: procesos en segundo plano que ejecutan partes de la consulta en paralelo.
- Recopilación/combinación de recopilaciones: operaciones que combinan los resultados de varios procesos de trabajo y los devuelven al líder

Durante la ejecución en paralelo, los procesos deben comunicarse entre sí a través de mecanismos de comunicación entre procesos (IPC). Estos eventos de espera IPC se producen durante diferentes fases:

- Inicio de procesos de trabajo: cuando se inicializan los procesos de trabajo paralelos
- Intercambio de datos: cuando los procesos de trabajo procesan datos y envían los resultados al líder
- Cierre de procesos de trabajo: cuando finaliza la ejecución paralela y se terminan los procesos de trabajo

- Puntos de sincronización: cuando los procesos necesitan coordinarse o esperar a que otros procesos completen las tareas

Comprender estos eventos de espera es fundamental para diagnosticar problemas de rendimiento relacionados con la ejecución de consultas en paralelo, sobre todo en entornos de alta simultaneidad en los que pueden ejecutarse varias consultas en paralelo al mismo tiempo.

Causas probables del aumento del tiempo de espera

Hay varios factores que pueden contribuir a un aumento de los eventos de espera IPC relacionados con la ejecución en paralelo:

Alta simultaneidad de consultas en paralelo

Cuando se ejecutan muchas consultas en paralelo al mismo tiempo, puede producirse un conflicto de recursos y un aumento de los tiempos de espera para las operaciones IPC. Esto es muy habitual en sistemas con grandes volúmenes de transacciones o cargas de trabajo de análisis.

Planes de consultas en paralelo poco óptimos

Si el planificador de consultas elige planes paralelos ineficientes, puede dar lugar a una paralelización innecesaria o a una distribución deficiente del trabajo entre los procesos de trabajo. Esto puede provocar un aumento de las esperas de IPC, sobre todo para los eventos IPC:ExecuteGather e IPC:ParallelFinish. Estos problemas de planificación suelen deberse a estadísticas obsoletas y a la saturación de tablas e índices.

Inicio y apagado frecuentes de procesos de trabajo paralelos

Las consultas de corta duración que inician y terminan con frecuencia procesos de trabajo paralelos pueden provocar un aumento de los eventos `IPC:BgWorkerStartup` y `IPC:BgWorkerShutdown`. Esto se observa a menudo en cargas de trabajo OLTP con muchas consultas pequeñas y paralelizables.

Limitaciones de recursos

La capacidad limitada de CPU, memoria o E/S puede provocar cuellos de botella en la ejecución paralela, lo que aumenta los tiempos de espera en todos los eventos IPC. Por ejemplo, si la CPU está saturada, los procesos de trabajo pueden tardar más en iniciarse o en procesar la parte del trabajo.

Estructuras de consultas complejas

Las consultas con varios niveles de paralelismo (por ejemplo, uniones paralelas seguidas de agregaciones paralelas) pueden dar lugar a patrones IPC más complejos y a un aumento de los tiempos de espera, sobre todo para eventos `IPC:ExecuteGather`.

Grandes conjuntos de resultados

Las consultas que producen conjuntos de resultados grandes pueden provocar un aumento de los tiempos de espera de `IPC:ExecuteGather`, ya que el proceso líder dedica más tiempo a recopilar y procesar los resultados de los procesos de trabajo.

Comprender estos factores puede ayudar a diagnosticar y resolver problemas de rendimiento relacionados con la ejecución de consultas paralelas en Aurora PostgreSQL.

Acciones

Cuando ve esperas relacionadas con consultas en paralelo, normalmente significa que un proceso backend está coordinando o esperando procesos de trabajo paralelos. Estas esperas son comunes durante la ejecución de planes paralelos. Puede investigar y mitigar el impacto de estas esperas mediante la supervisión del uso de procesos de trabajo paralelos, la revisión de la configuración de los parámetros y el ajuste de la ejecución de las consultas y asignación de recursos.

Temas

- Análisis de los planes de consulta en busca de paralelismo ineficiente
- Supervisión del uso de consultas en paralelo
- Revisión y ajuste de la configuración de consultas en paralelo
- Optimización de la asignación de recursos
- Investigación de la administración de conexiones
- Revisión y optimización de las operaciones de mantenimiento
- Uso de la administración de planes de consulta (QPM)

Análisis de los planes de consulta en busca de paralelismo ineficiente

La ejecución de consultas en paralelo a menudo puede provocar inestabilidad del sistema, picos de CPU y variaciones impredecibles en el rendimiento de las consultas. Es fundamental analizar a fondo si el paralelismo mejora realmente la carga de trabajo específica. Utilice EXPLAIN ANALYZE para revisar los planes de ejecución de consultas en paralelo.

Deshabilite temporalmente el paralelismo de la sesión para comparar la eficiencia del plan:

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

Vuelva a habilitar el paralelismo y compare:

```
RESET max_parallel_workers_per_gather;
EXPLAIN ANALYZE <your_query>;
```

Si al deshabilitar el paralelismo se obtienen resultados mejores o más coherentes, considere la posibilidad de deshabilitarlo para consultas específicas de sesión mediante comandos SET. Para lograr un impacto más amplio, es posible que desee deshabilitar el paralelismo de instancia mediante el ajuste de los parámetros pertinentes en el clúster o el grupo de parámetros de instancia. Para obtener más información, consulte Parámetros de Amazon Aurora PostgreSQL..

Supervisión del uso de consultas en paralelo

Utilice las siguientes consultas para obtener visibilidad de la actividad y la capacidad de las consultas en paralelo:

Compruebe los procesos de trabajo paralelos activos:

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
    backend_type = 'parallel worker';
```

Esta consulta muestra el número de procesos de trabajo paralelos activos. Un valor alto puede indicar que “max_parallel_workers” se ha configurado con un valor alto y es recomendable que considere la posibilidad de reducirlo.

Compruebe las consultas en paralelo simultáneas:

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

Esta consulta devuelve el número de procesos líderes distintos que han iniciado consultas en paralelo. Un número alto indica que varias sesiones están ejecutando consultas en paralelo simultáneamente, lo que puede aumentar la demanda de CPU y memoria.

Revisión y ajuste de la configuración de consultas en paralelo

Revise los siguientes parámetros para asegurarse de que se ajustan a la carga de trabajo:

- `max_parallel_workers`: número total de procesos de trabajo paralelos en todas las sesiones.
- `max_parallel_workers_per_gather`: número máximo de procesos de trabajo por consulta.

Para cargas de trabajo OLAP, aumentar estos valores puede mejorar el rendimiento. Para cargas de trabajo OLTP, por lo general, se prefieren valores más bajos.

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

Optimización de la asignación de recursos

Supervise la utilización de la CPU y considere la posibilidad de ajustar el número de vCPU si es coherentemente alto y si la aplicación se beneficia de las consultas en paralelo. Asegúrese de que haya memoria suficiente para las operaciones paralelas.

- Utilice las métricas de Información de rendimiento para determinar si el sistema está limitado por la CPU.
- Cada proceso de trabajo paralelo utiliza un `work_mem` propio. Asegúrese de que el uso total de memoria se encuentra dentro de los límites de la instancia.

Las consultas en paralelo pueden consumir muchos más recursos que las consultas que no son en paralelo, ya que cada proceso de trabajo es un proceso completamente independiente que tiene aproximadamente el mismo impacto en el sistema que una sesión de usuario adicional. Esto debe tenerse en cuenta al elegir un valor para esta configuración, así como al configurar otras opciones que controlan la utilización de recursos, como `work_mem`. Para obtener más información, consulte la documentación de PostgreSQL. Los límites de recursos, como `work_mem`, se aplican individualmente a cada proceso de trabajo, lo que significa que la utilización total puede ser mucho mayor en todos los procesos de lo que sería normalmente para un solo proceso.

Considere la posibilidad de aumentar las vCPU o ajustar los parámetros de memoria si la carga de trabajo está muy paralelizada.

Investigación de la administración de conexiones

Si se agota la conexión, revise las estrategias de agrupación de conexiones de la aplicación. Considere la posibilidad de implementar la agrupación de conexiones de aplicación si aún no se utiliza.

Revisión y optimización de las operaciones de mantenimiento

Coordine la creación de índices y otras tareas de mantenimiento para evitar la contienda por los recursos. Considere la posibilidad de programar estas operaciones durante las horas de menor actividad. Evite programar tareas de mantenimiento intensas (por ejemplo, creaciones de índices en paralelo) durante periodos de alta carga de consultas de los usuarios. Estas operaciones pueden consumir procesos de trabajo paralelos y afectar al rendimiento de las consultas habituales.

Uso de la administración de planes de consulta (QPM)

En Aurora PostgreSQL, la característica de administración de planes de consulta (QPM) está diseñada para garantizar la adaptabilidad y la estabilidad de los planes, independientemente de los cambios en el entorno de la base de datos que puedan causar la regresión del plan de consulta. Para obtener más información, consulte Descripción general de la administración de planes de consultas en Aurora PostgreSQL. QPM proporciona cierto control sobre el optimizador. Revise los planes aprobados en QPM para asegurarse de que se ajustan a la configuración de paralelismo actual. Actualice o elimine los planes obsoletos que puedan estar forzando una ejecución paralela poco óptima.

También puede corregir los planes con pg_hint_plan. Para obtener más información, consulte Corrección de planes mediante pg_hint_plan. Puede utilizar la sugerencia denominada `Parallel`

para forzar la ejecución paralela. Para obtener más información, consulte Sugerencias para planes paralelos.

IPC:ProcArrayGroupUpdate

El evento `IPC:ProcArrayGroupUpdate` se produce cuando una sesión está esperando a que líder de grupo actualice el estado de la transacción al final de esa operación. Mientras PostgreSQL generalmente asocia los eventos de espera de tipo IPC con las operaciones de consulta paralelas, este evento de espera en particular no es específico de las consultas paralelas.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Contexto

Descripción de la matriz de procesos: la matriz de procesos (proc) es una estructura de memoria compartida en PostgreSQL. Contiene información sobre todos los procesos en ejecución, incluidos los detalles de las transacciones. Al finalizar la transacción (`COMMIT` o `ROLLBACK`), el ProcArray debe actualizarse para reflejar el cambio y borrar el ID de transacción de la matriz. La sesión que intente finalizar su transacción debe adquirir un bloqueo exclusivo en el ProcArray. Esto evita que otros procesos obtengan bloqueos compartidos o exclusivos sobre ella.

Mecanismo de actualización grupal: al realizar un COMMIT o ROLLBACK, si un proceso de backend no puede obtener un ProcArrayLock en modo exclusivo, actualiza un campo especial llamado ProcArrayGroupMember. Esto agrega la transacción a la lista de sesiones que pretenden finalizar. A continuación, este proceso de backend se suspende y el tiempo que duerme se configura como el evento de espera ProcArrayGroupUpdate. El primer proceso de ProcArray con ProcArrayGroupMember, denominado proceso de líder, adquiere el ProcArrayLock en modo exclusivo. A continuación, borra la lista de procesos en espera de ser liquidados por el ID de

transacción del grupo. Una vez finalizada esta operación, el líder libera el ProcArrayLock y activa todos los procesos de la lista y les notifica que la transacción se ha completado.

Causas probables del aumento de las esperas

Cuantos más procesos estén en ejecución, más tiempo conservará el líder un ProcArrayLock en modo exclusivo. En consecuencia, cuantas más transacciones de escritura acaben en un escenario de actualización grupal, más se provocará una posible acumulación de procesos en espera en el evento de espera `ProcArrayGroupUpdate`. En la vista de SQL principal de información de base de datos, verá que COMMIT es la instrucción que contiene la mayor parte del tiempo de espera. Esto es de esperar, pero requerirá una investigación más profunda del SQL de escritura específico que se está ejecutando para determinar qué medidas se deben tomar.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera. Identifique eventos de `IPC:ProcArrayGroupUpdate` mediante Información de rendimiento de Amazon RDS o la consulta de la vista del sistema PostgreSQL `pg_stat_activity`.

Temas

- Supervisión de las operaciones de confirmación y reversión de transacciones
- Reducción de la simultaneidad
- Implementación de la agrupación de conexiones

Supervisión de las operaciones de confirmación y reversión de transacciones

Supervise las confirmaciones y las reversiones: un mayor número de confirmaciones y reversiones puede aumentar la presión sobre ProcArray. Por ejemplo, si una instrucción SQL comienza a producir errores debido al aumento de las infracciones de claves duplicadas, es posible que se produzca un aumento de las reversiones, lo que puede aumentar la contención de ProcArray y la sobrecarga de tablas.

La información de base de datos de Amazon RDS proporciona las métricas de PostgreSQL `xact_commit` y `xact_rollback` para informar del número de confirmaciones y reversiones por segundo.

Reducción de la simultaneidad

Transacciones por lotes: siempre que sea posible, agrupe las operaciones en transacciones individuales para reducir las operaciones de confirmación o reversión.

Limite la simultaneidad: reduzca la cantidad de transacciones activas simultáneamente para evitar la contención de bloqueos en ProcArray. Si bien requerirá algunas pruebas, reducir el número total de conexiones simultáneas puede reducir la contención y mantener el rendimiento.

Implementación de la agrupación de conexiones

Soluciones de agrupación de conexiones: utilice la agrupación de conexiones para administrar las conexiones de bases de datos de manera eficiente, lo que reduce la cantidad total de backends y, por lo tanto, la carga de trabajo de ProcArray. Si bien requerirá algunas pruebas, reducir el número total de conexiones simultáneas puede reducir la contención y mantener el rendimiento.

Para obtener más información, consulte Agrupación de conexiones para Aurora PostgreSQL.

Reduzca las tormentas de conexiones: del mismo modo, un patrón de creación y terminación frecuentes de conexiones genera una presión adicional sobre ProcArray. Al reducir este patrón, se reduce la contención general.

Lock:advisory

El evento `Lock:advisory` ocurre cuando una aplicación PostgreSQL utiliza un bloqueo para coordinar la actividad entre varias sesiones.

Temas

- Versiones del motor relevantes
- Context
- Causas
- Acciones

Versiones del motor relevantes

Esta información de eventos de espera es relevante para las versiones 9.6 y posteriores de Aurora PostgreSQL.

Context

Los bloqueos consultivos de PostgreSQL son bloqueos cooperativos a nivel de aplicación, bloqueados y desbloqueados explícitamente por el código de la aplicación del usuario. Una aplicación puede usar los bloqueos consultivos de PostgreSQL para coordinar la actividad a través de varias sesiones. A diferencia de los bloqueos regulares a nivel de objeto o de fila, la aplicación tiene control total sobre el tiempo de vida del bloqueo. Para más información, consulte Advisory Locks en la documentación de PostgreSQL.

Los bloqueos consultivos se pueden liberar antes de que finalice una transacción o mantenerse en una sesión a través de las transacciones. Esto no es verdadero para los bloqueos implícitos, forzados por el sistema, como un bloqueo de acceso exclusivo a una tabla adquirido por una instrucción `CREATE INDEX`.

Para una descripción de las funciones que se utilizan para adquirir (bloquear) y liberar (desbloquear) los bloqueos de asesoramiento, consulte Advisory Lock Functions en la documentación de PostgreSQL.

Los bloqueos consultivos se implementan sobre el sistema de bloqueo regular de PostgreSQL y son visibles en la vista del sistema `pg_locks`.

Causas

Este tipo de bloqueo es controlado exclusivamente por una aplicación que lo utiliza de forma explícita. Los bloqueos consultivos que se adquieren para cada fila como parte de una consulta pueden causar un pico de bloqueos o una acumulación a largo plazo.

Estos efectos se producen cuando la consulta se ejecuta de forma que adquiere bloqueos en más filas de las que devuelve la consulta. La aplicación debe liberar finalmente todos los bloqueos, pero si se adquieren bloqueos en filas que no se devuelven, la aplicación no puede encontrar todos los bloqueos.

El siguiente ejemplo proviene de Advisory Locks en la documentación de PostgreSQL.

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

En este ejemplo, la cláusula `LIMIT` solo puede detener la salida de la consulta después de que las filas se hayan seleccionado internamente y sus valores de ID se hayan bloqueado. Esto puede ocurrir de forma repentina cuando un volumen de datos creciente hace que el planificador elija un

plan de ejecución diferente que no fue probado durante el desarrollo. La acumulación en este caso ocurre porque la aplicación llama explícitamente a `pg_advisory_unlock` para cada valor de ID que fue bloqueado. Sin embargo, en este caso no se puede encontrar el conjunto de bloqueos adquiridos en las filas que no fueron devueltas. Como los bloqueos se adquieren a nivel de sesión, no se liberan automáticamente al final de la transacción.

Otra posible causa de los picos de intentos de bloqueo son los conflictos involuntarios. En estos conflictos, partes no relacionadas de la aplicación comparten el mismo espacio de ID de bloqueo por error.

Acciones

Revisar el uso de la aplicación de los bloqueos consultivos y detallar dónde y cuándo en el flujo de la aplicación se adquiere y libera cada tipo de bloqueo consultivo.

Determine si una sesión adquiere demasiados bloqueos o si una sesión de larga duración no libera los bloqueos con suficiente antelación, lo que provoca una acumulación lenta de bloqueos. Puede corregir una acumulación lenta de bloqueos de sesión si finaliza la sesión con `pg_terminate_backend(pid)`.

Un cliente en espera de un bloqueo consultivo aparece en `pg_stat_activity` con `wait_event_type=Lock` y `wait_event=advisory`. Puede obtener valores de bloqueo específicos al consultar la vista del sistema `pg_locks` para el mismo `pid`, buscando `locktype=advisory` y `granted=f`.

A continuación, puede identificar la sesión de bloqueo al consultar `pg_locks` para el mismo bloqueo consultivo con `granted=t`, como se muestra en el siguiente ejemplo.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
```

```
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
   FROM pg_catalog.pg_locks blocked_locks
   JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
blocked_locks.pid
   JOIN pg_catalog.pg_locks blocking_locks
       ON blocking_locks.locktype = blocked_locks.locktype
       AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
       AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
       AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
       AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
       AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
       AND blocking_locks.transactionid IS NOT DISTINCT FROM
blocked_locks.transactionid
       AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
       AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
       AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
       AND blocking_locks.pid != blocked_locks.pid
   JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
blocking_locks.pid
   WHERE NOT blocked_locks.GRANTED;
```

Todas las funciones de la API de bloqueo consultivo tienen dos conjuntos de argumentos, un argumento `bigint` o dos argumentos `integer`:

- Para las funciones API con un argumento `bigint`, los 32 bits superiores están en `pg_locks.classid` y los 32 bits inferiores están en `pg_locks.objid`.
- Para las funciones de la API con dos argumentos `integer`, el primer argumento es `pg_locks.classid` y el segundo argumento es `pg_locks.objid`.

El valor de `pg_locks.objsubid` indica qué forma de la API se utilizó: 1 significa un argumento `bigint`; 2 significa dos argumentos `integer`.

Lock:extend

El evento `Lock:extend` se produce cuando un proceso del backend espera bloquear una relación para extenderla mientras otro proceso tiene un bloqueo en esa relación con el mismo propósito.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Context

El evento `Lock:extend` indica que un proceso backend se encuentra a la espera de extender una relación sobre la que otro proceso backend tiene un bloqueo mientras está extendiendo esa relación. Debido a que solo un proceso a la vez puede extender una relación, el sistema genera un evento de espera `Lock:extend`. Las operaciones `INSERT`, `COPY` y `UPDATE` pueden generar este evento.

Causas probables del aumento de las esperas

Cuando el evento `Lock:extend` aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

Aumento de las inserciones o actualizaciones concurrentes en la misma tabla

Puede haber un aumento en el número de sesiones concurrentes con consultas que insertan o actualizan la misma tabla.

Ancho de banda de red insuficiente

El ancho de banda de la red en la instancia de base de datos puede ser insuficiente para las necesidades de comunicación del almacenamiento de la carga de trabajo actual. Esto puede contribuir a la latencia del almacenamiento que provoca un aumento de los eventos `Lock:extend`.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Reducir las inserciones y actualizaciones concurrentes en la misma relación
- Aumentar el ancho de banda de la red

Reducir las inserciones y actualizaciones concurrentes en la misma relación

En primer lugar, determine si hay un aumento de las métricas `tup_inserted` y `tup_updated` y un aumento de este evento de espera. Si es así, verifique qué relaciones están en alta contención para las operaciones de inserción y actualización. Para determinar esto, consulte la vista `pg_stat_all_tables` para los valores de los campos `n_tup_ins` y `n_tup_upd`. Para obtener información sobre la vista `pg_stat_all_tables`, consulte pg_stat_all_tables en la documentación de PostgreSQL.

Para obtener más información acerca de las consultas bloqueadas y no bloqueadas, consulte `pg_stat_activity` como en el siguiente ejemplo:

```
SELECT
    blocked.pid,
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+---------------------+--------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Después de identificar las relaciones que contribuyen a aumentar los eventos `Lock:extend`, utilice las siguientes técnicas para reducir la contención:

- Compruebe si puede utilizar el particionamiento para reducir la contención para la misma tabla. Separar las tuplas insertadas o actualizadas en diferentes particiones puede reducir la contención. Para obtener información sobre las particiones, consulte Administración de las particiones de PostgreSQL con la extensión pg_partman.
- Si el evento de espera se debe principalmente a la actividad de actualización, considere reducir el valor de fillfactor de la relación. Esto puede reducir las solicitudes de nuevos bloques durante la actualización. El fillfactor es un parámetro de almacenamiento para una tabla que determina la cantidad máxima de espacio para empaquetar una página de la tabla. Se expresa como un porcentaje del espacio total de una página. Para más información sobre el parámetro fillfactor, consulte CREATE TABLE en la documentación de PostgreSQL.

  > **Important**
  >
  > Recomendamos ampliamente que pruebe su sistema si cambia el fillfactor porque cambiar este valor puede impactar negativamente en el rendimiento, de acuerdo a su carga de trabajo.

Aumentar el ancho de banda de la red

Para ver si hay un aumento en la latencia de escritura, verifique la métrica `WriteLatency` en CloudWatch. Si lo hay, utilice las métricas `WriteThroughput` y `ReadThroughput` de Amazon CloudWatch para monitorear el tráfico relacionado con el almacenamiento en el clúster de base de datos. Estas métricas pueden ayudarle a determinar si el ancho de banda de la red es suficiente para la actividad de almacenamiento de su carga de trabajo.

Si el ancho de banda de su red no es suficiente, auméntelo. Si la instancia de base de datos alcanza los límites del ancho de banda de la red, la única forma de aumentar el ancho de banda es aumentar el tamaño de la instancia de base de datos.

Para obtener más información acerca de las métricas de CloudWatch, consulte Métricas de Amazon CloudWatch para Amazon Aurora. Para obtener información sobre el rendimiento de la red para cada clase de instancia de base de datos, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

Lock:Relation

El evento `Lock:Relation` ocurre cuando una consulta espera adquirir un bloqueo en una tabla o vista (relación) que se encuentra bloqueada por otra transacción.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Context

La mayoría de los comandos de PostgreSQL utilizan implícitamente bloqueos para controlar el acceso concurrente a los datos de las tablas. También puede utilizar estos bloqueos explícitamente en su código de aplicación con el comando `LOCK`. Muchos modos de bloqueo no son compatibles entre sí, y pueden bloquear las transacciones cuando intentan acceder al mismo objeto. Cuando esto sucede, Aurora PostgreSQL genera un evento `Lock:Relation`. Algunos ejemplos comunes son los siguientes:

- Los bloqueos exclusivos como `ACCESS EXCLUSIVE` pueden bloquear todos los accesos concurrentes. Las operaciones del lenguaje de definición de datos (DDL) como `DROP TABLE`, `TRUNCATE`, `VACUUM FULL` y `CLUSTER` adquieren bloqueos `ACCESS EXCLUSIVE` implícitamente. `ACCESS EXCLUSIVE` es también el modo de bloqueo por defecto para las instrucciones `LOCK TABLE` que no especifican un modo explícitamente.
- El uso de `CREATE INDEX (without CONCURRENT)` en una tabla entra en conflicto con las instrucciones de lenguaje de manipulación de datos (DML) `UPDATE`, `DELETE` e `INSERT`, que adquieren bloqueos `ROW EXCLUSIVE`.

Para más información sobre los bloqueos a nivel de tabla y los modos de bloqueo conflictivos, consulte Explicit Locking en la documentación de PostgreSQL.

Las consultas y transacciones que se bloquean normalmente se desbloquean de una de las siguientes maneras:

- Consulta de bloqueo: la aplicación puede cancelar la consulta o el usuario puede terminar el proceso. El motor también puede forzar la finalización de la consulta debido al tiempo de espera de una sesión o a un mecanismo de detección de bloqueos.
- Transacción bloqueada: una transacción deja de bloquearse cuando se ejecuta una instrucción `ROLLBACK` o `COMMIT`. Las restauraciones también ocurren de forma automática cuando las sesiones se desconectan por un cliente o por problemas de red, o se terminan. Las sesiones se pueden terminar cuando el motor de base de datos se apaga, cuando el sistema se queda sin memoria, etc.

Causas probables del aumento del tiempo de espera

Cuando el evento `Lock:Relation` se produce con más frecuencia de lo normal, puede indicar un problema de rendimiento. Las causas típicas son las siguientes:

Aumento de las sesiones concurrentes con bloqueos de tablas en conflicto

Puede haber un aumento en el número de sesiones concurrentes con consultas que bloquean la misma tabla con modos de bloqueo conflictivos.

Operaciones de mantenimiento

Las operaciones de mantenimiento de estado como `VACUUM` y `ANALYZE` pueden aumentar significativamente el número de bloqueos conflictivos. `VACUUM FULL` adquiere un bloqueo `ACCESS EXCLUSIVE`, y `ANALYZE` adquiere un bloqueo `SHARE UPDATE EXCLUSIVE`. Ambos tipos de bloqueos pueden causar un evento de espera `Lock:Relation`. Las operaciones de mantenimiento de datos de la aplicación, como la actualización de una vista materializada, también pueden aumentar las consultas y transacciones bloqueadas.

Bloqueos en instancias de lectura

Puede haber un conflicto entre los bloqueos de relaciones que mantienen el escritor y los lectores. En la actualidad, solo los bloqueos de relaciones de `ACCESS EXCLUSIVE` se replican en instancias de lector. Sin embargo, el bloqueo de relaciones `ACCESS EXCLUSIVE` entrará en conflicto con cualquier bloqueo de relaciones `ACCESS SHARE` que mantenga el lector. Esto puede provocar un aumento de los eventos de espera de las relaciones de bloqueo en el lector.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Reducir el impacto de las instrucciones SQL que se bloquean
- Minimizar el efecto de las operaciones de mantenimiento
- Verificar los bloqueos de los lectores

Reducir el impacto de las instrucciones SQL que se bloquean

Para reducir el impacto de las instrucciones SQL que se bloquean, modifique el código de su aplicación cuando sea posible. A continuación se presentan dos técnicas comunes para reducir los bloqueos:

- Utilizar la opción `NOWAIT`: algunos comandos SQL, como las instrucciones `SELECT` y `LOCK`, admiten esta opción. La directiva `NOWAIT` cancela la consulta que solicita el bloqueo si éste no puede adquirirse inmediatamente. Esta técnica puede ayudar a evitar que una sesión bloqueada provoque una acumulación de sesiones bloqueadas detrás de ella.

  Por ejemplo: Supongamos que la transacción A espera un bloqueo que tiene la transacción B. Ahora, si B solicita un bloqueo en una tabla que está bloqueada por la transacción C, la transacción A podría quedar bloqueada hasta que la transacción C finalice. Pero si la transacción B utiliza un `NOWAIT` cuando solicita el bloqueo en C, puede fallar rápido y asegurar que la transacción A no tenga que esperar de forma indefinida.
- Utilice `SET lock_timeout`: establezca un valor de `lock_timeout` para limitar el tiempo que una sentencia SQL espera para adquirir un bloqueo en una relación. Si el bloqueo no se adquiere en el intervalo de tiempo de espera especificado, se cancelará la transacción que solicita el bloqueo. Establezca este valor en la sesión.

Minimizar el efecto de las operaciones de mantenimiento

Las operaciones de mantenimiento como `VACUUM` y `ANALYZE` son importantes. Le recomendamos que no las desactive ya que encontrará eventos de espera `Lock:Relation` relacionados con estas operaciones de mantenimiento. Los siguientes enfoques pueden minimizar el efecto de estas operaciones:

- Ejecute las operaciones de mantenimiento de forma manual durante las horas de menor actividad.
- Para reducir las esperas de `Lock:Relation` causadas por las tareas de autovacuum, realice los ajustes de autovacuum necesarios. Para obtener información sobre el ajuste de autovacuum,

consulte Trabajo con Autovacuum de PostgreSQL en Amazon RDS en la Guía del usuario de Amazon RDS.

Verificar los bloqueos de los lectores

Puede ver cómo las sesiones concurrentes en un escritor y los lectores se pueden bloquear mutuamente. Una forma de hacer esto consiste en ejecutar consultas que devuelvan el tipo de bloqueo y la relación. En la tabla encontrará una secuencia de consultas en dos sesiones simultáneas de este tipo, una sesión de escritor y una sesión de lector.

El proceso de repetición espera el tiempo que dura `max_standby_streaming_delay` antes de cancelar la consulta del lector. Como se muestra en el ejemplo, el tiempo de espera de bloqueo de 100 ms está muy por debajo del `max_standby_streaming_delay` predeterminado de 30 segundos. El tiempo de espera del bloqueo se agota antes de que sea un problema.

Evento de secuencia	Sesión	Comando o salida

Establece una variable de entorno denominada READER con el valor especificado e intenta conectarse a la instancia de base de datos con este punto de conexión.	Sesión de lector	Comando de la CLI:
		`export READER=aurorapg2.1234567890.us-west-1.rds.amazonaws.com`
		`psql -h $READER`
		Salida:
		`psql (15devel, server 10.14)`
		`Type "help" for help.`

Establece una variable de entorno denominada WRITER e intenta conectarse a la instancia de base de datos con este punto de conexión.	Sesión de escritor	Comando de la CLI:
		`export WRITER=aurorapg1.1234567890.us-west-1.rds.amazonaws.com`
		`psql -h $WRITER`

Evento de secuencia	Sesión	Comando o salida

		Salida:
		`psql (15devel, server 10.14)`
		`Type "help" for help.`

La sesión de escritor crea la tabla t1 en la instancia de escritor.	Sesión de escritor	Consulta de PostgreSQL:
		`postgres=> CREATE TABLE t1(b integer);`
		`CREATE TABLE`

Si no hay consultas en conflicto en el escritor, se adquirirá inmediatamente en el escritor el bloqueo de ACCESS EXCLUSIVE.	Sesión de escritor	Bloqueo de `ACCESS EXCLUSIVE` habilitado

La sesión de lector establece un intervalo de bloqueo de 100 milisegundos.	Sesión de lector	Consulta de PostgreSQL:
		`postgres=> SET lock_timeout=100;`
		`SET`

La sesión de lector intenta leer datos de la tabla t1 en la instancia de lector.	Sesión de lector	Consulta de PostgreSQL:
		`postgres=> SELECT * FROM t1;`
		Resultado de ejemplo:
		`b`
		`---`
		`(0 rows)`

Evento de secuencia	Sesión	Comando o salida

La sesión de escritor elimina la tabla t1.	Sesión de escritor	Consulta de PostgreSQL:
		`postgres=>  BEGIN;`
		`BEGIN`
		`postgres=>  DROP TABLE t1;`
		`DROP TABLE`
		`postgres=>`

Se agota el tiempo de espera de la consulta y se cancela en la sesión de lector.	Sesión de lector	Consulta de PostgreSQL:
		`postgres=>  SELECT * FROM t1;`
		Resultado de ejemplo:
		`ERROR:  canceling statement due to lock timeout`
		`LINE 1: SELECT * FROM t1;`
		`^`

Para determinar la causa del error, la sesión de lector consulta `pg_locks` y `pg_stat_activity` .	Sesión de lector	Consulta de PostgreSQL:
		`postgres=>  SELECT locktype, relation, mode, backend_type`
		`postgres=>  FROM pg_locks l, pg_stat_activity t1`
		`postgres=>  WHERE l.pid=t1.pid AND relation = 't1'::regclass::oid;`

Evento de secuencia	Sesión	Comando o salida

El resultado indica que el proceso de `aurora wal replay` mantiene un bloqueo de `ACCESS EXCLUSIVE` en la tabla t1.	Sesión de lector	Resultado de la consulta:
		`locktype | relation | mode | backend_type`
		`----------+----------+------------------+---------------`
		`relation | 68628525 | AccessExclusiveLock | aurora wal replay`
		`(1 row)`

Lock:transactionid

El evento `Lock:transactionid` se produce cuando una transacción espera un bloqueo a nivel de fila.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Context

El evento `Lock:transactionid` ocurre cuando una transacción intenta adquirir un bloqueo a nivel de fila que ya fue otorgado a una transacción que se está ejecutando al mismo tiempo. La sesión que muestra el evento de espera `Lock:transactionid` se encuentra bloqueada debido

a este bloqueo. Después de que la transacción bloqueada termine con una instrucción `COMMIT` o `ROLLBACK`, la transacción bloqueada puede continuar.

La semántica de control de concurrencia multiversión de Aurora PostgreSQL garantiza que los lectores no bloqueen a los escritores y que los escritores no bloqueen a los lectores. ara que se produzcan conflictos en las filas, las transacciones bloqueantes y bloqueadas deben emitir instrucciones conflictivas de los siguientes tipos:

- `UPDATE`
- `SELECT … FOR UPDATE`
- `SELECT … FOR KEY SHARE`

La instrucción `SELECT … FOR KEY SHARE` es un caso especial. La base de datos utiliza la cláusula `FOR KEY SHARE` para optimizar el rendimiento de la integridad referencial. Un bloqueo en una fila puede bloquear los comandos `INSERT`, `UPDATE` y `DELETE` en otras tablas que hacen referencia a la fila.

Causas probables del aumento de las esperas

Cuando este evento aparece más de lo normal, la causa suele ser instrucciones `UPDATE`, `SELECT … FOR UPDATE` o `SELECT … FOR KEY SHARE` combinadas con las siguientes condiciones.

Temas

- Gran concurrencia
- Inactividad en la transacción
- Transacciones de larga duración

Gran concurrencia

Aurora PostgreSQL puede utilizar una semántica de bloqueo pormenorizada por filas. La probabilidad de conflictos entre filas aumenta cuando se cumplen las siguientes condiciones:

- Una carga de trabajo de gran concurrencia compite por las mismas filas.
- Aumenta la concurrencia.

Inactividad en la transacción

A veces la columna `pg_stat_activity.state` muestra el valor `idle in transaction`. Este valor aparece para las sesiones que iniciaron una transacción, pero aún no han emitido un `COMMIT` o `ROLLBACK`. Si el valor de `pg_stat_activity.state` no se encuentra `active`, la consulta mostrada en `pg_stat_activity` es la más reciente en terminar de ejecutarse. La sesión de bloqueo no se encuentra en proceso activo de una consulta porque una transacción abierta está manteniendo un bloqueo.

Si una transacción inactiva adquirió un bloqueo entre filas, puede impedir que otras sesiones lo adquieran. Esta condición conduce a la aparición frecuente del evento de espera `Lock:transactionid`. Para diagnosticar el problema, examine la salida de `pg_stat_activity` y `pg_locks`.

Transacciones de larga duración

Las transacciones que se ejecutan durante mucho tiempo obtienen bloqueos durante mucho tiempo. Estos bloqueos de larga duración pueden bloquear la ejecución de otras transacciones.

Acciones

El bloqueo de filas es un conflicto entre las instrucciones `UPDATE`, `SELECT … FOR UPDATE`, o `SELECT … FOR KEY SHARE`. Antes de intentar una solución, averigüe cuándo se están ejecutando estas instrucciones en la misma fila. Utilice esta información para elegir una estrategia descrita en las siguientes secciones.

Temas

- Responder a la alta concurrencia
- Responder a las transacciones inactivas
- Responder a las transacciones de larga duración

Responder a la alta concurrencia

Si el problema es la concurrencia, pruebe una de las siguientes técnicas:

- Reduzca la concurrencia en la aplicación. Por ejemplo, disminuya el número de sesiones activas.
- Implemente un grupo de conexiones. Para saber cómo agrupar conexiones con RDS Proxy, consulte Amazon RDS Proxy para Aurora.

- Diseñe la aplicación o el modelo de datos para evitar las instrucciones `UPDATE` y `SELECT … FOR UPDATE` en conflicto. También puede disminuir el número de claves foráneas a las que se accede mediante instrucciones `SELECT … FOR KEY SHARE`.

Responder a las transacciones inactivas

Si `pg_stat_activity.state` muestra una transacción `idle in transaction`, utilice las siguientes estrategias:

- Active la confirmación automática siempre que sea posible. Este enfoque evita que las transacciones bloqueen otras transacciones mientras esperan un `COMMIT` o `ROLLBACK`.
- Busque rutas de código en las que falten `COMMIT`, `ROLLBACK` o `END`.
- Asegúrese de que la lógica de manejo de excepciones en su aplicación siempre tiene una ruta hacia `end of transaction` válido.
- Asegúrese de que su aplicación procesa los resultados de la consulta después de finalizar la transacción con `COMMIT` o `ROLLBACK`.

Responder a las transacciones de larga duración

Si las transacciones de larga duración provocan la aparición frecuente de `Lock:transactionid`, pruebe las siguientes estrategias:

- Mantenga los bloqueos de filas fuera de las transacciones de larga duración.
- Limite la longitud de las consultas mediante la implementación de confirmación automática siempre que sea posible.

Lock:tuple

El evento `Lock:tuple` se produce cuando un proceso backend espera adquirir un bloqueo sobre una tupla.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas

- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Context

El evento `Lock:tuple` indica que un backend espera adquirir un bloqueo sobre una tupla mientras otro backend mantiene un bloqueo conflictivo sobre la misma tupla. La siguiente tabla ilustra un escenario en el que las sesiones generan el evento `Lock:tuple`.

Tiem	Sesión 1	Sesión 2	Sesión 3
t1	Inicia una transacción.		
t2	Actualiza la fila 1.		
t3		Actualiza la fila 1. La sesión adquiere un bloqueo exclusivo sobre la tupla y luego espera a que la sesión 1 libere el bloqueo mediante la confirmación o reversión.	
t4			Actualiza la fila 1. La sesión espera a que la sesión 2 libere el bloqueo exclusivo en la tupla.

También puede simular este evento de espera con la herramienta de punto de referencia `pgbench`. Configure un alto número de sesiones concurrentes para actualizar la misma fila en una tabla con un archivo SQL personalizado.

Para obtener más información sobre los modos de bloqueo conflictivos, consulte E en la documentación de PostgreSQL. Para más información sobre `pgbench`, consulte pgbench en la documentación de PostgreSQL.

Causas probables del aumento de las esperas

Cuando este evento aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

- Un gran número de sesiones concurrentes están intentando adquirir un bloqueo conflictivo para la misma tupla al ejecutar instrucciones `UPDATE` o `DELETE`.
- Las sesiones altamente concurrentes se encuentran en ejecución con una instrucción `SELECT` que utiliza los modos de bloqueo `FOR UPDATE` o `FOR NO KEY UPDATE`.
- Varios factores hacen que la aplicación o los grupos de conexión abran más sesiones para ejecutar las mismas operaciones. A medida que nuevas sesiones intentan modificar las mismas filas, la carga de la base de datos puede aumentar y puede aparecer `Lock:tuple`.

Para más información, consulte Row-Level Locks en la documentación de PostgreSQL.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Investigue la lógica de su aplicación
- Encontrar la sesión bloqueadora
- Reducir la concurrencia cuando es alta
- Solucionar los cuellos de botella

Investigue la lógica de su aplicación

Verifique si una sesión del bloqueador se encuentra en el estado `idle in transaction` por mucho tiempo. Si es así, considere la posibilidad de finalizar la sesión del bloqueador como una solución a corto plazo. Puede utilizar la función `pg_terminate_backend`. Para más información sobre esta función, consulte Server Signaling Functions en la documentación de PostgreSQL.

Para una solución a largo plazo, haga lo siguiente:

- Ajuste la lógica de la aplicación.
- Utilice el parámetro `idle_in_transaction_session_timeout`. Este parámetro finaliza cualquier sesión con una transacción abierta que haya estado inactiva durante más tiempo del

especificado. Para más información, consulte Client Connection Defaults en la documentación de PostgreSQL.

- Utilice la confirmación automática en la medida de lo posible. Para más información, consulte SET AUTOCOMMIT en la documentación de PostgreSQL.

Encontrar la sesión bloqueadora

Mientras se produce el evento de espera `Lock:tuple`, identifique el bloqueador y la sesión bloqueada mediante la búsqueda de los bloqueos que dependen unos de otros. Para más información, consulte la Información sobre dependencia de bloqueos en el wiki de PostgreSQL. Para analizar eventos `Lock:tuple` pasados, utilice la función de Aurora `aurora_stat_backend_waits`.

El siguiente ejemplo muestra todas las sesiones, con un filtro `tuple` y ordenadas por `wait_time`.

```
--AURORA_STAT_BACKEND_WAITS
      SELECT a.pid,
             a.usename,
             a.app_name,
             a.current_query,
             a.current_wait_type,
             a.current_wait_event,
             a.current_state,
             wt.type_name AS wait_type,
             we.event_name AS wait_event,
             a.waits,
             a.wait_time
        FROM (SELECT pid,
                     usename,
                     left(application_name,16) AS app_name,
                     coalesce(wait_event_type,'CPU') AS current_wait_type,
                     coalesce(wait_event,'CPU') AS current_wait_event,
                     state AS current_state,
                     left(query,80) as current_query,
                     (aurora_stat_backend_waits(pid)).*
                FROM pg_stat_activity
               WHERE pid <> pg_backend_pid()
                 AND usename<>'rdsadmin') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we
WHERE we.event_name = 'tuple'
```

```
    ORDER BY a.wait_time;

  pid  | usename | app_name |                 current_query                  |
 current_wait_type | current_wait_event | current_state | wait_type | wait_event |
 waits | wait_time
-------+---------+----------+------------------------------------------------
+------------------+--------------------+--------------+-----------+------------
+-------+-----------
 32136 | sys     | psql     | /*session3*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000018
 11999 | sys     | psql     | /*session4*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000024
```

Reducir la concurrencia cuando es alta

El evento `Lock:tuple` puede producirse de manera frecuente, especialmente en un momento de carga de trabajo elevada. En esta situación, considere reducir la alta concurrencia para las filas muy ocupadas. A menudo, solo unas pocas filas controlan una cola o la lógica booleana, lo que hace que estas filas estén muy ocupadas.

Puede reducir la concurrencia mediante el uso de diferentes enfoques basados en los requisitos de la empresa, lógica de la aplicación y tipo de carga de trabajo. Por ejemplo, puede hacer lo siguiente:

- Rediseñar la lógica de la tabla y los datos para reducir la alta concurrencia.
- Cambiar la lógica de la aplicación para reducir la alta concurrencia entre filas.
- Aprovechar y rediseñar las consultas con bloqueos entre filas.
- Utilizar la cláusula `NOWAIT` con operaciones de reintento.
- Considerar el uso de control de concurrencia optimista y de lógica de bloqueo híbrida.
- Considerar la posibilidad de cambiar el nivel de aislamiento de la base de datos.

Solucionar los cuellos de botella

La `Lock:tuple` producirse con cuellos de botella, como el agotamiento de la CPU o el uso máximo del ancho de banda de Amazon EBS. Para reducir los cuellos de botella, considere los siguientes enfoques:

- Escalar verticalmente el tipo de clase de instancia.

- Optimizar las consultas que consumen muchos recursos.
- Cambiar la lógica de la aplicación.
- Archivar los datos a los que rara vez se accede.

LWLock:buffer_content (BufferContent)

El evento `LWLock:buffer_content` ocurre cuando una sesión espera para leer o escribir una página de datos en memoria mientras otra sesión tiene esa página bloqueada para escribir. En Aurora PostgreSQL 13 y versiones posteriores, este evento de espera se llama `BufferContent`.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Context

Para leer o manipular datos, PostgreSQL accede a ellos a través de búferes de memoria compartida. Para leer del búfer, un proceso obtiene un bloqueo ligero (LWLock) sobre el contenido del búfer en modo compartido. Para escribir en el búfer, obtiene ese bloqueo en modo exclusivo. Los bloqueos compartidos permiten a otros procesos adquirir simultáneamente bloqueos compartidos sobre ese contenido. Los bloqueos exclusivos impiden que otros procesos obtengan cualquier tipo de bloqueo sobre él.

El evento `LWLock:buffer_content` (`BufferContent`) indica que varios procesos intentan obtener un bloqueo sobre el contenido de un búfer específico.

Causas probables del aumento de las esperas

Cuando el evento `LWLock:buffer_content` (`BufferContent`) aparece más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas típicas son las siguientes:

Aumento de las actualizaciones simultáneas de los mismos datos

Puede haber un aumento en el número de sesiones concurrentes con consultas que actualizan el mismo contenido del búfer. Esta contención puede ser más pronunciada en tablas con muchos índices.

Los datos de carga de trabajo no están en la memoria

Cuando los datos que la carga de trabajo activa está procesando no están en memoria, estos eventos de espera pueden aumentar. Este efecto se debe a que los procesos que mantienen bloqueos pueden mantenerlos durante más tiempo mientras hacen operaciones de E/S en disco.

Uso excesivo de restricciones de clave externa

Las restricciones de clave externa pueden aumentar el tiempo que un proceso mantiene un bloqueo de contenido de búfer. Este efecto se debe a que las operaciones de lectura requieren un bloqueo de contenido de búfer compartido en la clave referenciada mientras se actualiza dicha clave.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera. Puede identificar los eventos `LWLock:buffer_content` (`BufferContent`) mediante Información sobre rendimiento de Amazon RDS o consultar la vista `pg_stat_activity`.

Temas

- Mejorar la eficiencia en memoria
- Reducir el uso de restricciones de clave externa
- Eliminar los índices que no se utilizan

Mejorar la eficiencia en memoria

Para aumentar la posibilidad de que los datos de la carga de trabajo activa estén en memoria, particione las tablas o escale verticalmente su clase de instancia. Para obtener información acerca de las clases de instancia de base de datos, consulte Clases de instancia de base de datos de Amazon Aurora.

Reducir el uso de restricciones de clave externa

Examine las cargas de trabajo que experimentan un elevado número de eventos de espera `LWLock:buffer_content` (`BufferContent`) para comprobar el uso de las restricciones de clave externa. Elimine las restricciones de clave externa innecesarias.

Eliminar los índices que no se utilizan

Para las cargas de trabajo que experimentan un gran número de eventos de espera `LWLock:buffer_content` (`BufferContent`), identifique los índices que no se utilizan y elimínelos.

LWLock:buffer_mapping

Este evento se produce cuando una sesión espera asociar un bloque de datos con un búfer en el grupo de búferes compartidos.

> **Note**
>
> Este evento aparece como `LWLock:buffer_mapping` en la versión 12 de Aurora PostgreSQL e inferior, y `LWLock:BufferMapping` en la versión 13 y posterior.

Temas

- Versiones del motor admitidas
- Context
- Causas
- Acciones

Versiones del motor admitidas

La información del evento de espera es relevante para Aurora PostgreSQL versión 9.6 y posterior.

Context

El grupo de búferes compartidos es un área de memoria de Aurora PostgreSQL que contiene todas las páginas que se están o se estaban utilizando por los procesos. Cuando un proceso necesita una página, lee la página en el grupo de búferes compartidos. El parámetro `shared_buffers` establece el tamaño del búfer compartido y reserva un área de memoria para almacenar las páginas de tablas

e índices. Si cambia este parámetro, asegúrese de reiniciar la base de datos. Para obtener más información, consulte Búferes compartidos.

El evento de espera `LWLock:buffer_mapping` ocurre en los siguientes escenarios:

- Un proceso busca una página en la tabla de búferes y adquiere un bloqueo de asignación de búferes compartidos.
- Un proceso carga una página en el grupo de búferes y adquiere un bloqueo de asignación de búferes exclusivo.
- Un proceso elimina una página del grupo y adquiere un bloqueo de asignación de búferes exclusivo.

Causas

Cuando este evento aparece más de lo normal, lo que puede indicar un problema de rendimiento, la base de datos entra y sale del grupo de búferes compartidos. Las causas típicas son las siguientes:

- Consultas grandes
- Índices y tablas sobrecargados
- Escaneos completos de tablas
- Un tamaño del grupo compartido menor que el conjunto de trabajo

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Monitorear las métricas relacionadas con el buffer
- Evaluar la estrategia de indexación
- Reducir el número de búferes que deben ser asignados rápidamente

Monitorear las métricas relacionadas con el buffer

Cuando las esperas de `LWLock:buffer_mapping` se disparan, hay que investigar la tasa de aciertos del búfer. Puede utilizar estas métricas para comprender mejor lo que ocurre en la caché del búfer. Examina las siguientes métricas:

`BufferCacheHitRatio`

Esta métrica de Amazon CloudWatch mide el porcentaje de solicitudes que se atienden en la caché del búfer de una instancia de base de datos en su clúster de base de datos. Es posible que esta métrica disminuya en el periodo previo al evento de espera `LWLock:buffer_mapping`.

`blks_hit`

Esta métrica del contador de Información sobre rendimiento indica el número de bloques que se recuperaron del grupo de búferes compartidos. Después de que aparezca el evento de espera `LWLock:buffer_mapping`, se puede observar un pico en `blks_hit`.

`blks_read`

Esta métrica del contador de Información sobre rendimiento indica el número de bloques que requirieron E/S para leerse en el grupo de búferes compartidos. Puede observar un pico en `blks_read` en el periodo previo al evento de espera `LWLock:buffer_mapping`.

Evaluar la estrategia de indexación

Para confirmar que la estrategia de indexación no disminuye el rendimiento, verifique lo siguiente:

Sobrecarga del índice

Asegúrese de que el índice y la sobrecarga de la tabla no provocan la lectura de páginas innecesarias en el búfer compartido. Si las tablas contienen filas que no se utilizan, considere la posibilidad de archivar los datos y eliminar las filas de las tablas. A continuación, puede reconstruir los índices para las tablas redimensionadas.

Índices para consultas de uso frecuente

Para determinar si cuenta con los índices óptimos, monitoree las métricas del motor de base de datos en Información sobre rendimiento. La métrica `tup_returned` muestra el número de filas leídas. La métrica `tup_fetched` muestra el número de filas devueltas al cliente. Si `tup_returned` es mucho mayor que `tup_fetched`, es posible que los datos no estén bien indexados. Además, es posible que las estadísticas de la tabla no se encuentren actualizadas.

Reducir el número de búferes que deben ser asignados rápidamente

Para reducir los eventos de espera de `LWLock:buffer_mapping`, intente reducir el número de búferes que se deben asignar de forma rápida. Una estrategia es hacer operaciones por lotes más pequeños. Se pueden conseguir lotes más pequeños por medio de la partición de las tablas.

LWLock:BufferIO (IPC:BufferIO)

El evento `LWLock:BufferIO` ocurre cuando Aurora PostgreSQL o RDS for PostgreSQL espera que otros procesos terminen sus operaciones de entrada/salida (E/S) cuando intentan acceder a una página de forma simultánea. Su propósito es que la misma página se lea en el búfer compartido.

Temas

- Versiones del motor relevantes
- Contexto
- Causas
- Acciones

Versiones del motor relevantes

Esta información de eventos de espera es relevante para todas las versiones de Aurora PostgreSQL. Para Aurora PostgreSQL 12 y versiones anteriores, este evento de espera se denomina lwlock:buffer_io, mientras que en la versión 13 de Aurora PostgreSQL se denomina lwlock:bufferio. A partir de la versión 14 de Aurora PostgreSQL, el evento de espera BufferIO se movió de tipo de evento de espera `LWLock` a `IPC` (IPC:bufferIO).

Contexto

Cada búfer compartido tiene un bloqueo de E/S que está asociado con el evento de espera `LWLock:BufferIO`, cada vez que un bloque (o una página) se tiene que recuperar fuera del grupo de búferes compartidos.

Este bloqueo se utiliza para manejar múltiples sesiones que requieren acceso al mismo bloque. Este bloque se tiene que leer desde fuera del grupo de búferes compartidos, que se define con el parámetro `shared_buffers`.

Tan pronto como la página se lee dentro del grupo de búferes compartidos, el bloqueo `LWLock:BufferIO` se libera.

> **Note**
>
> El evento de espera `LWLock:BufferIO` precede al evento de espera IO:DataFileRead. El evento de espera `IO:DataFileRead` se produce mientras se leen datos del almacenamiento.

Para obtener más información sobre los bloqueos ligeros, consulte Información general sobre los bloqueos.

Causas

Las causas más comunes para que el evento `LWLock:BufferIO` aparezca en el máximo de esperas son las siguientes:

- Varios backends o conexiones que intentan acceder a la misma página que también tiene pendiente una operación de E/S
- La relación entre el tamaño del grupo de búferes compartidos (definido por el parámetro `shared_buffers`) y el número de búferes que necesita la carga de trabajo actual
- El tamaño del grupo de búferes compartidos no está bien equilibrado con el número de páginas que se desalojan por otras operaciones
- Índices grandes o sobrecargados que requieren que el motor lea más páginas de las necesarias en el grupo de búferes compartidos
- La falta de índices obliga al motor de la base de datos a leer más páginas de las necesarias en las tablas
- Picos repentinos de conexiones a la base de datos que intentan hacer operaciones en la misma página

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera:

- Observe las métricas de Amazon CloudWatch en busca de una correlación entre los descensos bruscos de los eventos de espera `BufferCacheHitRatio` y `LWLock:BufferIO`. Este efecto a veces significa que tiene una configuración de búferes compartidos pequeña. Puede que tenga que aumentarla o escalar verticalmente la clase de instancia de base de datos. Puede dividir su carga de trabajo en más nodos de lectura.
- Verifique si tiene índices sin utilizar y elimínelos.
- Utilice tablas particionadas (que también tengan índices particionados). Esto ayuda a mantener un bajo nivel de reordenación de índices y reduce su impacto.
- Evite indexar columnas innecesariamente.
- Evite los picos repentinos de conexión a la base de datos, utilice un grupo de conexiones.
- Limite el número máximo de conexiones a la base de datos como práctica recomendada.

LWLock:lock_manager

Este evento ocurre cuando el motor de Aurora PostgreSQL mantiene el área de memoria del bloqueo compartido para asignar, verificar y desasignar un bloqueo cuando no es posible un bloqueo de ruta rápida.

Temas

- Versiones del motor admitidas
- Context
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

La información del evento de espera es relevante para Aurora PostgreSQL versión 9.6 y posterior.

Context

Cuando se emite una instrucción SQL, Aurora PostgreSQL registra bloqueos para proteger la estructura, datos e integridad de la base de datos durante las operaciones simultáneas. El motor puede lograr este objetivo con un bloqueo de ruta rápido o con un bloqueo de ruta que no es rápido. Un bloqueo de ruta que no es rápido es más caro y crea más sobrecarga que un bloqueo de ruta rápido.

Bloqueo rápido de la ruta

Para reducir la sobrecarga de los bloqueos que se toman y liberan con frecuencia, pero que rara vez entran en conflicto, los procesos del backend pueden utilizar el bloqueo de ruta rápido. La base de datos utiliza este mecanismo para los bloqueos que cumplen los siguientes criterios:

- Utilizan el método de bloqueo DEFAULT.
- Representan un bloqueo en una relación de la base de datos y no en una relación compartida.
- Son bloqueos débiles que probablemente no entren en conflicto.
- El motor puede verificar rápidamente que no pueden existir bloqueos conflictivos.

El motor no puede utilizar el bloqueo de ruta rápida cuando se cumple alguna de las siguientes condiciones:

- El bloqueo no cumple los criterios anteriores.
- No hay más ranuras disponibles para el proceso de backend.

Para más información sobre el bloqueo de ruta rápida, consulte fast path en el README del administrador de bloqueos de PostgreSQL y pg-locks en la documentación de PostgreSQL.

Ejemplo de un problema de escalado para el administrador de bloqueos

En este ejemplo, una tabla con el nombre `purchases` almacena cinco años de datos, particionados por día. Cada partición tiene dos índices. Se produce la siguiente secuencia de eventos:

1. Se consultan los datos de muchos días, lo que requiere que la base de datos lea muchas particiones.
2. La base de datos crea una entrada de bloqueo para cada partición. Si los índices de las particiones forman parte de la ruta de acceso del optimizador, la base de datos también crea una entrada de bloqueo para ellos.
3. Cuando el número de entradas de bloqueo solicitadas para el mismo proceso backend es superior a 16, que es el valor `FP_LOCK_SLOTS_PER_BACKEND`, el administrador de bloqueos utiliza el método de bloqueo de ruta no rápida.

Las aplicaciones modernas pueden tener cientos de sesiones. Si las sesiones simultáneas consultan la base de datos principal sin una poda adecuada de las particiones, la base de datos puede crear cientos o incluso miles de bloqueos de ruta no rápida. Normalmente, cuando esta simultaneidad es mayor que el número de vCPU, aparece el evento de espera `LWLock:lock_manager`.

> **Note**
>
> El evento de espera `LWLock:lock_manager` no está relacionado con el número de particiones o índices en un esquema de base de datos. En cambio, está relacionado con el número de bloqueos de rutas no rápidas que la base de datos debe controlar.

Causas probables del aumento de las esperas

Cuando el evento de espera `LWLock:lock_manager` ocurre más de lo normal, lo que posiblemente indica un problema de rendimiento, las causas más probables de los picos repentinos son las siguientes:

- Las sesiones activas simultáneas ejecutan consultas que no utilizan bloqueos de ruta rápida. Estas sesiones también exceden el máximo de vCPU.
- Un gran número de sesiones activas simultáneas acceden a una tabla con muchas particiones. Cada partición tiene múltiples índices.
- La base de datos está experimentando una tormenta de conexiones. De forma predeterminada, algunas aplicaciones y software de grupo de conexiones crean más conexiones cuando la base de datos es lenta. Esta práctica empeora el problema. Ajuste el software de grupo de conexiones para que no se produzcan tormentas de conexiones.
- Un gran número de sesiones consultan una tabla principal sin borrar particiones.
- Un lenguaje de definición de datos (DDL), un lenguaje de manipulación de datos (DML) o un comando de mantenimiento bloquea exclusivamente una relación ocupada o tuplas a las que se accede o modifica con frecuencia.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

Temas

- Utilizar la poda de particiones
- Eliminar índices innecesarios
- Ajustar sus consultas para el bloqueo rápido de rutas
- Ajustar otros eventos de espera
- Reducir los cuellos de botella del hardware
- Utilizar un grupo de conexiones
- Actualice la versión Aurora PostgreSQL

Utilizar la poda de particiones

La poda de particiones es una estrategia de optimización de consultas que excluye las particiones innecesarias de los escaneos de tablas, lo que mejora el rendimiento. La poda de particiones está activada de forma predeterminada. Si está desactivada, actívela de la siguiente manera.

```
SET enable_partition_pruning = on;
```

Las consultas pueden aprovechar la poda de particiones cuando la cláusula `WHERE` contiene la columna que se utiliza para la partición. Para más información, consulte Partition Pruningc en la documentación de PostgreSQL.

Eliminar índices innecesarios

Es posible que la base de datos contenga índices que no se utilicen o que se utilicen muy poco. Si es así, considere eliminarlos. Haga una de estas dos operaciones:

- Aprenda a encontrar índices innecesarios al leer Índices no utilizados en el wiki de PostgreSQL.
- Ejecute PG Collector. Este script SQL recopila información de la base de datos y la presenta en un informe HTML consolidado. Verifique la sección “Índices no utilizados”. Para más información, consulte pg-collector en el repositorio GitHub de AWS Labs.

Ajustar sus consultas para el bloqueo rápido de rutas

Para averiguar si las consultas utilizan el bloqueo de ruta rápida, consulte la columna `fastpath` en la tabla `pg_locks`. Si las consultas no utilizan el bloqueo de ruta rápida, intente reducir el número de relaciones por consulta a menos de 16.

Ajustar otros eventos de espera

Si `LWLock:lock_manager` es el primero o el segundo en la lista de esperas principales, verifique si los siguientes eventos de espera también aparecen en la lista:

- `Lock:Relation`
- `Lock:transactionid`
- `Lock:tuple`

Si los eventos anteriores aparecen en primer lugar en la lista, considere la posibilidad de ajustar estos eventos de espera en primer lugar. Estos eventos pueden ser un controlador para `LWLock:lock_manager`.

Reducir los cuellos de botella del hardware

Es posible que tenga un cuello de botella en el hardware, como el agotamiento de la CPU o el uso máximo de su ancho de banda de Amazon EBS. En estos casos, considere la posibilidad de reducir los cuellos de botella de hardware. Considere las siguientes acciones:

- Escalar verticalmente la clase de instancia.
- Optimizar las consultas que consumen grandes cantidades de CPU y memoria.
- Cambiar la lógica de su aplicación.
- Archivar los datos.

Para más información sobre la CPU, memoria y ancho de banda de red de EBS, consulte Tipos de instancias de Amazon RDS.

Utilizar un grupo de conexiones

Si el número total de conexiones activas supera el máximo de vCPU, más procesos del sistema operativo requieren CPU de lo que su tipo de instancia puede admitir. En este caso, considere la posibilidad de utilizar o ajustar un grupo de conexiones. Para más información sobre las vCPU de su tipo de instancia, consulte Tipos de instancias de Amazon RDS.

Para más información sobre la agrupación de conexiones, consulte los siguientes recursos:

- Amazon RDS Proxy para Aurora
- pgbouncer
- Connection Pools and Data Sources en la documentación de PostgreSQL

Actualice la versión Aurora PostgreSQL

Si la versión actual de Aurora PostgreSQL es inferior a 12, actualice a la versión 12 o posterior. Las versiones 12 y 13 de PostgreSQL tienen un mecanismo de partición mejorado. Para más información sobre la versión 12, consulte las Notas de la versión 12.0 de PostgreSQL. Para más información sobre la actualización de Aurora PostgreSQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL.

LWLock:MultiXact

Los eventos de espera `LWLock:MultiXactMemberBuffer`, `LWLock:MultiXactOffsetBuffer`, `LWLock:MultiXactMemberSLRU` y `LWLock:MultiXactOffsetSLRU` indican que una sesión está esperando recuperar una lista de transacciones que modifican la misma fila de una tabla determinada.

- `LWLock:MultiXactMemberBuffer`: un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para un miembro de multixact.

- `LWLock:MultiXactMemberSLRU`: un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) de un miembro de multixact.
- `LWLock:MultiXactOffsetBuffer`: un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para un desplazamiento de multixact.
- `LWLock:MultiXactOffsetSLRU`: un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) de un desplazamiento de multixact.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Contexto

Un multixact es una estructura de datos que almacena una lista de identificadores de transacciones (XID) que modifican la misma fila de la tabla. Cuando una sola transacción hace referencia a una fila de una tabla, el identificador de la transacción se almacena en la fila de encabezados de la tabla. Cuando varias transacciones hacen referencia a la misma fila de una tabla, la lista de identificadores de transacciones se almacena en la estructura de datos multixact. Los eventos de espera multixact indican que una sesión está recuperando de la estructura de datos la lista de transacciones que hacen referencia a una fila determinada de una tabla.

Causas probables del aumento del tiempo de espera

Estas son tres causas comunes del uso de multixact:

- Subtransacciones desde puntos de guardado explícitos: al crear explícitamente un punto de guardado en sus transacciones, se generan nuevas transacciones para la misma fila. Por ejemplo, usar `SELECT FOR UPDATE`, luego `SAVEPOINT` y luego `UPDATE`.

Algunos controladores, asignadores relacionales de objetos (ORM) y capas de abstracción tienen opciones de configuración para ajustar automáticamente todas las operaciones con puntos de guardado. Esto puede generar muchos eventos de espera multixact en algunas cargas de trabajo. La opción `autosave` del controlador JDBC de PostgreSQL es un ejemplo de esto. Para obtener más información, consulte pgJDBC en la documentación de JDBC de PostgreSQL. Otro ejemplo es el controlador ODBC de PostgreSQL y su opción `protocol`. Para más información, consulte psqlODBC Configuration Options (Opciones de configuración de psqlODBC) en la documentación del controlador ODBC de PostgreSQL.

- Subtransacciones desde cláusulas PL/pgSQL EXCEPTION: cada cláusula `EXCEPTION` que escriba en sus funciones o procedimientos PL/pgSQL crea un `SAVEPOINT` interno.
- Claves externas: varias transacciones adquieren un bloqueo compartido sobre el registro principal.

Cuando una fila determinada se incluye en una operación de transacciones múltiples, el procesamiento de la fila requiere recuperar los identificadores de transacción de los listados de `multixact`. Si las búsquedas no pueden obtener el multixact de la memoria caché, la estructura de datos debe leerse desde la capa de almacenamiento de Aurora. Esta E/S desde el almacenamiento significa que las consultas SQL pueden tardar más tiempo. Las pérdidas de memoria caché pueden empezar a producirse cuando hay un uso intensivo debido a la gran cantidad de transacciones múltiples. Todos estos factores contribuyen a un aumento de este evento de espera.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera. Algunas de estas acciones pueden ayudar a reducir inmediatamente los eventos de espera. Sin embargo, es posible que otras requieran cierta investigación y corrección para aumentar la carga de trabajo.

Temas

- Realización de inmovilización por vacuum en tablas con este evento de espera
- Aumente la frecuencia del vaciado automático en las tablas con este evento de espera
- Aumento de los parámetros de memoria
- Reducción de transacciones de larga duración
- Acciones a largo plazo

Realización de inmovilización por vacuum en tablas con este evento de espera

Si este evento de espera se intensifica repentinamente y afecta a su entorno de producción, puede usar cualquiera de los siguientes métodos temporales para reducir su recuento.

- Utilice VACUUM FREEZE en la tabla o partición de tabla afectada para resolver el problema inmediatamente. Para obtener más información, consulte VACUUM.
- Utilice la cláusula VACUUM (FREEZE, INDEX_CLEANUP FALSE) para realizar un vaciado omitiendo los índices. Para obtener más información, consulte Vaciado de una tabla lo más rápido posible.

Aumente la frecuencia del vaciado automático en las tablas con este evento de espera

Tras escanear todas las tablas de todas las bases de datos, VACUUM eliminará finalmente los valores multixact y avanzará los valores multixact más antiguos. Para obtener más información, consulte Multixacts y Wraparound. Para reducir al mínimo los eventos de espera de LWLock:MultiXact, debe ejecutar VACUUM tantas veces como sea necesario. Para ello, asegúrese de que el VACUUM de su clúster de base de datos PostgreSQL de Aurora esté configurado de forma óptima.

Si al utilizar VACUUM FREEZE en la tabla o partición de tabla afectada se resuelve el problema del evento de espera, le recomendamos que utilice un planificador, por ejemplo, `pg_cron`, para ejecutar el VACUUM en lugar de ajustar el vaciado automático en el nivel de instancia.

Para que el autovacuum se realice con mayor frecuencia, puede reducir el valor del parámetro de almacenamiento `autovacuum_multixact_freeze_max_age` en la tabla afectada. Para obtener más información, consulte autovacuum_multixact_freeze_max_age.

Aumento de los parámetros de memoria

Puede optimizar el uso de la memoria para las cachés multixact ajustando los siguientes parámetros. Esta configuración controla la cantidad de memoria que se reserva para estas cachés, lo que puede ayudar a reducir los eventos de espera multixact en la carga de trabajo. Recomendamos empezar con los siguientes valores:

Para Aurora PostgreSQL 17 y posteriores:

- `multixact_offset_buffers` = 128
- `multixact_member_buffers` = 256

Para Aurora PostgreSQL 16 y anteriores:

- `multixact_offsets_cache_size` = 128
- `multixact_members_cache_size` = 256

> **Note**
>
> En Aurora PostgreSQL 17, los nombres de los parámetros se cambiaron de `multixact_offsets_cache_size` a `multixact_offset_buffers` y de `multixact_members_cache_size` a `multixact_member_buffers` para alinearse con PostgreSQL 17 de la comunidad.

Puede establecer estos parámetros en el nivel del clúster para que todas las instancias del clúster se mantengan coherentes. Le recomendamos que pruebe y ajuste los valores para que se adapten mejor a los requisitos de carga de trabajo específicos y a la clase de instancia. Debe reiniciar la instancia de escritor para que los cambios de parámetro tengan efecto.

Los parámetros se expresan en términos de entradas de caché multixact. Cada entrada de caché utiliza `8 KB` de memoria. Para calcular la memoria total reservada, multiplique el valor de cada parámetro por `8 KB`. Por ejemplo, si establece un parámetro en 128, la memoria reservada total sería `128 * 8 KB = 1 MB`.

Reducción de transacciones de larga duración

Las transacciones de larga duración provocan que vacuum retenga la información hasta que se confirme la transacción o hasta que se cierre la transacción de solo lectura. Le recomendamos que supervise y administre de forma proactiva las transacciones de larga duración. Para obtener más información, consulte La base de datos lleva mucho tiempo inactiva en la conexión de la transacción. Intente modificar la aplicación para evitar o minimizar el uso de transacciones de larga duración.

Acciones a largo plazo

Examine su carga de trabajo para descubrir la causa de la propagación de multixact. Debe solucionar el problema para escalar su carga de trabajo y reducir el evento de espera.

- Debe analizar el DDL (lenguaje de definición de datos) utilizado para crear las tablas. Asegúrese de que las estructuras y los índices de las tablas estén bien diseñados.

- Cuando las tablas afectadas tengan claves externas, determine si son necesarias o si hay otra forma de reforzar la integridad referencial.
- Cuando una tabla tiene índices no utilizados de gran tamaño, es posible que el autovacuum no se adapte a la carga de trabajo y que impida su ejecución. Para evitarlo, compruebe si hay índices no utilizados y elimínelos por completo. Para obtener más información, consulte Administración de autovacuum con índices de gran tamaño.
- Reduzca el uso de puntos de guardado en sus transacciones.

LWLock:pg_stat_statements

El evento de espera LWLock:pg_stat_statements se produce cuando la extensión `pg_stat_statements` toma un bloqueo exclusivo en la tabla hash que rastrea las instrucciones SQL. Esto ocurre en las siguientes situaciones:

- Cuando el número de instrucciones rastreadas alcanza el valor del parámetro `pg_stat_statements.max` configurado y es necesario dejar espacio para más entradas, la extensión realiza una clasificación del número de llamadas, elimina el 5 % de las instrucciones menos ejecutadas y vuelve a rellenar el hash con las entradas restantes.
- Cuando `pg_stat_statements` realiza una operación de `garbage collection` en el archivo `pgss_query_texts.stat` del disco y lo vuelve a escribir.

Temas

- Versiones del motor admitidas
- Contexto
- Causas probables del aumento del tiempo de espera
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Contexto

Descripción de la extensión pg_stat_statements: la extensión pg_stat_statements rastrea las estadísticas de ejecución de instrucción SQL en una tabla hash. La extensión rastrea las

instrucciones SQL hasta el límite definido por el parámetro `pg_stat_statements.max`. Este parámetro determina el número máximo de instrucciones que se pueden rastrear, lo que corresponde al número máximo de filas en la vista de pg_stat_statements.

Persistencia de las estadísticas de las instrucciones: la extensión mantiene las estadísticas de las instrucciones a través de reinicios de instancia mediante:

- Escritura de datos en un archivo llamado pg_stat_statements.stat
- Uso del parámetro pg_stat_statements.save para controlar el comportamiento de persistencia

Cuando pg_stat_statements.save se establece en:

- activado (predeterminado): las estadísticas se guardan al apagar el servidor y se vuelven a cargar al iniciar el servidor
- desactivado (predeterminado): las estadísticas no se guardan al apagar el servidor ni se vuelven a cargar al iniciar el servidor

Almacenamiento de texto de consulta: la extensión almacena el texto de las consultas rastreadas en un archivo denominado `pgss_query_texts.stat`. Este archivo puede crecer hasta duplicar el tamaño medio de todas las instrucciones SQL rastreadas antes de que se produzca la recopilación de elementos no utilizados. La extensión requiere un bloqueo exclusivo en la tabla de hash durante las operaciones de limpieza y reescritura del archivo `pgss_query_texts.stat`.

Proceso de desasignación de instrucciones: cuando el número de instrucciones rastreadas alcanza el límite `pg_stat_statements.max` y es necesario realizar un seguimiento de las nuevas, la extensión:

- Toma un bloqueo exclusivo (LWLock:pg_stat_statements) en la tabla de hash.
- Carga los datos existentes en la memoria local.
- Realiza una clasificación rápida en función del número de llamadas.
- Elimina las instrucciones menos solicitadas (el 5 % inferior).
- Vuelve a rellenar la tabla hash con las entradas restantes.

Supervisión de la desasignación de instrucciones: en PostgreSQL 14 y versiones posteriores, puede supervisar la desasignación de las instrucciones mediante la vista pg_stat_statements_info. Esta

vista incluye una columna dealloc que muestra cuántas veces se desasignaron las instrucciones para dejar espacio a otras nuevas

Si la desasignación de las instrucciones se produce con frecuencia, se realizará con más frecuencia la recopilación de elementos no utilizados del archivo `pgss_query_texts.stat` en el disco.

Causas probables del aumento del tiempo de espera

Entre las causas típicas del aumento de las esperas de `LWLock:pg_stat_statements` se incluyen las siguientes:

- Un aumento en el número de consultas únicas utilizadas por la aplicación.
- El valor del parámetro `pg_stat_statements.max` es pequeño en comparación con el número de consultas únicas que se utilizan.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera. Puede identificar los eventos de `LWLock:pg_stat_statements` mediante Información de rendimiento de Amazon RDS o consultando la vista de `pg_stat_activity`.

Ajuste los siguientes parámetros `pg_stat_statements` para controlar el comportamiento de seguimiento y reducir los eventos de espera de las instrucciones LWLock:pg_stat_statements.

Temas

- Desactivación del parámetro pg_stat_statements.track
- Aumento del parámetro pg_stat_statements.max
- Desactivación del parámetro pg_stat_statements.track_utility

Desactivación del parámetro pg_stat_statements.track

Si el evento de espera de LWLock:pg_stat_statements está afectando negativamente al rendimiento de la base de datos y se requiere una solución rápida antes de continuar con el análisis de la vista de `pg_stat_statements` para identificar la causa raíz, el parámetro `pg_stat_statements.track` se puede desactivar configurándolo en `none`. Esto desactivará la recopilación de estadísticas de instrucciones.

Aumento del parámetro pg_stat_statements.max

Para reducir la desasignación y minimizar la recopilación de elementos no utilizados del archivo `pgss_query_texts.stat` en el disco, aumente el valor del parámetro `pg_stat_statements.max`. El valor predeterminado es `5,000`.

> **Note**
>
> El parámetro `pg_stat_statements.max` está estático. Debe reiniciar la instancia de base de datos para aplicar los cambios a este parámetro.

Desactivación del parámetro pg_stat_statements.track_utility

Puede analizar la vista de pg_stat_statements para determinar qué comandos de utilidad consumen la mayor cantidad de recursos rastreados por `pg_stat_statements`.

El parámetro `pg_stat_statements.track_utility` controla si el módulo realiza un seguimiento de los comandos de utilidad, que incluyen todos los comandos excepto SELECT, INSERT, UPDATE, DELETE y MERGE. Este parámetro está establecido en on de forma predeterminada.

Por ejemplo, cuando la aplicación utiliza muchas consultas de puntos de guardado, que son intrínsecamente únicas, puede aumentar la desasignación de instrucciones. Para abordar este problema, puede desactivar el parámetro `pg_stat_statements.track_utility` para impedir que `pg_stat_statements` realice un seguimiento de las consultas de puntos de guardado.

> **Note**
>
> El parámetro `pg_stat_statements.track_utility` es un parámetro dinámico. Puede cambiar su valor sin necesidad de reiniciar la instancia de la base de datos.

Example Ejemplo de consultas de punto de guardado únicas en pg_stat_statements

```
                    query                         |       queryid
--------------------------------------------------+---------------------
 SAVEPOINT JDBC_SAVEPOINT_495701                  | -7249565344517699703
 SAVEPOINT JDBC_SAVEPOINT_1320                    | -1572997038849006629
 SAVEPOINT JDBC_SAVEPOINT_26739                   |  54791337410474486
```

```
SAVEPOINT JDBC_SAVEPOINT_1294466                | 8170064357463507593
ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_65016      | -33608214779996400
SAVEPOINT JDBC_SAVEPOINT_14185                  | -2175035613806809562
SAVEPOINT JDBC_SAVEPOINT_45837                  | -6201592986750645383
SAVEPOINT JDBC_SAVEPOINT_1324                   | 6388797791882029332
```

PostgreSQL 17 presenta varias mejoras para el seguimiento de comandos de utilidades:

- Los nombres de los puntos de guardado ahora se muestran como constantes.
- Los ID de transacción globales (GID) de los comandos de confirmación de dos fases ahora se muestran como constantes.
- Los nombres de las instrucciones DEALLOCATE se muestran como constantes.
- Los parámetros CALL ahora se muestran como constantes.

Timeout:PgSleep

El evento `Timeout:PgSleep` ocurre cuando un proceso del servidor llama a la función `pg_sleep` y espera a que el tiempo de espera expire.

Temas

- Versiones del motor admitidas
- Causas probables del aumento de las esperas
- Acciones

Versiones del motor admitidas

Esta información de eventos de espera es compatible con todas las versiones de Aurora PostgreSQL.

Causas probables del aumento de las esperas

Este evento de espera ocurre cuando una aplicación, función almacenada o usuario emite una sentencia SQL que llama a una de las siguientes funciones:

- `pg_sleep`
- `pg_sleep_for`
- `pg_sleep_until`

Las funciones anteriores retrasan la ejecución hasta que transcurra el número de segundos especificado. Por ejemplo, `SELECT pg_sleep(1)` hace una pausa de 1 segundo. Para más información, consulte Delaying Execution en la documentación de PostgreSQL.

Acciones

Identifique la sentencia que estaba ejecutando la función `pg_sleep`. Determine si el uso de la función es adecuado.

Ajuste de Aurora PostgreSQL con información proactiva de Amazon DevOps Guru

La información proactiva de DevOps Guru detecta las condiciones en sus clústeres de bases de datos de Aurora PostgreSQL que pueden causar problemas y le permiten conocerlos antes de que se produzcan. La información proactiva puede alertarle sobre algún elemento que lleve tiempo inactivo en la conexión de la transacción. Para obtener más información sobre cómo resolver un problema relacionado con una inactividad prolongada en las conexiones de transacciones, consulte La base de datos lleva mucho tiempo inactiva en la conexión de la transacción.

DevOps Guru puede hacer lo siguiente:

- Evitar muchos problemas comunes en las bases de datos cotejando la configuración de la base de datos con la configuración habitual recomendada.
- Alertar sobre problemas críticos en su flota que, si no se comprueban, pueden provocar problemas mayores en el futuro.
- Avisarle de los problemas que acaban de descubrirse.

Cada información proactiva contiene un análisis de la causa del problema y recomendaciones para las acciones correctivas.

Para obtener más información sobre Amazon DevOps Guru para Amazon RDS, consulte Análisis de anomalías de rendimiento de Aurora con Amazon DevOps Guru para Amazon RDS.

La base de datos lleva mucho tiempo inactiva en la conexión de la transacción

Una conexión a la base de datos lleva en el estado `idle in transaction` más de 1800 segundos.

Temas

- Versiones del motor admitidas
- Context
- Causas probables de este problema
- Acciones
- Métricas relevantes

Versiones del motor admitidas

Esta información es compatible con todas las versiones de Aurora PostgreSQL.

Context

Una transacción en el estado `idle in transaction` puede contener bloqueos que bloqueen otras consultas. También puede evitar que `VACUUM` (incluido autovacuum) limpie las filas inactivas, lo que provoca una sobrecarga de índices o tablas o un resumen de los ID de transacciones.

Causas probables de este problema

Una transacción iniciada en una sesión interactiva con BEGIN o START TRANSACTION no ha finalizado con los comandos COMMIT, ROLLBACK o END. Esto hace que la transacción pase al estado `idle in transaction`.

Acciones

Para encontrar transacciones inactivas, consulte `pg_stat_activity`.

En su cliente SQL, ejecute la siguiente consulta para ver todas las conexiones en el estado `idle in transaction` y ordenarlas por duración:

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

Recomendamos diferentes acciones en función de las causas.

Temas

- Finalización de la transacción
- Finalización de la conexión
- Configure el parámetro idle_in_transaction_session_timeout
- Compruebe el estado AUTOCOMMIT
- Compruebe la lógica de la transacción en el código de su aplicación

Finalización de la transacción

Al iniciar una transacción en una sesión interactiva con BEGIN o START TRANSACTION, esta pasa al estado `idle in transaction`. Permanecerá en este estado hasta que finalice la transacción al emitir los comandos COMMIT, ROLLBACK, END o hasta que finalice la conexión por completo para revertir la transacción.

Finalización de la conexión

Finalice la conexión con una transacción inactiva mediante la siguiente consulta:

```
SELECT pg_terminate_backend(pid);
```

pid es el ID del proceso de la conexión.

Configure el parámetro idle_in_transaction_session_timeout

Configure el parámetro `idle_in_transaction_session_timeout` en el grupo de parámetros. La ventaja de configurar este parámetro es que no requiere una intervención manual para finalizar el periodo de inactividad prolongado de la transacción. Para obtener más información sobre este parámetro, consulte la documentación de PostgreSQL.

El siguiente mensaje aparecerá en el archivo de registro de PostgreSQL una vez finalizada la conexión y cuando haya una transacción en el estado idle_in_transaction durante más tiempo del especificado.

```
FATAL: terminating connection due to idle in transaction timeout
```

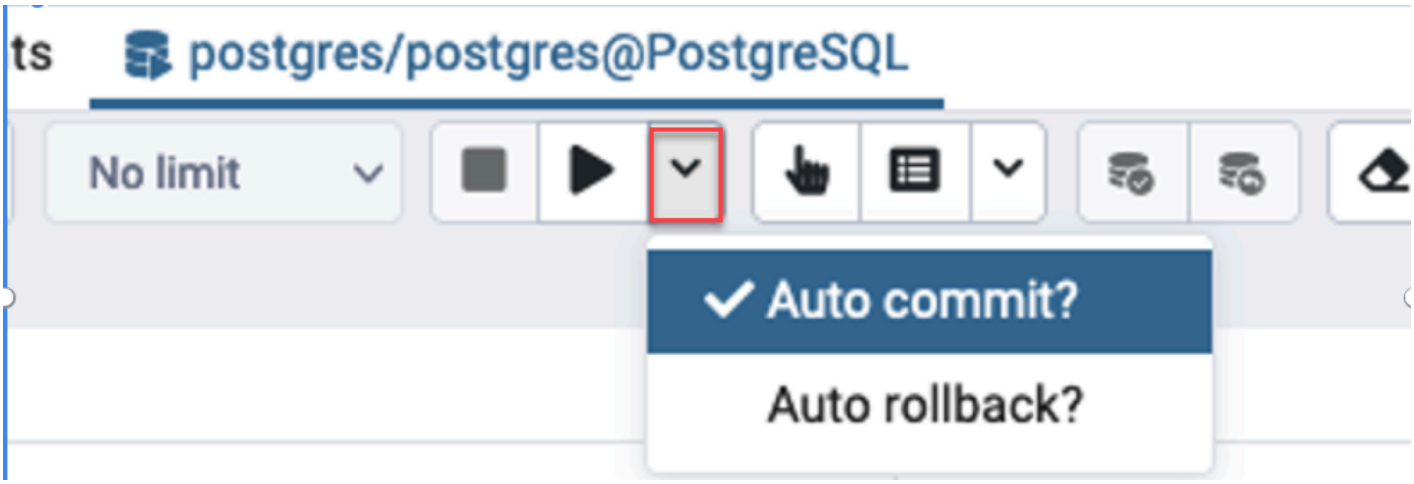
Compruebe el estado AUTOCOMMIT

AUTOCOMMIT está activado de forma predeterminada. Pero si se desactiva accidentalmente en el cliente, asegúrese de volver a activarlo.

- En su cliente psql, ejecute el siguiente comando:

```
postgres=> \set AUTOCOMMIT on
```

- En pgadmin, para activarlo, seleccione la opción AUTOCOMMIT en la flecha hacia abajo.



Compruebe la lógica de la transacción en el código de su aplicación

Investigue la lógica de su aplicación para detectar posibles problemas. Considere las siguientes acciones:

- Compruebe si la confirmación automática de JDBC está activada en su aplicación. Además, considere la posibilidad de usar comandos `COMMIT` explícitos en su código.
- Compruebe su lógica de gestión de errores para ver si cierra una transacción después de que se produzcan errores.
- Compruebe si su aplicación tarda mucho en procesar las filas devueltas por una consulta mientras la transacción está abierta. Si es así, considere la posibilidad de programar la aplicación para que cierre la transacción antes de procesar las filas.
- Compruebe si una transacción contiene muchas operaciones de larga duración. Si es así, divida una sola transacción en varias transacciones.

Métricas relevantes

Las siguientes métricas de PI están relacionadas con esta información:

- idle_in_transaction_count: número de sesiones en el estado `idle in transaction`.
- idle_in_transaction_max_time: la duración de la transacción en ejecución de más larga duración en el estado `idle in transaction`.

Prácticas recomendadas con Amazon Aurora PostgreSQL

A continuación, puede ver varias prácticas recomendadas para administrar el clúster de base de datos de Amazon Aurora PostgreSQL. Asegúrese de revisar también las tareas de mantenimiento básicas. Para obtener más información, consulte Rendimiento y escalado para Amazon Aurora PostgreSQL.

Temas

- Prevención del rendimiento lento, el reinicio automático y la conmutación por error de las instancias de base de datos Aurora PostgreSQL
- Diagnóstico de sobrecarga de tablas e índices
- Administración de memoria mejorada en Aurora PostgreSQL
- Conmutación por error rápida con Amazon Aurora PostgreSQL
- Recuperación rápida después de una conmutación por error con la administración de caché del clúster para Aurora PostgreSQL
- Administración de la pérdida de conexión de Aurora PostgreSQL con agrupación
- Gestión de conexiones inactivas en PostgreSQL
- Configuración de los parámetros de memoria para Aurora PostgreSQL
- Uso de las métricas de Amazon CloudWatch para analizar el uso de los recursos de Aurora PostgreSQL
- Uso de la replicación lógica para realizar una actualización de la versión principal para Aurora PostgreSQL
- Solución de problemas de almacenamiento en Aurora PostgreSQL

Prevención del rendimiento lento, el reinicio automático y la conmutación por error de las instancias de base de datos Aurora PostgreSQL

Si ejecuta una carga de trabajo pesada o cargas de trabajo que superan los recursos asignados a su instancia de base de datos, puede agotar los recursos en los que ejecuta la aplicación y la base de datos de Aurora. Para obtener métricas de su instancia de base de datos, como el uso de la CPU, el uso de la memoria y el número de conexiones de base de datos utilizadas, puede consultar las métricas proporcionadas por Amazon CloudWatch, Performance Insights y Enhanced Monitoring. Para obtener más información acerca de la monitorización de las métricas de las instancias de base de datos, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Si su carga de trabajo agota los recursos que utiliza, su instancia de base de datos podría ralentizarse, reiniciarse o incluso realizar una conmutación por error a otra instancia de base de datos. Para evitarlo, supervise la utilización de los recursos, examine la carga de trabajo que se ejecuta en la instancia de base de datos y realice las optimizaciones necesarias. Si las optimizaciones no mejoran las métricas de la instancia ni mitigan el agotamiento de los recursos, considere la posibilidad de ampliar la instancia de base de datos antes de alcanzar sus límites. Para obtener más información sobre las clases de instancias de base de datos disponibles y sus especificaciones, consulte Clases de instancia de base de datos de Amazon Aurora.

Diagnóstico de sobrecarga de tablas e índices

Puede utilizar el control de simultaneidad multiversión (MVCC) de PostgreSQL para ayudar a preservar la integridad de los datos. El MVCC de PostgreSQL funciona guardando una copia interna de las filas actualizadas o eliminadas (también denominadas tuplas) hasta que se confirme o anule una transacción. Esta copia interna guardada es invisible para los usuarios. Sin embargo, la tabla se puede sobrecargar si las utilidades VACUUM o AUTOVACUUM no limpian esas copias invisibles con regularidad. Si no se controla, la sobrecarga de las tablas puede generar mayores costes de almacenamiento y ralentizar la velocidad de procesamiento.

En muchos casos, la configuración predeterminada de VACUUM o AUTOVACUUM en Aurora es suficiente para gestionar la sobrecarga no deseada de las tablas. Sin embargo, es posible que desee comprobar si existe sobrecarga si su aplicación presenta las siguientes condiciones:

- Procesa una gran cantidad de transacciones en un tiempo relativamente corto entre los procesos de VACUUM.
- Funciona mal y se queda sin espacio de almacenamiento.

Para empezar, recopile la información más precisa sobre cuánto espacio ocupan las tuplas inactivas y cuánto espacio puede recuperar si elimina la sobrecarga de tablas e índices. Para ello, utilice la extensión `pgstattuple` para recopilar estadísticas de su clúster de Aurora. Para obtener más información, consulte pgstattuple. Los privilegios para usar la extensión `pgstattuple` están limitados al rol `pg_stat_scan_tables` y a los superusuarios de la base de datos.

Para crear la extensión `pgstattuple` en Aurora, conecte una sesión de cliente al clúster, por ejemplo, psql o pgAdmin, y utilice el siguiente comando:

```
CREATE EXTENSION pgstattuple;
```

Cree la extensión en cada base de datos que desee perfilar. Tras crear la extensión, utilice la interfaz de línea de comandos (CLI) para medir cuánto espacio inutilizable puede recuperar. Antes de recopilar estadísticas, modifique el grupo de parámetros del clúster configurando AUTOVACUUM en 0. Un ajuste de 0 impide que Aurora limpie automáticamente las tuplas inactivas que haya dejado la aplicación, lo que puede afectar a la precisión de los resultados. Introduzca el siguiente comando para crear una tabla sencilla:

```
postgres=> CREATE TABLE lab AS SELECT generate_series (0,100000);
SELECT 100001
```

En el siguiente ejemplo, ejecutamos la consulta con AUTOVACUUM activado para el clúster de base de datos. `dead_tuple_count` es 0, lo que indica que AUTOVACUUM ha eliminado los datos o tuplas obsoletos de la base de datos PostgreSQL.

Para usar `pgstattuple` para recopilar información sobre la tabla, especifique el nombre de la tabla o un identificador de objeto (OID) en la consulta:

```
postgres=> SELECT * FROM pgstattuple('lab');
```

```
table_len  | tuple_count | tuple_len | tuple_percent | dead_tuple_count |
 dead_tuple_len | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056    | 100001      | 2800028   | 77.16         | 0                | 0
  | 0                | 16616      | 0.46
(1 row)
```

En la siguiente consulta, desactivamos AUTOVACUUM e introducimos un comando que elimina 25 000 filas de la tabla. Como resultado, `dead_tuple_count` aumenta a 25 000.

```
postgres=> DELETE FROM lab WHERE generate_series < 25000;

DELETE 25000
```

```
SELECT * FROM pgstattuple('lab');
```

```
table_len | tuple_count | tuple_len | tuple_percent | dead_tuple_count | dead_tuple_len
 | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056 | 75001 | 2100028 | 57.87 | 25000 | 700000 | 19.29 | 16616 | 0.46
(1 row)
```

Para recuperar esas tuplas inactivas, inicie un proceso VACUUM.

Observación de la sobrecarga sin interrumpir la aplicación

La configuración de un clúster de Aurora está optimizada para proporcionar las prácticas recomendadas para la mayoría de las cargas de trabajo. Sin embargo, es posible que desee optimizar un clúster para que se adapte mejor a sus aplicaciones y patrones de uso. En este caso, puede utilizar la extensión `pgstattuple` sin interrumpir una aplicación ocupada. Para ello, realice estos pasos:

1. Clone su instancia de Aurora.
2. Modifique el archivo de parámetros para desactivar AUTOVACUUM en el clon.

3. Realice una consulta `pgstattuple` mientras prueba el clon con una carga de trabajo de ejemplo o con pgbench, que es un programa para ejecutar pruebas de referencia en PostgreSQL. Para obtener más información, consulte pgbench.

Tras ejecutar las aplicaciones y ver el resultado, utilice pg_repack o VACUUM FULL en la copia restaurada y compare las diferencias. Si observa una reducción significativa en dead_tuple_count, dead_tuple_len o dead_tuple_percent, ajuste la programación de vacuum en su clúster de producción para minimizar la sobrecarga.

Evitar la sobrecarga en las tablas temporales

Si la aplicación crea tablas temporales, asegúrese de que las elimina cuando ya no sean necesarias. Los procesos Autovacuum no localizan tablas temporales. Si no se seleccionan, las tablas temporales pueden sobrecargar rápidamente la base de datos. Además, la sobrecarga puede extenderse a las tablas del sistema, que son las tablas internas que realizan un seguimiento de los objetos y atributos de PostgreSQL, como pg_attribute y pg_depend.

Cuando ya no necesite una tabla temporal, puede utilizar la instrucción TRUNCATE para vaciar la tabla y liberar espacio. A continuación, vacíe manualmente las tablas pg_attribute y pg_depend. Al vaciar estas tablas, se garantiza que al crear y truncar/eliminar tablas temporales de forma continua no se añaden tuplas ni se contribuye a la sobrecarga del sistema.

Para evitar este problema al crear una tabla temporal, incluya la siguiente sintaxis, que elimina las filas nuevas cuando se confirma el contenido:

```
CREATE TEMP TABLE IF NOT EXISTS table_name(table_description) ON COMMIT DELETE ROWS;
```

La cláusula `ON COMMIT DELETE ROWS` trunca la tabla temporal cuando se confirma la transacción.

Evitar la sobrecarga en los índices

Al cambiar un campo indexado de una tabla, la actualización del índice da como resultado una o más tuplas inactivas en ese índice. El proceso autovacuum elimina la sobrecarga de los índices de forma predeterminada, pero esa limpieza consume una cantidad importante de tiempo y recursos. Para especificar las preferencias de limpieza del índice al crear una tabla, incluya la cláusula vacuum_index_cleanup. De forma predeterminada, en el momento de creación de la tabla, la cláusula se establece en AUTO, lo que significa que el servidor decide si es necesario limpiar el índice cuando vacía la tabla. Puede establecer la cláusula en ON para activar la limpieza de índices

de una tabla específica, o en OFF para desactivarla. Recuerde que, aunque es posible que ahorre tiempo si desactiva la limpieza de índices, podría dar lugar a un índice sobrecargado.

Puede controlar manualmente la limpieza de índices si utiliza VACUUM en una tabla en la línea de comandos. Para vaciar una tabla y eliminar las tuplas inactivas de los índices, incluya la cláusula INDEX_CLEANUP con el valor ON y el nombre de la tabla:

```
acctg=> VACUUM (INDEX_CLEANUP ON) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Para vaciar una tabla sin limpiar los índices, especifique el valor OFF:

```
acctg=> VACUUM (INDEX_CLEANUP OFF) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Administración de memoria mejorada en Aurora PostgreSQL

Aurora PostgreSQL ahora incluye funciones avanzadas de administración de memoria para optimizar el rendimiento y la resiliencia de las bases de datos en distintas cargas de trabajo. Estas mejoras ayudan a Aurora PostgreSQL a mantener una disponibilidad y una capacidad de respuesta uniformes, incluso durante los períodos de gran demanda de memoria.

Esta característica está disponible y se activa de forma predeterminada en las siguientes versiones de Aurora PostgreSQL para instancias aprovisionadas:

- 15.3 y todas las versiones secundarias posteriores
- 14.8 y versiones secundarias posteriores
- 13.11 y versiones secundarias posteriores
- 12.15 y versiones secundarias posteriores
- 11.20 y versiones secundarias posteriores

Esta característica está disponible y se habilita de forma predeterminada en las siguientes instancias de Aurora PostgreSQL para Aurora Serverless:

- 16.3 y todas las versiones secundarias posteriores
- 15.7 y todas las versiones secundarias posteriores
- 14.12 y versiones secundarias posteriores
- 13.5 y versiones secundarias posteriores

Cuando las cargas de trabajo de los clientes agotan toda la memoria libre disponible, el sistema operativo puede reiniciar la base de datos para proteger los recursos, lo que desencadena una falta de disponibilidad temporal. Las nuevas mejoras en la administración de memoria de Aurora PostgreSQL cancelan de forma proactiva ciertas transacciones cuando el sistema experimenta una gran presión de memoria, lo que ayuda a mantener la estabilidad de la base de datos.

Las principales características de la administración de memoria mejorada son las siguientes:

- Cancela las transacciones de bases de datos que solicitan más memoria cuando el sistema se acerca a una presión de memoria crucial.
- Se dice que el sistema está bajo una presión de memoria crucial, cuando agota toda la memoria física y está a punto de agotar el intercambio. En estas circunstancias, cualquier transacción que solicite memoria se cancelará para reducir inmediatamente la presión de memoria en la instancia de base de datos.
- Los indicadores de PostgreSQL y los procesos de trabajo secundarios esenciales como los procesos de autovacuum, siempre están protegidos.

Gestión de los parámetros de administración de memoria

Para activar la administración de memoria

Esta función está activada de forma predeterminada. Cuando se cancela una transacción por falta de memoria aparece un mensaje de error, como se muestra en el siguiente ejemplo:

```
ERROR: out of memory Detail: Failed on request of size 16777216.
```

Para desactivar la administración de memoria

Para desactivar esta característica, conéctese al clúster de base de datos de Aurora PostgreSQL con psql y utilice la instrucción SET para los valores de parámetros, como se indica a continuación.

> **Note**
>
> Recomendamos que mantenga la administración de la memoria habilitada. Esto ayuda a evitar posibles errores de memoria insuficiente que podrían provocar el reinicio de la base de datos inducido por la carga de trabajo debido al agotamiento de la memoria.

La siguiente tabla muestra cómo desactivar la característica de administración de memoria para las distintas versiones de Aurora PostgreSQL:

Versiones de Aurora PostgreSQL	Parámetro	Predeterminado/a	Comando para desactivar la administración de memoria en el nivel de sesión
11.20, 11.21, 12.15, 12.16, 13.11, 13.12, 14.8, 14.9, 15.3, 15.4	`rds.memory_allocation_guard`	`false`	`SET rds.memory_allocation_guard = true;`
12.17, 13.13, 14.10, 15.5 y versiones posteriores	`rds.enable_memory_management`	`true`	`SET rds.enable_memory_management = false;`

> **Note**
>
> El parámetro `rds.memory_allocation_guard` ha quedado obsoleto en las versiones 12.17, 13.13, 14.10, 15.5 y posteriores de Aurora PostgreSQL.

Al establecer los valores de estos parámetros en el grupo de parámetros de clúster de base de datos, se evita que se cancelen las consultas. Para obtener más información acerca de los grupos de parámetros de clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Limitación

- Esta característica no se admite en la clase de instancia db.t3.medium.

Conmutación por error rápida con Amazon Aurora PostgreSQL

A continuación puede ver información sobre cómo asegurarse de que la conmutación por error se produzca lo más rápido posible. Para recuperarse rápidamente tras una conmutación por error, puede utilizar la administración de caché del clúster para el clúster de bases de datos de Aurora PostgreSQL. Para obtener más información, consulte Recuperación rápida después de una conmutación por error con la administración de caché del clúster para Aurora PostgreSQL.

Algunos de los pasos que puede adoptar para que la conmutación por error pase rápidamente son los siguientes:

- Establezca keepalives del protocolo de control de transmisión (TCP, por sus siglas en inglés) con plazos cortos para detener las consultas que se ejecutan durante más tiempo antes de que venza el tiempo de espera de lectura en caso de que se produzca un error.
- Establezca tiempos de espera para que el almacenamiento en caché del sistema de nombres de dominio (DNS, por sus siglas en inglés) de Java sea agresivo. Esto ayuda a garantizar que el punto de conexión de solo lectura de Aurora pueda desplazarse correctamente por nodos de solo lectura en posteriores intentos de conexión.
- Establezca las variables de tiempo de inactividad, que se utilizan en la cadena de la conexión de JDBC, con los valores más bajos posibles. Utilice objetos de conexión independientes para consultas de ejecución corta y prolongada.
- Utilice los puntos de conexión de Aurora de lectura y escritura que se proporcionan para establecer una conexión al clúster.
- Utilice las operaciones de la API de RDS para probar la respuesta de la aplicación en caso de errores del lado del servidor. Además, puede usar una herramienta para supresión de paquetes para probar la respuesta de la aplicación ante errores del lado del cliente.
- Utilice el controlador JDBC de AWS para aprovechar al máximo las capacidades de conmutación por error de Aurora PostgreSQL. Para obtener más información sobre el controlador JDBC de AWS e instrucciones completas para utilizarlo, consulte el repositorio GitHub del controlador JDBC de Amazon Web Services (AWS).

A continuación, se analizan con más detalle.

Temas

- Configuración de parámetros Keepalive de TCP
- Configuración de su aplicación para una conmutación por error rápida
- Prueba de conmutación por error
- Ejemplo de conmutaciones por error rápidas en Java

Configuración de parámetros Keepalive de TCP

Si configura una conexión TCP, se asocia una serie de temporizadores a la conexión. Cuando el temporizador keepalive llega a cero, se envía un paquete de sondeo keepalive al punto de conexión. Si recibe una respuesta al sondeo, puede presuponer que la conexión sigue en funcionamiento.

Al habilitar parámetros keepalive de TCP y configurarlos agresivamente, se garantiza que si su cliente no puede conectarse a la base de datos, se cierre rápidamente cualquier conexión activa. A continuación, la aplicación se puede conectar a un nuevo punto de conexión.

Tiene que establecer los siguientes parámetros keepalive de TCP:

- `tcp_keepalives_idle` controla el tiempo, en segundos, después del cual se envía un paquete keepalive si el socket no ha enviado datos. Los ACK no se consideran datos. Recomendamos la siguiente configuración:

  ```
  tcp_keepalives_idle = 1
  ```
- `tcp_keepalives_interval` controla el tiempo, en segundos, entre el envío de posteriores paquetes keepalive después de enviar el paquete inicial. Establezca esta hora con el parámetro `tcp_keepalives_idle`. Recomendamos la siguiente configuración:

  ```
  tcp_keepalives_interval = 1
  ```
- `tcp_keepalives_count` es la cantidad de sondeos keepalive sin confirmar que tienen lugar antes de que se produzca la notificación a la aplicación. Recomendamos la siguiente configuración:

  ```
  tcp_keepalives_count = 5
  ```

Esta configuración debería realizar una notificación a la aplicación en un plazo de cinco segundos cuando la base de datos deja de responder. Puede establecer un valor de `tcp_keepalives_count` más elevado si los paquetes keepalive se suprimen con frecuencia

dentro de la red de la aplicación. Esto incrementa el tiempo que se tarda en detectar un error real, pero ofrece más capacidad de búfer en redes menos fiables.

Para configurar parámetros keepalive de TCP en Linux

1. Pruebe cómo configurar los parámetros keepalive de TCP.

   Recomendamos hacerlo con la línea de comandos y los siguientes comandos. Esta configuración sugerida es para todo el sistema. En otras palabras, también afecta a todas las demás aplicaciones que crean sockets con la opción `SO_KEEPALIVE` activada.

   ```
   sudo sysctl net.ipv4.tcp_keepalive_time=1
   sudo sysctl net.ipv4.tcp_keepalive_intvl=1
   sudo sysctl net.ipv4.tcp_keepalive_probes=5
   ```

2. Una vez que haya encontrado una configuración que funcione para su aplicación, esta configuración debe almacenarse de forma persistente agregando las siguientes líneas a `/etc/sysctl.conf`, incluido cualquier cambio que haya realizado:

   ```
   tcp_keepalive_time = 1
   tcp_keepalive_intvl = 1
   tcp_keepalive_probes = 5
   ```

Configuración de su aplicación para una conmutación por error rápida

A continuación, encontrará un análisis de varios cambios de configuración para Aurora PostgreSQL que puede realizar para lograr una conmutación por error rápida. Para obtener más información sobre la configuración y configuración del controlador de JDBC de PostgreSQL, consulte la documentación del controlador de JDBC de PostgreSQL.

Temas

- Reducción de los tiempos de espera de la caché de DNS
- Configuración de una cadena de conexión de Aurora PostgreSQL para conmutaciones por error rápidas
- Otras opciones para la obtención de la cadena de host

Reducción de los tiempos de espera de la caché de DNS

Cuando su aplicación trate de establecer una conexión después de una conmutación por error, el nuevo escritor Aurora PostgreSQL será un lector anterior. Puede encontrarlo mediante el punto de conexión de solo lectura de Aurora, antes de que las actualizaciones de DNS se hayan propagado por completo. Establecer el tiempo de vida (TTL, por sus siglas en inglés) de DNS de Java en un valor bajo (como debajo de 30 segundos) ayuda a desplazarse por los nodos del lector en intentos de conexión posteriores.

```
// Sets internal TTL to match the Aurora RO Endpoint TTL
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
// If the lookup fails, default to something like small to retry
java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
```

Configuración de una cadena de conexión de Aurora PostgreSQL para conmutaciones por error rápidas

Para utilizar la conmutación por error rápida de Aurora PostgreSQL, asegúrese de que la cadena de conexión de su aplicación tenga una lista de hosts en lugar de un solo host. Mostramos aquí una cadena de conexión de ejemplo que podría utilizar para conectarse a un clúster de Aurora PostgreSQL: En este ejemplo, los hosts aparecen en negrita.

```
jdbc:postgresql://myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
/postgres?user=<primaryuser>&password=<primarypw>&loginTimeout=2
&connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60
&tcpKeepAlive=true&targetServerType=primary
```

Para una mejor disponibilidad y evitar depender de la API de RDS, le recomendamos que mantenga un archivo con el que conectarse. Este archivo contiene una cadena de host desde la que su aplicación lee cuando establece una conexión con la base de datos. Esta cadena de host tiene todos los puntos de conexión de Aurora disponibles para el clúster. Para obtener más información acerca de los puntos de conexión de Aurora, consulte Conexiones de puntos de conexión de Amazon Aurora.

Por ejemplo, puede almacenar sus puntos de conexión en un archivo local como se muestra a continuación.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
```

```
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

Su aplicación lee de ese archivo para rellenar la sección host de la cadena de conexión de JDBC. Cambiar el nombre del clúster de base de datos provoca que estos puntos de conexión también cambien. Asegúrese de que la aplicación administre este evento, si se produce.

Otra opción consiste en usar una lista de nodos de instancia de base de datos, del siguiente modo.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node3.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node4.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

El beneficio de este enfoque es que el controlador de la conexión JDBC de PostgreSQL recorre todos los nodos de esta lista para encontrar una conexión válida. Por el contrario, cuando se usan los puntos de conexión de Aurora, solo se prueban dos nodos en cada intento de conexión. Sin embargo, el uso de nodos de instancias de base de datos tiene un inconveniente. Si agrega o elimina nodos de su clúster y la lista de puntos de conexión de la instancia se queda obsoleta, el controlador de la conexión podría no encontrar nunca el host correcto al que conectarse.

Configure los siguientes parámetros, de manera agresiva, para contribuir a garantizar que su aplicación no tenga que esperar demasiado para conectarse a cualquier host:

- `targetServerType`: controla si el controlador se conecta a un nodo de escritura o lectura. Para asegurarse de que las aplicaciones se vuelvan a conectar solo a un nodo de escritura, establezca el valor `targetServerType` en `primary`.

  Los valores posibles para el parámetro `targetServerType` incluyen `primary`, `secondary`, `any` y `preferSecondary`. El valor `preferSecondary` primero intenta establecer una conexión con un lector. Se conecta al escritor si no se puede establecer una conexión con el lector.
- `loginTimeout`: controla cuánto tiempo espera su aplicación para iniciar sesión en la base de datos después de que se haya establecido una conexión de socket.
- `connectTimeout`: controla cuánto tiempo espera el socket para establecer una conexión con la base de datos.

Puede modificar otros parámetros de la aplicación para acelerar el proceso de conexión, en función del grado de agresividad deseado en la aplicación.

- `cancelSignalTimeout`: en algunas aplicaciones, sería conveniente enviar la “mejor” señal de cancelación para una consulta cuyo tiempo se haya agotado. Si esta señal de cancelación se encuentra en su ruta de conmutación por error, debería plantearse configurarla agresivamente para evitar enviar esta señal a un host inoperativo.
- `socketTimeout`: este parámetro controla cuánto tiempo espera el socket a que se produzcan las operaciones de lectura. Es posible usar este parámetro como tiempo de espera de consulta "global" para garantizar que nunca se supere este valor. Una práctica recomendada es tener dos controladores de conexión. Un controlador de conexión ejecuta consultas de corta duración y establece un valor más bajo. Y otro controlador de conexión, para consultas de larga duración, tiene este valor más alto. De este modo, puede confiar en que los parámetros keepalive de TCP detengan las consultas de larga duración si el servidor deja de funcionar.
- `tcpKeepAlive`: active este parámetro para asegurarse de que se respeten los parámetros keepalive de TCP configurados.
- `loadBalanceHosts`: cuando se establece en `true`, este parámetro hace que la aplicación se conecte a un alojamiento aleatorio elegido entre una lista de alojamientos candidatos.

Otras opciones para la obtención de la cadena de host

Puede obtener la cadena de host de diferentes lugares, entre incluida la función `aurora_replica_status`, y mediante la API de Amazon RDS.

En muchos casos, debe determinar quién es el escritor del clúster o encontrar otros nodos de lector del clúster. Para ello, su aplicación puede conectarse a cualquier instancia de base de datos del clúster de base de datos y consultar la función `aurora_replica_status`. Puede utilizar esta función para reducir la cantidad de tiempo que se tarda en encontrar un host al que conectarse. Sin embargo, en ciertos escenarios de fallo de red, la función `aurora_replica_status` podría mostrar información obsoleta o incompleta.

Una buena manera de garantizar que la aplicación pueda encontrar un nodo al cual conectarse es intentar conectarse al punto de conexión del escritor del clúster y, a continuación, al punto de conexión del lector del clúster. Hágalo hasta que pueda establecer una conexión legible. Estos puntos de conexión no cambian a menos que cambie el nombre del clúster de base de datos. En general, pueden dejarse como miembros estáticos de su aplicación o almacenarse en un archivo de recursos desde el que lea la aplicación.

Después de establecer una conexión con uno de estos puntos de conexión, puede llamar a la función para obtener información sobre el resto del clúster. Para ello, llame a la función

`aurora_replica_status`. Por ejemplo, el siguiente comando obtiene información con `aurora_replica_status`.

```
postgres=> SELECT server_id, session_id, highest_lsn_rcvd, cur_replay_latency_in_usec,
 now(), last_update_timestamp
FROM aurora_replica_status();

server_id | session_id | highest_lsn_rcvd | cur_replay_latency_in_usec | now |
 last_update_timestamp
-----------+--------------------------------------+-----------------
+--------------------------+-------------------------------+-----------------------
mynode-1 | 3e3c5044-02e2-11e7-b70d-95172646d6ca | 594221001 | 201421 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-2 | 1efdd188-02e4-11e7-becd-f12d7c88a28a | 594221001 | 201350 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-3 | MASTER_SESSION_ID | | | 2017-03-07 19:50:24.695322+00 | 2017-03-07
 19:50:23+00
(3 rows)
```

Por ejemplo, la sección hosts de su cadena de conexión podría empezar con los puntos de conexión del clúster del escritor y del lector, tal como se muestra a continuación.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

Ante esta situación, su aplicación tratará de establecer una conexión a cualquier tipo de nodo, principal o secundario. Cuando la aplicación esté conectada, una buena práctica es examinar en primer lugar el estado de lectura-escritura del nodo. Para ello, consulte el resultado del comando `SHOW transaction_read_only`.

Si el valor de retorno de la consulta es `OFF`, significa que se ha conectado correctamente al nodo principal. Sin embargo, supongamos que el valor devuelto es `ON` y la aplicación requiere una conexión de lectura/escritura. En este caso, puede llamar a la función `aurora_replica_status` para determinar el `server_id` que tiene `session_id='MASTER_SESSION_ID'`. Esta función le proporciona el nombre del nodo principal. Puede usar esto con `endpointPostfix`, tal como se describe a continuación.

Asegúrese de estar al tanto cuando se conecte a una réplica con datos obsoletos. Cuando esto ocurre, la función `aurora_replica_status` podría mostrar información desactualizada. Puede establecer un umbral de obsolescencia en el nivel de aplicación. Para comprobarlo, puede ver

la diferencia entre la hora del servidor y el valor de `last_update_timestamp`. En general, su aplicación debe evitar pasar de un alojamiento a otro debido a la información conflictiva devuelta por la función `aurora_replica_status`. Su aplicación debe probar primero todos los hosts conocidos en lugar de seguir los datos devueltos por `aurora_replica_status`.

Enumeración de instancias mediante la operación de la API DescribeDBClusters (ejemplo en Java)

Mediante programación, puede consultar la lista de instancias con AWS SDK para Java, concretamente con la operación de la API DescribeDBClusters.

Se muestra aquí un breve ejemplo de cómo podría hacer esto en Java 8.

```
AmazonRDS client = AmazonRDSClientBuilder.defaultClient();
DescribeDBClustersRequest request = new DescribeDBClustersRequest()
   .withDBClusterIdentifier(clusterName);
DescribeDBClustersResult result =
rdsClient.describeDBClusters(request);

DBCluster singleClusterResult = result.getDBClusters().get(0);

String pgJDBCEndpointStr =
singleClusterResult.getDBClusterMembers().stream()
   .sorted(Comparator.comparing(DBClusterMember::getIsClusterWriter)
   .reversed()) // This puts the writer at the front of the list
   .map(m -> m.getDBInstanceIdentifier() + endpointPostfix + ":" +
 singleClusterResult.getPort()))
   .collect(Collectors.joining(","));
```

Aquí, `pgJDBCEndpointStr` contiene una lista con formato de puntos de conexión, tal como se muestra a continuación.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

La variable `endpointPostfix` puede ser una constante que establece su aplicación. O bien, su aplicación puede obtenerla consultando la operación de la API `DescribeDBInstances` para una sola instancia de su clúster. Este valor es constante dentro de un Región de AWS y para un cliente individual. Por lo tanto, guarda una llamada a la API para mantener esta constante en un archivo de recursos desde el que su aplicación lee. En el ejemplo anterior, se establece en lo siguiente.

```
.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com
```

Para fines de disponibilidad, una práctica recomendada sería utilizar, de manera predeterminada, los puntos de conexión de Aurora de su clúster de bases de datos si la API no respondiera o tardara demasiado en responder. Se garantiza que los puntos de enlace estén actualizados en el tiempo que se tarda en actualizar el registro de DNS. La actualización del registro DNS con un punto de conexión suele tardar menos de 30 segundos. Puede almacenar el punto de conexión en un archivo de recursos que consume la aplicación.

Prueba de conmutación por error

En todos los casos, debe tener un clúster de bases de datos con dos o más instancias de base de datos en él.

Desde el lado del servidor, algunas operaciones de la API pueden causar una interrupción del servicio que se puede usar para probar cómo responden sus aplicaciones:

- FailoverDBCluster: esta operación trata de promover a escritor una instancia de base de datos nueva en su clúster de bases de datos.

  En el siguiente ejemplo de código se muestra cómo se puede utilizar `failoverDBCluster` para provocar una interrupción. Para obtener más detalles sobre la configuración de un cliente de Amazon RDS, consulte Uso del SDK de AWS para Java.

  ```
  public void causeFailover() {

      final AmazonRDS rdsClient = AmazonRDSClientBuilder.defaultClient();

      FailoverDBClusterRequest request = new FailoverDBClusterRequest();
      request.setDBClusterIdentifier("cluster-identifier");

      rdsClient.failoverDBCluster(request);
  }
  ```

- RebootDBInstance: la conmutación por error no está garantizada en esta operación de la API. Sin embargo, cierra la base de datos del escritor. Puede usarla para probar cómo responde la aplicación ante caídas de las conexiones. El parámetro `ForceFailover` no se aplica a motores de Aurora. En su lugar, use la operación de la API `FailoverDBCluster`.
- ModifyDBCluster: la modificación del parámetro `Port` causa una interrupción cuando los nodos del clúster comienzan a escuchar en un puerto nuevo. En general, su aplicación puede responder primero a este error asegurándose de que solo su aplicación controle los cambios en los puertos. Además, asegúrese de que pueda actualizar adecuadamente los puntos de conexión de los

que depende. Para hacerlo, pida a alguien que actualice manualmente el puerto cuando realice modificaciones en el nivel de la API. O puede hacerlo mediante la API de RDS de la aplicación para determinar si el puerto ha cambiado.

- ModifyDBInstance: modificar el parámetro `DBInstanceClass` provoca una interrupción.
- DeleteDBInstance: eliminar el principal (escritor) provoca que una instancia de base de datos nueva se promueva al escritor en su clúster de bases de datos.

Desde el lado de la aplicación o del cliente, si está utilizando Linux, puede probar cómo responde la aplicación ante supresiones repentinas de paquetes. Puede hacerlo en función del puerto o el host o de si se envían o reciben paquetes TCP keepalive mediante el comando iptables.

Ejemplo de conmutaciones por error rápidas en Java

El siguiente ejemplo de código ilustra cómo una aplicación podría configurar un administrador del controlador de Aurora PostgreSQL.

La aplicación llamará a la función `getConnection` cuando necesite una conexión. Una llamada a `getConnection` puede no encontrar un host válido. Un ejemplo es cuando no se encuentra ningún escritor, pero el parámetro `targetServerType` está establecido en `primary`. En este caso, la aplicación de llamada simplemente debe volver a intentar llamar a la función.

Esto puede integrarse fácilmente en un concentrador de conexiones para evitar forzar el reintento hacia la aplicación. Con la mayoría de los concentradores de conexiones, puede especificar una cadena de conexión JDBC. Para que su solicitud pueda llamar a `getJdbcConnectionString` y pasarlo al concentrador de conexiones. Esto significa que puede usar una conmutación por error más rápida con Aurora PostgreSQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import org.joda.time.Duration;

public class FastFailoverDriverManager {
   private static Duration LOGIN_TIMEOUT = Duration.standardSeconds(2);
```

```
  private static Duration CONNECT_TIMEOUT = Duration.standardSeconds(2);
  private static Duration CANCEL_SIGNAL_TIMEOUT = Duration.standardSeconds(1);
  private static Duration DEFAULT_SOCKET_TIMEOUT = Duration.standardSeconds(5);

  public FastFailoverDriverManager() {
      try {
           Class.forName("org.postgresql.Driver");
      } catch (ClassNotFoundException e) {
           e.printStackTrace();
      }

      /*
        * RO endpoint has a TTL of 1s, we should honor that here. Setting this
aggressively makes sure that when
        * the PG JDBC driver creates a new connection, it will resolve a new different
RO endpoint on subsequent attempts
        * (assuming there is > 1 read node in your cluster)
        */
       java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
      // If the lookup fails, default to something like small to retry
      java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
  }

  public Connection getConnection(String targetServerType) throws SQLException {
      return getConnection(targetServerType, DEFAULT_SOCKET_TIMEOUT);
  }

  public Connection getConnection(String targetServerType, Duration queryTimeout)
throws SQLException {
       Connection conn =
DriverManager.getConnection(getJdbcConnectionString(targetServerType, queryTimeout));

      /*
        * A good practice is to set socket and statement timeout to be the same thing
since both
        * the client AND server will stop the query at the same time, leaving no
running queries
        * on the backend
        */
       Statement st = conn.createStatement();
       st.execute("set statement_timeout to " + queryTimeout.getMillis());
       st.close();

      return conn;
```

```
  }

  private static String urlFormat = "jdbc:postgresql://%s"
          + "/postgres"
          + "?user=%s"
          + "&password=%s"
          + "&loginTimeout=%d"
          + "&connectTimeout=%d"
          + "&cancelSignalTimeout=%d"
          + "&socketTimeout=%d"
          + "&targetServerType=%s"
          + "&tcpKeepAlive=true"
          + "&ssl=true"
          + "&loadBalanceHosts=true";
  public String getJdbcConnectionString(String targetServerType, Duration
 queryTimeout) {
      return String.format(urlFormat,
               getFormattedEndpointList(getLocalEndpointList()),
               CredentialManager.getUsername(),
               CredentialManager.getPassword(),
               LOGIN_TIMEOUT.getStandardSeconds(),
               CONNECT_TIMEOUT.getStandardSeconds(),
               CANCEL_SIGNAL_TIMEOUT.getStandardSeconds(),
               queryTimeout.getStandardSeconds(),
               targetServerType
      );
  }

  private List<String> getLocalEndpointList() {
      /*
        * As mentioned in the best practices doc, a good idea is to read a local
 resource file and parse the cluster endpoints.
        * For illustration purposes, the endpoint list is hardcoded here
        */
       List<String> newEndpointList = new ArrayList<>();
       newEndpointList.add("myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");
       newEndpointList.add("myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");

      return newEndpointList;
  }

  private static String getFormattedEndpointList(List<String> endpoints) {
```

```
        return IntStream.range(0, endpoints.size())
                .mapToObj(i -> endpoints.get(i).toString())
                .collect(Collectors.joining(","));
    }
}
```

Recuperación rápida después de una conmutación por error con la administración de caché del clúster para Aurora PostgreSQL

Para la recuperación rápida de la instancia de base de datos del escritor en sus clústeres de Aurora PostgreSQL si se produce una conmutación por error, utilice la administración de caché del clúster para Amazon Aurora PostgreSQL. La administración de caché del clúster garantiza que el rendimiento de la aplicación se mantiene si se produce una conmutación por error.

En una situación de conmutación por error típica, puede que observe una degradación del rendimiento temporal, pero acusada, después de la conmutación por error. Esta degradación se produce porque cuando se inicia la instancia de base de datos de conmutación por error, la caché del búfer está vacía. Una caché vacía se conoce también como caché fría. Una caché fría degrada el rendimiento porque la instancia de base de datos tiene que realizar la lectura a partir del disco inferior en lugar de aprovechar los valores almacenados en la caché del búfer.

Con la administración de la caché del clúster, establecerá una instancia de base de datos del lector específica como destino de conmutación por error. La administración de la caché del clúster garantiza que los datos en la caché del lector designada se mantiene sincronizada con los datos en la caché de la instancia de la base de datos del escritor. La caché del lector designado con los valores precompletados se conoce como caché templada. Si se produce una conmutación por error, el lector designado utiliza valores en su caché templada de inmediato cuando se promociona en la nueva instancia de base de datos del escritor. Este enfoque proporciona a su aplicación un rendimiento de recuperación mucho mejor.

La administración de caché de clústeres requiere que la instancia de lector designada tenga el mismo tipo y tamaño de clase de instancia (`db.r5.2xlarge` o `db.r5.xlarge`, por ejemplo) que el escritor. Tenga esto en cuenta cuando cree sus clústeres de base de datos Aurora PostgreSQL para que el clúster pueda recuperarse durante una conmutación por error. Para obtener una lista de los tipos y tamaños de clase de instancia, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

> **Note**
>
> La administración de caché de clúster no es compatible con el clúster de base de datos secundario de Aurora PostgreSQL que forma parte de bases de datos globales de Aurora. Para los clústeres principales en los que se admite esta característica, se recomienda no ejecutar ninguna carga de trabajo en el lector de nivel 0 designado.

Contenido

- Configuración de la administración de la caché del clúster
  - Habilitación de la administración de la caché del clúster
  - Establecimiento de la prioridad de la capa de promoción para la instancia de base de datos del escritor
  - Establecimiento de la prioridad de la capa de promoción para una instancia de base de datos del lector
- Monitoreo de la caché del búfer
- Solución de problemas de configuración de CCM

Configuración de la administración de la caché del clúster

Para configurar la administración de caché de clúster, realice los siguientes procesos en orden.

Temas

- Habilitación de la administración de la caché del clúster
- Establecimiento de la prioridad de la capa de promoción para la instancia de base de datos del escritor
- Establecimiento de la prioridad de la capa de promoción para una instancia de base de datos del lector

> **Note**
>
> Deje que transcurra al menos un minuto después de completar estos pasos para que la gestión de la caché del clúster esté totalmente operativa.

Habilitación de la administración de la caché del clúster

Para habilitar la administración de caché de clúster, siga los pasos que se describen a continuación.

Consola

Para habilitar la administración de la caché del clúster, realice el siguiente procedimiento:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, elija el grupo de parámetros para el clúster de base de datos Aurora PostgreSQL.

   El clúster de base de datos debe utilizar un grupo de parámetros distinto al predeterminado, ya que no puede cambiar los valores en un grupo de parámetros predeterminado.
4. En Parameter group actions (Acciones de grupos de parámetros), seleccione Edit (Editar).
5. Establezca el valor del parámetro del clúster de `apg_ccm_enabled` en 1.
6. Seleccione Save changes (Guardar cambios).

AWS CLI

Para habilitar la administración de la caché del clúster para el clúster de base de datos de Aurora PostgreSQL, utilice el comando de la AWS CLI `modify-db-cluster-parameter-group` con los siguientes parámetros necesarios:

- `--db-cluster-parameter-group-name`
- `--parameters`

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my-db-cluster-parameter-group \
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
```

```
    --db-cluster-parameter-group-name my-db-cluster-parameter-group ^
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Establecimiento de la prioridad de la capa de promoción para la instancia de base de datos del escritor

Para la administración de caché de clúster, asegúrese de que la prioridad de promoción sea tier-0 para la instancia de base de datos del escritor del clúster de base de datos de Aurora PostgreSQL. La prioridad de la capa de promoción es un valor que especifica el orden en el que se promociona el lector de Aurora en la instancia de base de datos del escritor después de un error. Los valores válidos con de 0 a 15, donde 0 es la primera prioridad y 15 la última. Para obtener más información sobre la capa de promoción, consulte Tolerancia a errores para un clúster de base de datos de Aurora.

Consola

Para establecer la prioridad de la promoción para la instancia de base de datos del escritor en tier-0, realice el siguiente procedimiento

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija la instancia de base de datos del escritor del clúster de base de datos de Aurora PostgreSQL.
4. Elija Modify (Modificar). Aparece la página Modify DB Instance.
5. En el panel Configuración adicional elija el nivel 0 para Failover priority (Prioridad de conmutación por error).
6. Elija Continue (Continuar) y consulte el resumen de las modificaciones.
7. Para aplicar los cambios inmediatamente después de guardarlos, elija Apply immediately (Aplicar inmediatamente).
8. Seleccione Modify DB Instance (Modificar instancia de base de datos) para guardar los cambios.

AWS CLI

Para configurar la prioridad de la capa de promoción en 0 para la instancia de base de datos del escritor mediante la AWS CLI, invoque el comando modify-db-instance con los siguientes parámetros necesarios:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier writer-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier writer-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Establecimiento de la prioridad de la capa de promoción para una instancia de base de datos del lector

Debe configurar solamente una instancia de base de datos del lector para la administración de la caché del clúster. Para ello, elija un lector del clúster de Aurora PostgreSQL que sea del mismo tipo y tamaño de instancia que la instancia de base de datos del escritor. Por ejemplo, si el escritor utiliza `db.r5.xlarge`, elija un lector que utilice el mismo tipo y tamaño de clase de instancia. A continuación, establezca su prioridad de la capa de promoción en 0.

La prioridad de la capa de promoción es un valor que especifica el orden en el que se promociona el lector de Aurora en la instancia de base de datos del escritor después de un error. Los valores válidos con de 0 a 15, donde 0 es la primera prioridad y 15 la última.

Consola

Para establecer la prioridad de la promoción para la instancia de base de datos del escritor en tier-0, realice el siguiente procedimiento:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija una instancia de base de datos del lector del clúster de base de datos de Aurora PostgreSQL que sea la misma clase de instancia que la instancia de base de datos del escritor.
4. Elija Modify (Modificar). Aparece la página Modify DB Instance.
5. En el panel Configuración adicional elija el nivel 0 para Failover priority (Prioridad de conmutación por error).
6. Elija Continue (Continuar) y consulte el resumen de las modificaciones.
7. Para aplicar los cambios inmediatamente después de guardarlos, elija Apply immediately (Aplicar inmediatamente).
8. Seleccione Modify DB Instance (Modificar instancia de base de datos) para guardar los cambios.

AWS CLI

Para configurar la prioridad de la capa de promoción en 0 para la instancia de base de datos del lector mediante la AWS CLI, invoque el comando modify-db-instance con los siguientes parámetros necesarios:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier reader-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier reader-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Monitoreo de la caché del búfer

Después de configurar la gestión de la caché del clúster, podrá monitorizar el estado de la sincronización entre la caché del búfer de la instancia de base de datos del escritor y la caché del búfer en caliente del lector designado. Para examinar el contenido de la caché del búfer en la instancia de base de datos del escritor y la instancia de base de datos del lector designada, utilice el módulo `pg_buffercache` de PostgreSQL. Para obtener más información, consulte la documentación de PostgreSQL sobre `pg_buffercache`.

Uso de la función **`aurora_ccm_status`**

La administración de la caché del clúster también proporciona la función `aurora_ccm_status`. Utilice la función `aurora_ccm_status` en la instancia de base de datos del escritor para obtener la siguiente información sobre el progreso del calentamiento de la caché en el lector designado:

- `buffers_sent_last_minute`: la cantidad de búferes enviados al lector designado en el último minuto.
- `buffers_found_last_minute`: el número de búferes de acceso frecuente identificados durante el último minuto.
- `buffers_sent_last_scan`: la cantidad de búferes enviados al lector designado durante el último análisis completo de la caché del búfer.
- `buffers_found_last_scan`: la cantidad de búferes identificados como de acceso frecuente y que tienen que enviarse durante el último análisis completo de la caché del búfer. Los búferes ya almacenados en la caché en el lector designado no se envían.
- `buffers_sent_current_scan`: la cantidad de búferes enviados hasta el momento durante el análisis actual.
- `buffers_found_current_scan`: la cantidad de búferes identificados como de acceso frecuente en el análisis actual.
- `current_scan_progress`: la cantidad de búferes visitados hasta el momento durante el análisis actual.

En el siguiente ejemplo se muestra cómo utilizar la función `aurora_ccm_status` para convertir algunos de sus resultados en una tasa templada y un porcentaje templado.

```
SELECT buffers_sent_last_minute*8/60 AS warm_rate_kbps,
   100*(1.0-buffers_sent_last_scan::float/buffers_found_last_scan) AS warm_percent
   FROM aurora_ccm_status();
```

Solución de problemas de configuración de CCM

Al habilitar el parámetro de clúster `apg_ccm_enabled`, la administración de la caché del clúster se activa automáticamente en el nivel de instancia en la instancia de base de datos de escritura y en una instancia de base de datos de lectura en el clúster de base de datos de Aurora PostgreSQL. La instancia de escritura y la de lectura deben usar el mismo tipo y tamaño de clase de instancia. La prioridad de su nivel de promoción está establecida en `0`. Las demás instancias de lectura del clúster de base de datos deben tener un nivel de promoción distinto de cero y la administración de la caché del clúster está deshabilitada para esas instancias.

Los siguientes motivos pueden provocar problemas en la configuración e inhabilitar la administración de la caché del clúster:

- Cuando no hay una instancia de base de datos de lectura única configurada para el nivel de promoción 0.
- Cuando la instancia de base de datos del escritor no está configurada en el nivel de promoción 0.
- Cuando hay más de un lector, las instancias de base de datos configuradas en el nivel de promoción 0.
- Cuando las instancias de base de datos del escritor y de un lector con el nivel de promoción 0 no tienen el mismo tamaño de instancia.

Administración de la pérdida de conexión de Aurora PostgreSQL con agrupación

Cuando las aplicaciones cliente se conectan y desconectan con tanta frecuencia que el tiempo de respuesta del clúster de base de datos de Aurora PostgreSQL se ralentiza, se dice que el clúster está experimentando pérdida de conexión. Cada nueva conexión al punto de conexión del clúster de base de datos de Aurora PostgreSQL consume recursos, lo que reduce el número de recursos que se puede usar para procesar la carga de trabajo real. La pérdida de conexión es un problema que le

aconsejamos que administre siguiendo algunas de las prácticas recomendadas que se describen a continuación.

Para empezar, puede mejorar los tiempos de respuesta en los clústeres de base de datos de Aurora PostgreSQL que tienen altas tasas de pérdida de conexión. Por ejemplo, puede utilizar un concentrador de conexiones como RDS Proxy. Un concentrador de conexiones proporciona una caché de conexiones listas para usar para los clientes. Casi todas las versiones de Aurora PostgreSQL admiten RDS Proxy. Para obtener más información, consulte Proxy de Amazon RDS con Aurora PostgreSQL.

Si su versión específica de Aurora PostgreSQL no admite RDS Proxy, puede usar otro agrupador de conexiones compatible con PostgreSQL, como PgBouncer. Para obtener más información, consulte el sitio web de PgBouncer.

Para comprobar si el clúster de base de datos de Aurora PostgreSQL puede beneficiarse de la agrupación de conexiones, consulte el archivo `postgresql.log` para conexiones y desconexiones. También puede usar Performance Insights para averiguar cuánta pérdida de conexión está experimentando su clúster de base de datos de Aurora PostgreSQL. A continuación, encontrará información sobre ambos temas.

Conexiones y desconexiones de registro

Los parámetros `log_connections` y `log_disconnections` pueden capturar conexiones y desconexiones en la instancia de escritor instancia del clúster de base de datos de Aurora PostgreSQL. De forma predeterminada, este parámetro está desactivado. Para activar estos parámetros, utilice un grupo de parámetros personalizado y actívelo cambiando el valor a 1. Para obtener más información acerca de los grupos de parámetros personalizados, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora. Para comprobar la configuración, conéctese al punto de conexión del clúster de base de datos para Aurora PostgreSQL mediante psql y realice la consulta de la siguiente manera.

```
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_connections';
 setting
---------
on
(1 row)
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_disconnections';
setting
```

```
---------
on
(1 row)
```

Con ambos parámetros activados, el registro captura todas las conexiones y desconexiones nuevas. Verá el usuario y la base de datos de cada nueva conexión autorizada. En el momento de la desconexión, también se registra la duración de la sesión, tal como se muestra en el ejemplo siguiente.

```
2022-03-07 21:44:53.978 UTC [16641] LOG: connection authorized: user=labtek
 database=labdb application_name=psql
2022-03-07 21:44:55.718 UTC [16641] LOG: disconnection: session time: 0:00:01.740
 user=labtek database=labdb host=[local]
```

Para comprobar la pérdida de conexión de su aplicación, active estos parámetros si aún no están activados. A continuación, recopile datos en el registro de PostgreSQL para analizarlos ejecutando su aplicación con una carga de trabajo y un período de tiempo realistas. Puede usar la consola de RDS para ver los archivos de registro. Elija la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL y, a continuación, elija la pestaña Logs & events (Registros y eventos). Para obtener más información, consulte Visualización y descripción de archivos de registro de base de datos.

También puede descargar el archivo de registro de la consola y usar la siguiente secuencia de comandos. Esta secuencia encuentra el número total de conexiones autorizadas y descartadas por minuto.

```
grep "connection authorized\|disconnection: session time:"
 postgresql.log.2022-03-21-16|\
awk {'print $1,$2}' |\
sort |\
uniq -c |\
sort -n -k1
```

En el resultado del ejemplo, puede ver un pico en las conexiones autorizadas seguido de desconexiones a partir de las 16:12:10.

```
.....
,.......
.........
```

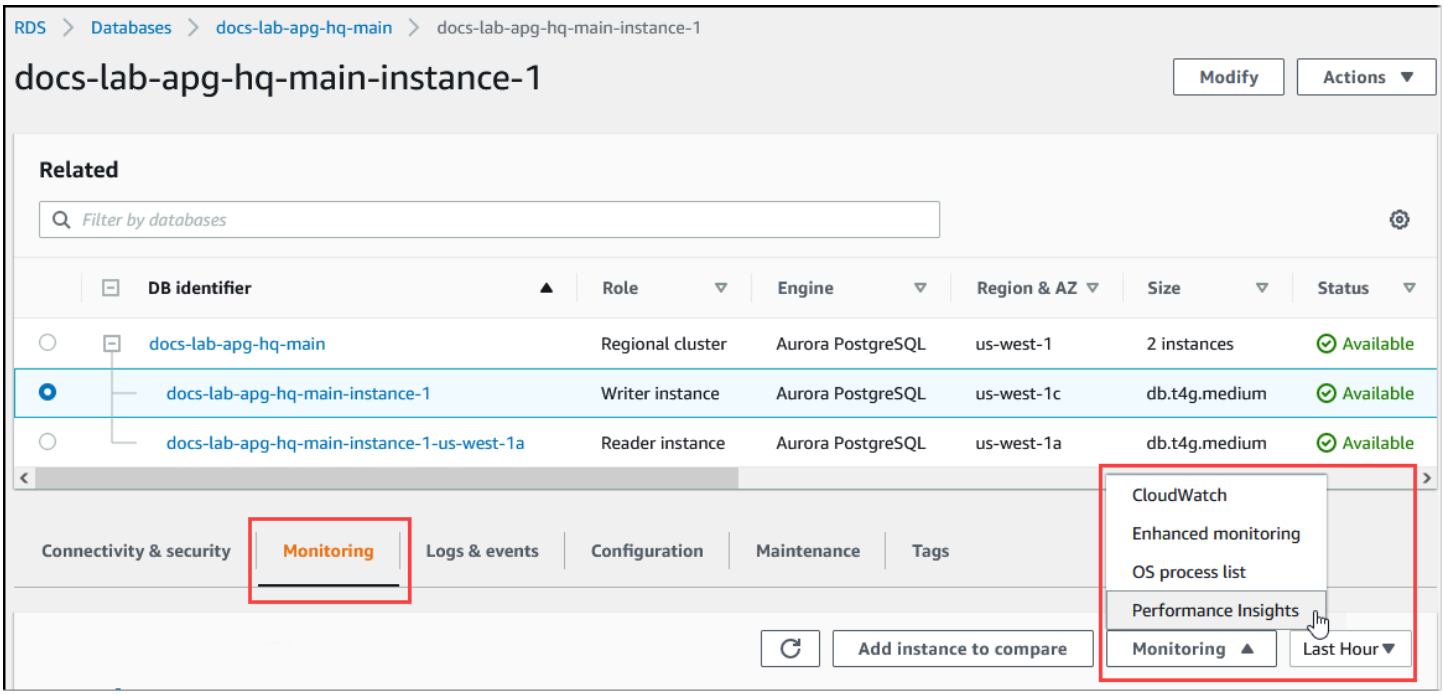
```
5 2022-03-21 16:11:55 connection authorized:
9 2022-03-21 16:11:55 disconnection: session
5 2022-03-21 16:11:56 connection authorized:
5 2022-03-21 16:11:57 connection authorized:
5 2022-03-21 16:11:57 disconnection: session
32 2022-03-21 16:12:10 connection authorized:
30 2022-03-21 16:12:10 disconnection: session
31 2022-03-21 16:12:11 connection authorized:
27 2022-03-21 16:12:11 disconnection: session
27 2022-03-21 16:12:12 connection authorized:
27 2022-03-21 16:12:12 disconnection: session
41 2022-03-21 16:12:13 connection authorized:
47 2022-03-21 16:12:13 disconnection: session
46 2022-03-21 16:12:14 connection authorized:
41 2022-03-21 16:12:14 disconnection: session
24 2022-03-21 16:12:15 connection authorized:
29 2022-03-21 16:12:15 disconnection: session
28 2022-03-21 16:12:16 connection authorized:
24 2022-03-21 16:12:16 disconnection: session
40 2022-03-21 16:12:17 connection authorized:
42 2022-03-21 16:12:17 disconnection: session
40 2022-03-21 16:12:18 connection authorized:
40 2022-03-21 16:12:18 disconnection: session
.....
,.....
.........
1 2022-03-21 16:14:10 connection authorized:
1 2022-03-21 16:14:10 disconnection: session
1 2022-03-21 16:15:00 connection authorized:
1 2022-03-21 16:16:00 connection authorized:
```

Con esta información, puede decidir si su carga de trabajo puede beneficiarse de un agrupador de conexiones. Para realizar análisis más detallados, puede utilizar Performance Insights.

Detección de la pérdida de conexión con Performance Insights

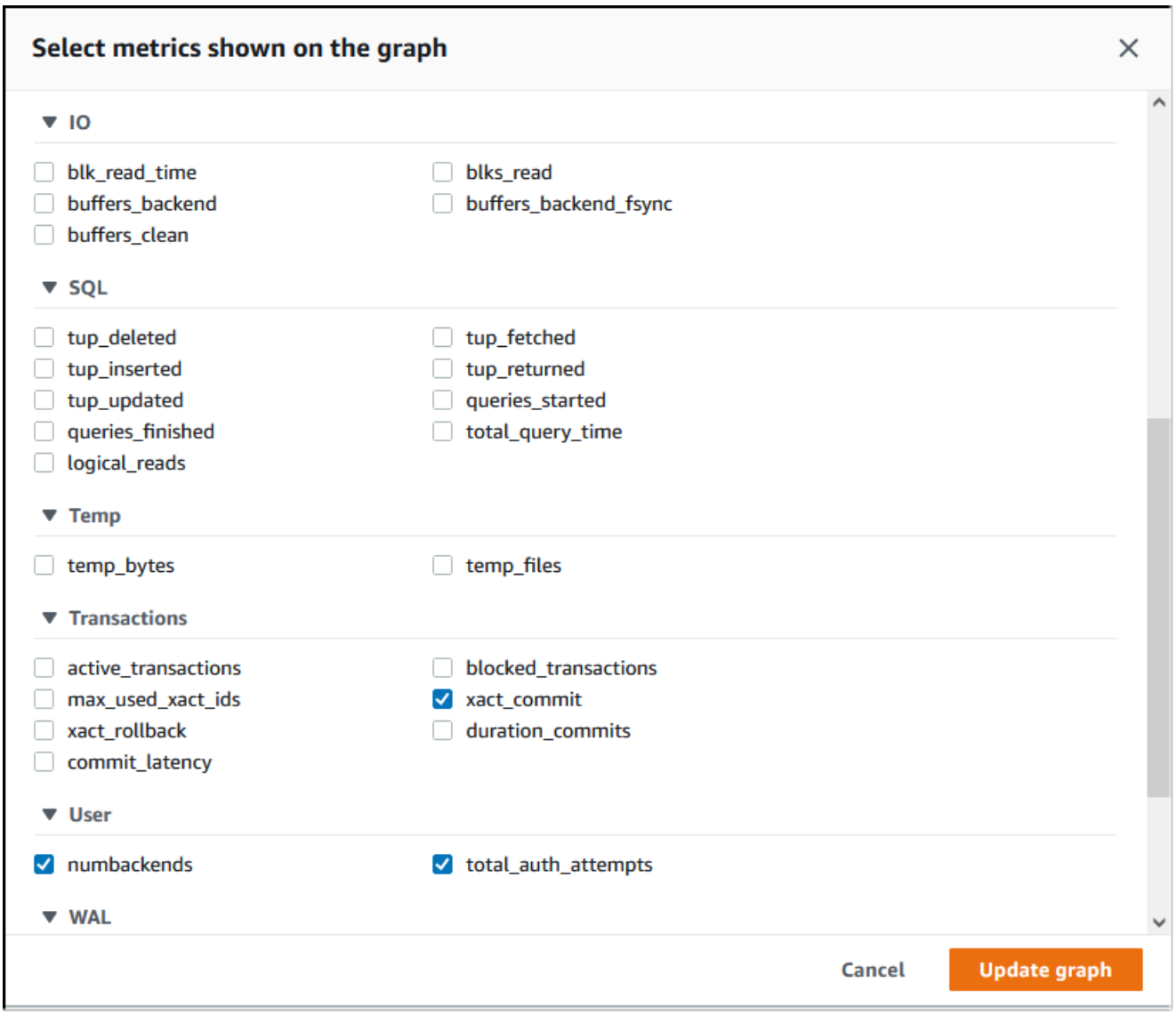
Puede utilizar Performance Insights para evaluar la cantidad de pérdida de conexión en su clúster de base de datos de Aurora PostgreSQL-Compatible Edition. Al crear un clúster de base de datos de Aurora PostgreSQL, la configuración de Performance Insights se activa de forma predeterminada. Si ha desactivado esta opción al crear el clúster de base de datos, modifique el clúster para activar la función. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Si Performance Insights se ejecuta en su clúster de base de datos de Aurora PostgreSQL, puede elegir las métricas que desee supervisar. Puede acceder a Performance Insights desde el panel de navegación de la consola. También puede acceder a Performance Insights desde la pestaña Monitoring (Monitorización)de la instancia de escritura del clúster de base de datos de Aurora PostgreSQL, tal como se muestra en la siguiente imagen.



En la consola de Performance Insights, elija Manage metrics (Administrar métricas). Para analizar la actividad de conexión y desconexión del clúster de base de datos de Aurora PostgreSQL, elija las siguientes métricas. Todas estas son métricas de PostgreSQL.

- `xact_commit`: número de transacciones confirmadas.
- `total_auth_attempts`: número de intentos de conexión de usuarios autenticados por minuto.
- `numbackends`: número de backends conectados actualmente a la base de datos.

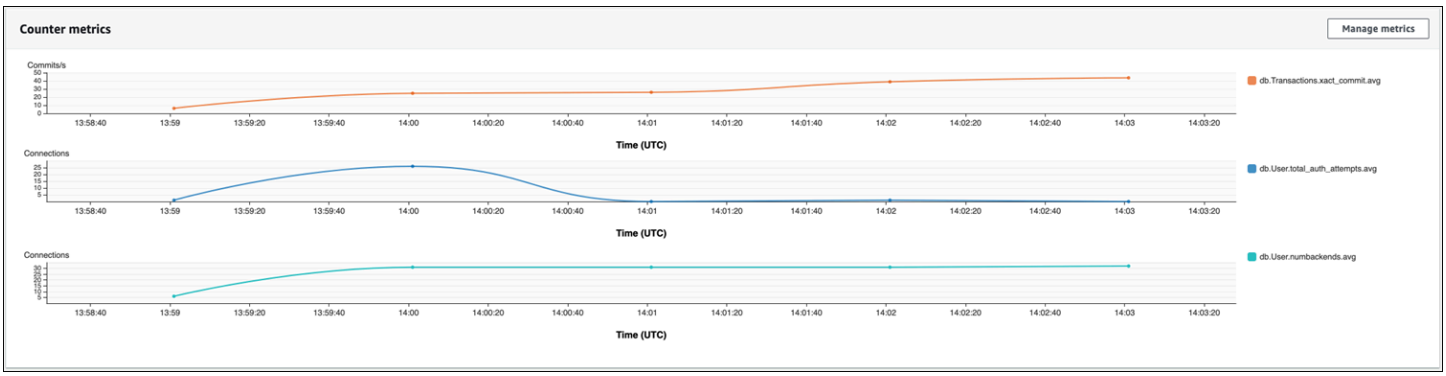


Para guardar la configuración y mostrar la actividad de conexión, elija Update graph (Actualizar gráfico).

En la siguiente imagen, puede ver el impacto de ejecutar pgbench con 100 usuarios. La línea que muestra las conexiones está en una pendiente ascendente constante. Para más información sobre pgbench y su uso, consulte pgbench en la documentación de PostgreSQL.

La imagen muestra que ejecutar una carga de trabajo con tan solo 100 usuarios sin un agrupador de conexiones puede provocar un aumento significativo en la cantidad de `total_auth_attempts` durante todo el procesamiento de la carga de trabajo. Tenga en cuenta que es mejor mantener `total_auth_attempts` lo más cerca posible de cero.

Con la agrupación de conexiones de RDS Proxy, los intentos de conexión aumentan al inicio de la carga de trabajo. Después de configurar el grupo de conexiones, el promedio disminuye. Los recursos utilizados por las transacciones y el uso de backend se mantienen consistentes durante todo el procesamiento de la carga.



Para obtener más información acerca del uso de Performance Insights con su clúster de base de datos de Aurora PostgreSQL, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora. Para analizar las métricas, consulte Análisis de métricas mediante el panel de Información sobre rendimiento.

Demostración de los beneficios de la agrupación de conexiones

Tal como se menciona anteriormente, si determina que el clúster de base de datos de Aurora PostgreSQL tiene un problema de pérdida de conexión, puede usar RDS Proxy para mejorar el rendimiento. A continuación, encontrará un ejemplo que muestra las diferencias en el procesamiento

de una carga de trabajo cuando las conexiones están agrupadas y cuando no lo están. El ejemplo usa pgbench para modelar una carga de trabajo de transacciones.

De igual modo que psql, pgbench es una aplicación cliente de PostgreSQL que puede instalar y ejecutar desde su máquina cliente local. También puede instalarla y ejecutarla desde la instancia de Amazon EC2 que utiliza para administrar el clúster de base de datos de Aurora PostgreSQL. Para obtener más información, consulte pgbench en la documentación de PostgreSQL.

Para seguir este ejemplo, primero debe crear el entorno pgbench en su base de datos. El siguiente comando es la plantilla básica para inicializar las tablas pgbench en la base de datos especificada. En este ejemplo se utiliza la cuenta de usuario principal predeterminada, `postgres`, para iniciar sesión. Cámbiela según sea necesario para el clúster de base de datos de Aurora PostgreSQL. El entorno pgbench se crea en una base de datos en la instancia de escritor de su clúster.

> **Note**
>
> El proceso de inicialización de pgbench descarta y vuelve a crear tablas denominadas `pgbench_accounts`, `pgbench_branches`, `pgbench_history` y `pgbench_tellers`. Asegúrese de que la base de datos que elija para *`dbname`* cuando inicialice pgbench no use estos nombres.

```
pgbench -U postgres -h db-cluster-instance-1.111122223333.aws-region.rds.amazonaws.com
 -p 5432 -d -i -s 50 dbname
```

Para pgbench, especifique los siguientes parámetros.

-d

Genera un informe de depuración a medida que se ejecuta pgbench.

-h

Especifica el punto de conexión de la instancia de escritor del clúster de la base de datos de Aurora PostgreSQL.

-i

Inicializa el entorno pgbench en la base de datos para las pruebas de referencia.

-p

Identifica el puerto utilizado para las conexiones de la base de datos El valor predeterminado para Aurora PostgreSQL suele ser 5432 o 5433.

-s

Especifica el factor de escala que se utilizará para rellenar las tablas con filas. El factor de escala predeterminado es 1, lo que genera 1 fila en la tabla `pgbench_branches`, 10 filas en la tabla `pgbench_tellers` y 100 000 filas en la tabla `pgbench_accounts`.

-U

Especifica la cuenta de usuario de la instancia de escritor del clúster de la base de datos de Aurora PostgreSQL.

Después de configurar el entorno pgbench, puede ejecutar pruebas de referencia con y sin agrupación de conexiones. La prueba predeterminada consiste en una serie de cinco comandos SELECT, UPDATE e INSERT por transacción que se ejecutan repetidamente durante el tiempo especificado. Puede especificar el factor de escala, el número de clientes y otros detalles para modelar sus propios casos de uso.

Por ejemplo, el siguiente comando ejecuta la referencia durante 60 segundos (opción -T, para el tiempo) con 20 conexiones simultáneas (la opción -c). La opción -C hace que la prueba se ejecute con una nueva conexión cada vez, en lugar de hacerlo una vez por sesión de cliente. Esta configuración le da una indicación de la sobrecarga de la conexión.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 -C labdb
Password:**********
pgbench (14.3, server 13.3)
  starting vacuum...end.
  transaction type: <builtin: TPC-B (sort of)>
    scaling factor: 50
    query mode: simple
    number of clients: 20
    number of threads: 1
    duration: 60 s
    number of transactions actually processed: 495
    latency average = 2430.798 ms
    average connection time = 120.330 ms
```

```
    tps = 8.227750 (including reconnection times)
```

La ejecución de pgbench en la instancia de escritor de un clúster de base de datos de Aurora PostgreSQL sin volver a utilizar las conexiones muestra que solo se procesan unas 8 transacciones por segundo. Esto da un total de 495 transacciones durante la prueba de 1 minuto.

Si reutiliza las conexiones, la respuesta del clúster de base de datos de Aurora PostgreSQL para el número de usuarios es casi 20 veces más rápida. Con la reutilización, se procesan un total de 9042 transacciones, si comparamos con las 495 con la misma cantidad de tiempo y para el mismo número de conexiones de usuario. La diferencia es que, a continuación, cada conexión se vuelve a utilizar.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 labdb
Password:*********
pgbench (14.3, server 13.3)
      starting vacuum...end.
      transaction type: <builtin: TPC-B (sort of)>
      scaling factor: 50
      query mode: simple
      number of clients: 20
      number of threads: 1
      duration: 60 s
      number of transactions actually processed: 9042
      latency average = 127.880 ms
      initial connection time = 2311.188 ms
      tps = 156.396765 (without initial connection time)
```

En este ejemplo se muestra que agrupar conexiones puede mejorar significativamente los tiempos de respuesta. Para obtener información acerca de cómo configurar RDS Proxy para el clúster de base de datos de Aurora PostgreSQL, consulte Amazon RDS Proxy para Aurora.

Gestión de conexiones inactivas en PostgreSQL

Las conexiones inactivas se producen cuando una sesión de base de datos permanece activa en el servidor a pesar de que la aplicación cliente se haya abandonado o terminado de forma anormal. Esta situación suele producirse cuando los procesos del cliente se bloquean o terminan de forma inesperada sin cerrar correctamente las conexiones de la base de datos ni cancelar las solicitudes en curso.

PostgreSQL identifica y limpia de manera eficiente las conexiones inactivas cuando los procesos del servidor están inactivos o intentan enviar datos a los clientes. Sin embargo, la detección

es difícil en el caso de las sesiones que están inactivas, esperan la intervención del cliente o en las que se están ejecutando consultas de forma activa. Para gestionar estos escenarios, PostgreSQL proporciona los parámetros `tcp_keepalives_*`, `tcp_user_timeout`, y `client_connection_check_interval`.

Temas

- Descripción de keepalive de TCP
- Parámetros clave keepalive de TCP en Aurora PostgreSQL
- Casos de uso de la configuración de keepalive de TCP
- Prácticas recomendadas

Descripción de keepalive de TCP

Keepalive de TCP es un mecanismo en el nivel de protocolo que ayuda a mantener y verificar la integridad de la conexión. Cada conexión TCP mantiene configuraciones en el nivel del kernel que rigen el comportamiento de keepalive. Cuando el temporizador de keepalive expira, el sistema realiza lo siguiente:

- Envía un paquete de sondeo sin datos y con el indicador ACK activado.
- Espera una respuesta del punto de conexión remoto de acuerdo con las especificaciones de TCP/IP.
- Administra el estado de la conexión en función de la respuesta o la falta de respuesta.

Parámetros clave keepalive de TCP en Aurora PostgreSQL

Parámetro	Descripción	Valores predeterminados
`tcp_keepalives_idle`	Specifies number of seconds of inactivity before sending keepalive message.	300
`tcp_keepalives_interval`	Specifies number of seconds between retransmissions of unacknowledged keepalive messages.	30

Parámetro	Descripción	Valores predeterminados
`tcp_keepalives_count`	Maximum lost keepalive messages before declaring connection dead	2
`tcp_user_timeout`	Specifies how long (in Milliseconds) unacknowledged data can remain before forcibly closing the connection.	0
`client_connection_check_interval`	Sets the interval (in Milliseconds) for checking client connection status during long-running queries. This ensures quicker detection of closed connections.	0

Casos de uso de la configuración de keepalive de TCP

Mantenimiento de las sesiones inactivas como activas

Para evitar que los firewalls o enrutadores terminen las conexiones inactivas debido a la inactividad:

- Configure `tcp_keepalives_idle` para enviar paquetes de keepalive a intervalos regulares.

Detección de conexiones inactivas

Para detectar rápidamente las conexiones inactivas:

- Ajuste `tcp_keepalives_idle`, `tcp_keepalives_interval` y `tcp_keepalives_count`. Por ejemplo, con los valores predeterminados de Aurora PostgreSQL, se tarda aproximadamente un minuto (2 sondas × 30 segundos) en detectar una conexión inactiva. Reducir estos valores puede acelerar la detección.
- Utilice `tcp_user_timeout` para especificar el tiempo máximo de espera para una confirmación.

La configuración de keepalive de TCP ayuda al kernel a detectar conexiones inactivas, pero PostgreSQL puede no actuar hasta que se utilice el socket. Si una sesión está ejecutando una consulta larga, es posible que las conexiones inactivas solo se detecten una vez completada la consulta. En PostgreSQL 14 y versiones posteriores, `client_connection_check_interval` puede acelerar la detección de conexiones inactivas consultando periódicamente el socket durante la ejecución de la consulta.

Prácticas recomendadas

- Establezca intervalos de keepalive razonables: ajuste `tcp_user_timeout`, `tcp_keepalives_idle`, `tcp_keepalives_count` y `tcp_keepalives_interval` para equilibrar la velocidad de detección y el uso de recursos.
- Optimice para su entorno: alinee la configuración con el comportamiento de la red, las políticas de firewall y las necesidades de la sesión.
- Aproveche las características de PostgreSQL: utilice `client_connection_check_interval` en PostgreSQL 14 y versiones posteriores para comprobar la conexión de forma eficaz.

Configuración de los parámetros de memoria para Aurora PostgreSQL

En Amazon Aurora PostgreSQL, puede usar varios parámetros que controlan la cantidad de memoria utilizada para las distintas tareas de procesamiento. Si una tarea ocupa más memoria que la cantidad establecida para un parámetro determinado, Aurora PostgreSQL utiliza otros recursos para el procesamiento, como escribir en el disco. Esto puede provocar que el clúster de base de datos de Aurora PostgreSQL se ralentice o se detenga, con un error de memoria insuficiente.

La configuración predeterminada de cada parámetro de memoria normalmente puede gestionar las tareas de procesamiento previstas. Sin embargo, también puede ajustar los parámetros relacionados con la memoria del clúster de base de datos de Aurora PostgreSQL . Realice este ajuste para asegurarse de que se asigne suficiente memoria para procesar su carga de trabajo específica.

A continuación, encontrará información sobre los parámetros que controlan la gestión de memoria. También puede aprender a evaluar la utilización de la memoria.

Comprobación y configuración de los valores de los parámetros

Los parámetros que puede configurar para administrar la memoria y evaluar el uso de memoria del clúster de base de datos de Aurora PostgreSQL son los siguientes:

- `work_mem`: especifica la cantidad de memoria que el clúster de base de datos de Aurora PostgreSQL utiliza para las operaciones internas de ordenación y las tablas hash antes de escribir en los archivos temporales del disco.
- `log_temp_files`: registra la creación de archivos temporales, los nombres y los tamaños de los archivos. Cuando se activa este parámetro, se almacena una entrada de registro para cada archivo temporal que se crea. Actívelo para ver con qué frecuencia el clúster de base de datos de Aurora PostgreSQL necesita escribir en el disco. Desactívelo de nuevo después de recopilar información sobre la generación de archivos temporales del clúster de base de datos de Aurora PostgreSQL, para evitar un registro excesivo.
- `logical_decoding_work_mem`: especifica la cantidad de memoria (en kilobytes) que debe usar cada búfer de reordenamiento interno antes del volcado en el disco. Esta memoria se usa para la descodificación lógica, que es el proceso que se utiliza para crear una réplica. Se lleva a cabo convirtiendo los datos del archivo de registro de escritura anticipada (WAL) a la salida de transmisión lógica que necesita el destino.

  El valor de este parámetro crea un búfer único del tamaño especificado para cada conexión de replicación. De forma predeterminada, es de 65 536 KB. Después de llenar este búfer, el exceso se escribe en el disco como un archivo. Para minimizar la actividad del disco, puede establecer el valor de este parámetro a un valor mucho más alto que el de `work_mem`.

Todos estos son parámetros dinámicos, por lo que puede cambiarlos para la sesión actual. Para ello, conéctese a la instrucción del clúster de base de datos de Aurora PostgreSQL con psql y usando la instrucción `SET`, tal como se muestra a continuación.

```
SET parameter_name TO parameter_value;
```

La configuración de la sesión dura tanto como la sesión. Cuando finaliza la sesión, el parámetro vuelve a su configuración en el grupo de parámetros del clúster de base de datos. Antes de cambiar cualquier parámetro, compruebe primero los valores actuales consultando la tabla `pg_settings`, de la siguiente manera.

```
SELECT unit, setting, max_val
   FROM pg_settings WHERE name='parameter_name';
```

Por ejemplo, para encontrar el valor del parámetro `work_mem`, conéctese a la instancia de escritor del clúster de base de datos de Aurora PostgreSQL y ejecute la siguiente consulta.

```
SELECT unit, setting, max_val, pg_size_pretty(max_val::numeric)
  FROM pg_settings WHERE name='work_mem';
unit  | setting  | max_val   | pg_size_pretty
------+----------+-----------+----------------
 kB   | 1024     | 2147483647| 2048 MB
(1 row)
```

Para cambiar la configuración de los parámetros para que se mantengan, es necesario utilizar un grupo de parámetros del clúster de base de datos personalizado. Después de practicar con el clúster de base de datos de Aurora PostgreSQL con valores diferentes para estos parámetros usando el parámetro SET, puede crear un grupo de parámetros personalizado y aplicarlo a su clúster de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.

Comprensión del parámetro de memoria de trabajo

El parámetro de memoria de trabajo (work_mem) especifica la cantidad máxima de memoria que Aurora PostgreSQL puede usar para procesar consultas complejas. Las consultas complejas incluyen aquellas que implican operaciones de clasificación o agrupación; en otras palabras, las consultas que utilizan las siguientes cláusulas:

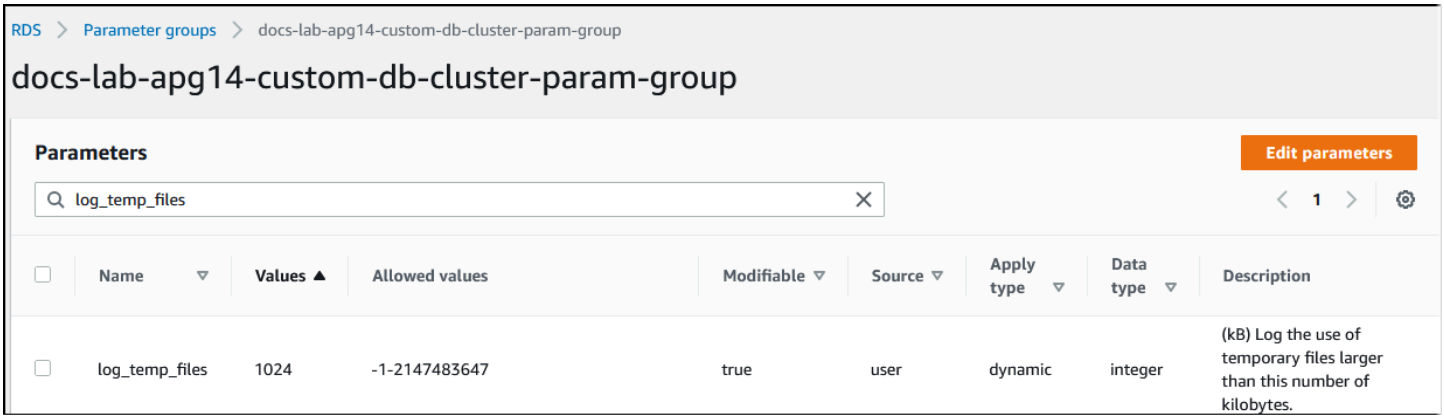
- ORDER BY
- DISTINCT
- GROUP BY
- JOIN (MERGE y HASH)

El planificador de consultas afecta indirectamente a la forma en que el clúster de base de datos de Aurora PostgreSQL utiliza la memoria El planificador de consultas genera planes de ejecución para procesar instrucciones SQL. Un plan determinado puede dividir una consulta compleja en varias unidades de trabajo que se pueden ejecutar en paralelo. Cuando es posible, Aurora PostgreSQL utiliza la cantidad de memoria especificada en el parámetro work_mem para cada sesión antes de escribir en el disco para cada proceso paralelo.

Varios usuarios de bases de datos que ejecutan varias operaciones simultáneamente y generan varias unidades de trabajo en paralelo pueden agotar la memoria de trabajo asignada al clúster de base de datos de Aurora PostgreSQL. Esto puede provocar una creación excesiva de archivos temporales y E/S de disco o, lo que es peor, puede provocar un error de falta de memoria.

Identificación del uso de archivos temporales

Siempre que la memoria necesaria para procesar consultas supere el valor especificado en el parámetro `work_mem`, los datos de trabajo se descargan al disco en un archivo temporal. Puede ver la frecuencia con la que ocurre esto activando el parámetro `log_temp_files`. De forma predeterminada, este parámetro está desactivado (establecido en -1). Para capturar toda la información del archivo temporal, defina este parámetro en 0. Establezca `log_temp_files` en cualquier otro entero positivo para capturar información de archivos temporales para archivos iguales o superiores a esa cantidad de datos (en kilobytes). En la siguiente imagen, puede ver un ejemplo de Consola de administración de AWS.



Tras configurar el registro de archivos temporales, puede realizar pruebas con su propia carga de trabajo para comprobar si la configuración de la memoria de trabajo es suficiente. También puede simular una carga de trabajo mediante pgbench, una sencilla aplicación de evaluación de referencia de la comunidad de PostgreSQL.

En el siguiente ejemplo se inicializa (`-i`) `pgbench` creando las tablas y filas necesarias para ejecutar las pruebas. En este ejemplo, el factor de escalado (`-s` 50) crea 50 filas en la tabla `pgbench_branches`, 500 filas en la tabla `pgbench_tellers` y 5 000 000 filas en la tabla `pgbench_accounts` de la base de datos de `labdb`.

```
pgbench -U postgres -h your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -i -s 50 labdb
Password:
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
```

```
generating data (client-side)...
5000000 of 5000000 tuples (100%) done (elapsed 15.46 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 61.13 s (drop tables 0.08 s, create tables 0.39 s, client-side generate 54.85
 s, vacuum 2.30 s, primary keys 3.51 s)
```

Después de inicializar el entorno, puede ejecutar la referencia para un tiempo específico (`-T`) y el número de clientes (`-c`). En este ejemplo también se utiliza la opción `-d` para generar información de depuración a medida que el clúster de base de datos de Aurora PostgreSQL procesa las transacciones.

```
pgbench -h -U postgres your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -d -T 60 -c 10 labdb
Password:*******
pgbench (14.3)
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 1
duration: 60 s
number of transactions actually processed: 1408
latency average = 398.467 ms
initial connection time = 4280.846 ms
tps = 25.096201 (without initial connection time)
```

Para obtener más información acerca de pgbench, consulte pgbench en la documentación de PostgreSQL.

Puede utilizar el comando de metacomandos psql (\d) para enumerar las relaciones, como tablas, vistas e índices, creadas por pgbench.

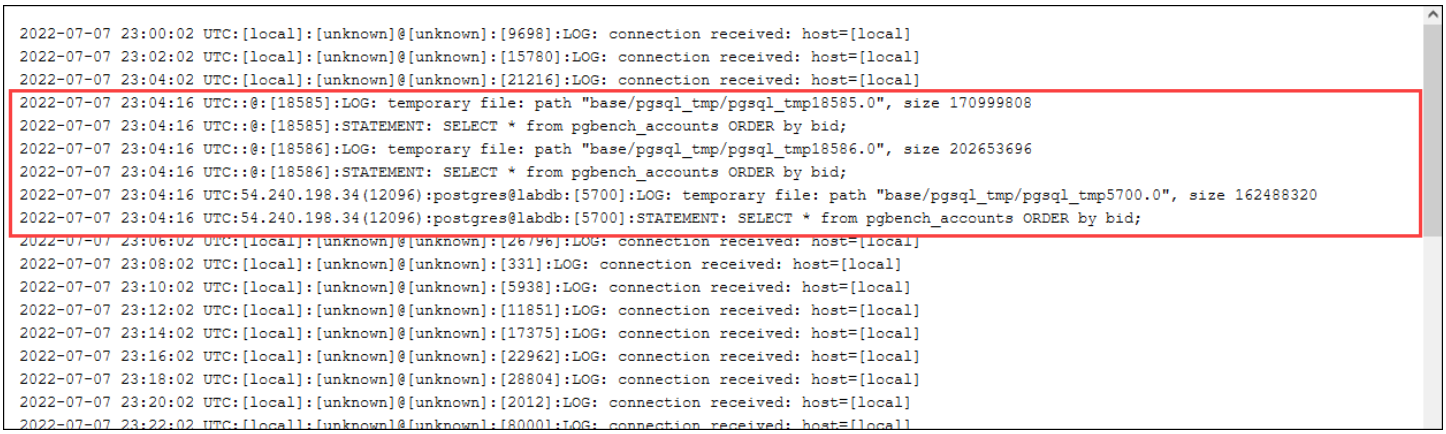
```
labdb=>  \d pgbench_accounts
 Table "public.pgbench_accounts"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 aid      | integer       |           | not null |
 bid      | integer       |           |          |
 abalance | integer       |           |          |
 filler   | character(84) |           |          |
```

```
Indexes:
    "pgbench_accounts_pkey" PRIMARY KEY, btree (aid)
```

Como se muestra en el resultado, la tabla `pgbench_accounts` está indexada en la columna `aid`. Para asegurarse de que la siguiente consulta utilice memoria de trabajo, consulte cualquier columna no indexada, como la que se muestra en el siguiente ejemplo.

```
postgres=> SELECT * FROM pgbench_accounts ORDER BY bid;
```

Compruebe los archivos temporales en el registro. Para ello, abra Consola de administración de AWS, elija la instancia del clúster de base de datos de Aurora PostgreSQL y, a continuación, la pestaña Logs & events (Registros y eventos). Puede ver los registros en la consola o descargarlos para analizarlos en detalle. Tal como se muestra en la siguiente imagen, el tamaño de los archivos temporales necesarios para procesar la consulta indica que debe considerar aumentar la cantidad especificada para el parámetro `work_mem`.



Puede configurar este parámetro de forma diferente para individuos y grupos, en función de sus necesidades operativas. Por ejemplo, puede establecer el parámetro `work_mem` en 8 GB para el rol denominado `dev_team`.

```
postgres=> ALTER ROLE dev_team SET work_mem='8GB';
```

Con esta configuración de `work_mem`, a cualquier rol que sea miembro del rol `dev_team` se le asignan hasta 8 GB de memoria de trabajo.

Uso de índices para un tiempo de respuesta más rápido

Si sus consultas tardan demasiado en devolver resultados, puede comprobar que los índices se están utilizando de la forma esperada. En primer lugar, active `\timing`, el metacomando psql, tal como se indica a continuación.

```
postgres=>  \timing on
```

Después de activar la temporización, utilice una instrucción SELECT sencilla.

```
postgres=> SELECT COUNT(*) FROM
  (SELECT * FROM pgbench_accounts
  ORDER BY bid)
  AS accounts;
count
-------
5000000
(1 row)
Time: 3119.049 ms (00:03.119)
```

Como se ve en el resultado, esta consulta ha tardado poco más de 3 segundos en completarse. Para mejorar el tiempo de respuesta, cree un índice en `pgbench_accounts`, de la siguiente manera.

```
postgres=> CREATE INDEX ON pgbench_accounts(bid);
CREATE INDEX
```

Vuelva a ejecutar la consulta y observe que el tiempo de respuesta es más rápido. En este ejemplo, la consulta se completó unas 5 veces más rápido, en aproximadamente medio segundo.

```
postgres=>  SELECT COUNT(*) FROM (SELECT * FROM pgbench_accounts ORDER BY bid) AS
 accounts;
 count
-------
 5000000
(1 row)
Time: 567.095 ms
```

Ajuste de la memoria de trabajo para la descodificación lógica

La replicación lógica ha estado disponible en todas las versiones de Aurora PostgreSQL desde su introducción en PostgreSQL versión 10. Al configurar la replicación lógica, también puede establecer

el parámetro `logical_decoding_work_mem` para especificar la cantidad de memoria que el proceso de descodificación lógica puede usar para el proceso de descodificación y transmisión.

Durante la descodificación lógica, los registros de registro de escritura anticipada (WAL) se convierten en instrucciones SQL que luego se envían a otro destino para la replicación lógica u otra tarea. Cuando se escribe una transacción en el WAL y, a continuación, se convierte, la transacción completa debe ajustarse al valor especificado para `logical_decoding_work_mem`. De forma predeterminada, este parámetro se establece en 65536 KB. Los desbordamientos se escriben en el disco. Esto significa que se debe volver a leer desde el disco antes de que se pueda enviar a su destino, lo que ralentiza el proceso general.

Puede evaluar la cantidad de desbordamiento de transacciones en su carga de trabajo actual en un momento específico mediante la función `aurora_stat_file`, tal como se muestra en el siguiente ejemplo.

```
SELECT split_part (filename, '/', 2)
   AS slot_name, count(1) AS num_spill_files,
   sum(used_bytes) AS slot_total_bytes,
   pg_size_pretty(sum(used_bytes)) AS slot_total_size
   FROM aurora_stat_file()
   WHERE filename like '%spill%'
   GROUP BY 1;
  slot_name | num_spill_files | slot_total_bytes | slot_total_size
------------+-----------------+------------------+-----------------
 slot_name  |       590       |      411600000   | 393 MB
(1 row)
```

Esta consulta devuelve el recuento y el tamaño de los archivos de vertido en el clúster de base de datos de Aurora PostgreSQL cuando se invoca la consulta. Es posible que las cargas de trabajo más largas no tengan aún ningún archivo de vertido en el disco. Para crear perfiles de cargas de trabajo de larga duración, le recomendamos que cree una tabla para capturar la información del archivo de vertido a medida que se ejecute la carga de trabajo. Puede crear la tabla tal y como se indica a continuación.

```
CREATE TABLE spill_file_tracking AS
    SELECT now() AS spill_time,*
    FROM aurora_stat_file()
    WHERE filename LIKE '%spill%';
```

Para ver cómo se usan los archivos de vertido durante la replicación lógica, configure un publicador y un suscriptor y, a continuación, inicie una replicación simple. Para obtener más información, consulte Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL. Con la replicación en marcha, puede crear un trabajo que capture el conjunto de resultados de la función de archivo de vertido `aurora_stat_file()`, de la siguiente manera.

```
INSERT INTO spill_file_tracking
  SELECT now(),*
  FROM aurora_stat_file()
  WHERE filename LIKE '%spill%';
```

Utilice el siguiente comando psql para ejecutar el trabajo una vez por segundo.

```
\watch 0.5
```

Mientras se ejecuta el trabajo, conéctese a la instancia de escritor desde otra sesión de psql. Utilice la siguiente serie de instrucciones para ejecutar una carga de trabajo que supere la configuración de memoria y haga que Aurora PostgreSQL cree un archivo de vertido.

```
labdb=> CREATE TABLE my_table (a int PRIMARY KEY, b int);
CREATE TABLE
labdb=> INSERT INTO my_table SELECT x,x FROM generate_series(0,10000000) x;
INSERT 0 10000001
labdb=> UPDATE my_table SET b=b+1;
UPDATE 10000001
```

Estas instrucciones pueden tardar varios minutos en completarse. Cuando haya terminado, pulse la tecla Ctrl y la tecla C a la vez para detener la función de monitorización. A continuación, utilice el siguiente comando para crear una tabla que contenga la información sobre el uso de archivos de vertido del clúster de base de datos de Aurora PostgreSQL.

```
SELECT spill_time, split_part (filename, '/', 2)
    AS slot_name, count(1)
    AS spills, sum(used_bytes)
    AS slot_total_bytes, pg_size_pretty(sum(used_bytes))
    AS slot_total_size FROM spill_file_tracking
  GROUP BY 1,2 ORDER BY 1;
                  spill_time | slot_name             | spills | slot_total_bytes |
 slot_total_size
```

```
------------------------------+----------------------+--------+-----------------
+-----------------
2022-04-15 13:42:52.528272+00 | replication_slot_name | 1      | 142352280        | 136
 MB
2022-04-15 14:11:33.962216+00 | replication_slot_name | 4      | 467637996        | 446
 MB
2022-04-15 14:12:00.997636+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:03.030245+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:05.059761+00 | replication_slot_name | 5      | 618410996        | 590
 MB
2022-04-15 14:12:07.22905+00  | replication_slot_name | 5      | 640585316        | 611
 MB
(6 rows)
```

El resultado muestra que al ejecutar el ejemplo se crearon cinco archivos de vertido que utilizaron 611 MB de memoria. Para evitar escribir en el disco, recomendamos configurar el parámetro `logical_decoding_work_mem` con el siguiente tamaño de memoria más alto: 1024.

Uso de las métricas de Amazon CloudWatch para analizar el uso de los recursos de Aurora PostgreSQL

Aurora envía automáticamente datos de métricas a CloudWatch en periodos de 1 minuto. Puede analizar el uso de los recursos de Aurora PostgreSQL mediante métricas de CloudWatch. Puede evaluar el rendimiento de la red y el uso de la red con las métricas.

Evaluación del rendimiento de la red con CloudWatch

Cuando el uso del sistema se acerca a los límites de recursos para el tipo de instancia, el procesamiento puede ralentizarse. Puede utilizar CloudWatch Logs Insights para supervisar el uso de sus recursos de almacenamiento y asegurarse de que haya suficientes recursos disponibles. Cuando sea necesario, se puede cambiar la instancia de base de datos por una clase de instancia mayor.

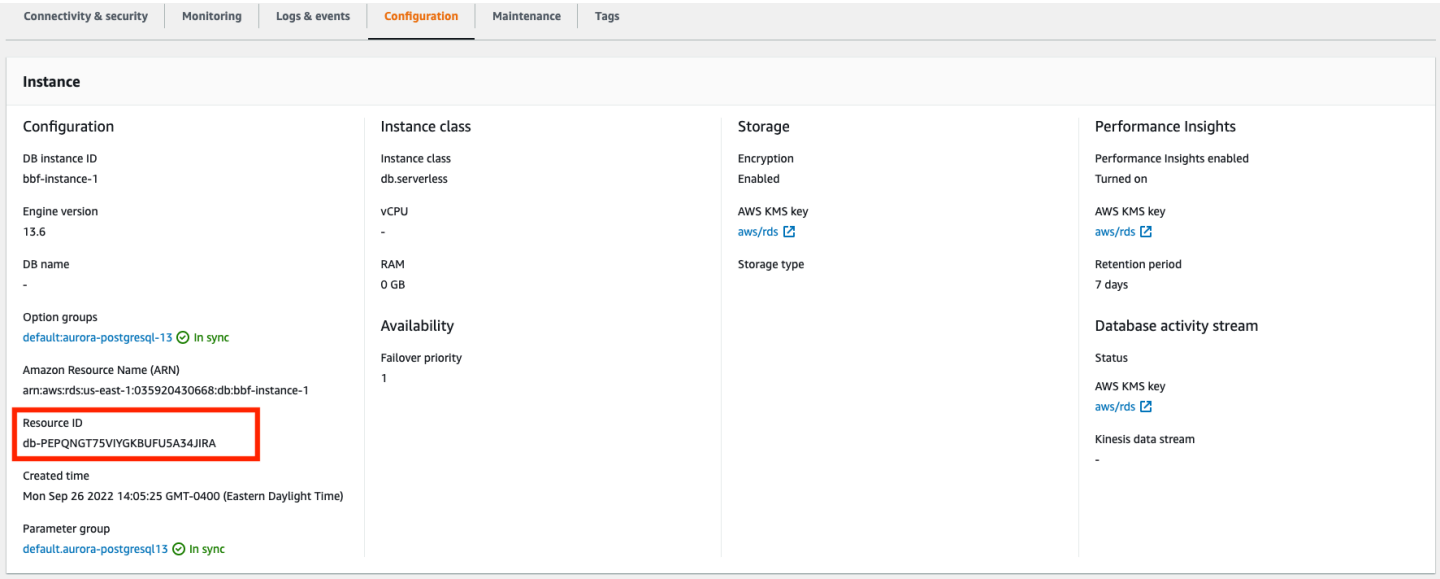
El procesamiento del almacenamiento de Aurora puede ser lento debido a:

- El ancho de banda de la red entre el cliente y la base de datos es insuficiente.
- El ancho de banda de red al subsistema de almacenamiento es insuficiente.
- Hay una carga de trabajo grande para su tipo de instancia.

Puede consultar CloudWatch Logs Insights para generar un gráfico del uso de los recursos de almacenamiento de Aurora con el fin de supervisar los recursos. El gráfico muestra el uso de la CPU y las métricas para ayudarte a decidir si debe escalar verticalmente a un tamaño de instancia mayor. Para obtener información sobre la sintaxis de consulta de CloudWatch Logs Insights, consulte Sintaxis de consulta de CloudWatch Logs Insights.

Para usar CloudWatch, debe exportar los archivos de registro de Aurora PostgreSQL a CloudWatch. También se puede modificar el clúster existente para exportar registros a CloudWatch. Para obtener más información acerca de la exportación de registros a CloudWatch, consulte Activación de la opción de publicación de registros en Amazon CloudWatch.

Necesita el Resource ID (ID de recurso) de su instancia de base de datos para consultar CloudWatch Logs Insights. El Resource ID (ID de recurso) está disponible en la pestaña Configuration (Configuración) de la consola:



Para consultar las métricas de almacenamiento de recursos en sus archivos de registro:

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.

   Aparece la página de inicio de información general de CloudWatch.
2. Si es necesario, cambie la Región de AWS. En la barra de navegación, elija la Región de AWS donde se encuentran sus recursos de AWS. Para obtener más información, consulte Regiones y puntos de enlace.
3. En el panel de navegación, elija Logs (Registros) y, luego, Logs Insights.

   Aparece la página Logs Insights.

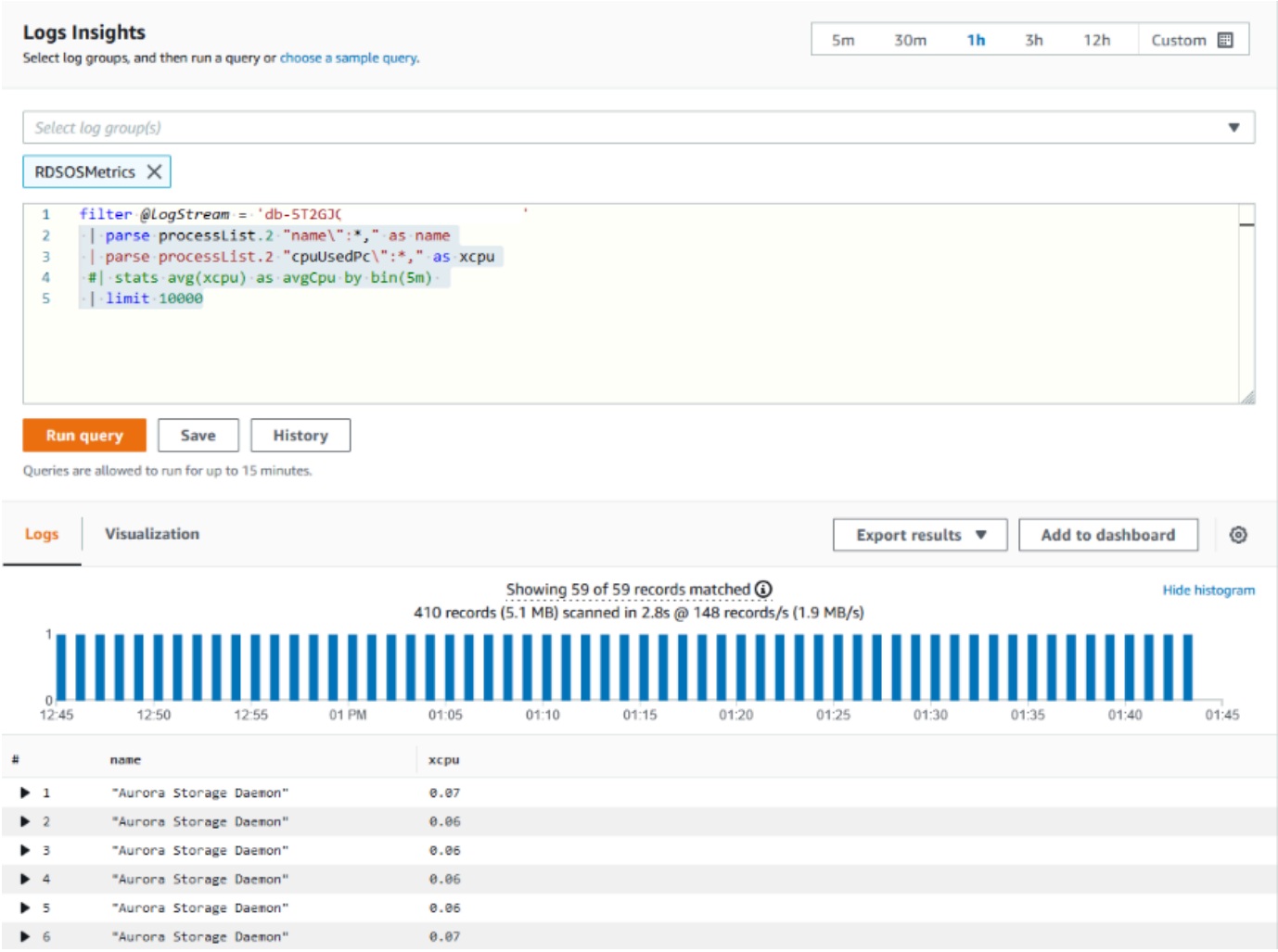
4. Seleccione los archivos de registro de la lista desplegable para analizarlos.
5. Introduzca la siguiente consulta en el campo y sustittuya `<resource ID>` por el ID de recurso de su clúster de base de datos:

   ```
   filter @logStream = <resource ID> | parse @message "\"Aurora Storage Daemon\"*memoryUsedPc\":*,\"cpuUsedPc\":*," as a,memoryUsedPc,cpuUsedPc | display memoryUsedPc,cpuUsedPc #| stats avg(xcpu) as avgCpu by bin(5m) | limit 10000
   ```

6. Haga clic en Run query (Ejecutar consulta).

   Se muestra el gráfico de utilización del almacenamiento.

   La siguiente imagen muestra la página Logs Insights y la visualización del gráfico.



Evaluación de la utilización de instancias de base de datos para Aurora PostgreSQL con métricas de CloudWatch

Puede usar las métricas de CloudWatch para observar el rendimiento de su instancia y descubrir si su clase de instancia proporciona recursos suficientes para sus aplicaciones. Para obtener información sobre los límites de la clase de instancia de base de datos, vaya a Especificaciones de hardware para clases de instancia de base de datos para Aurora y busque las especificaciones de la clase de instancia de base de datos para encontrar el rendimiento de la red.

Si el uso de la instancia de base de datos está cerca del límite de las clases de instancia, es posible que el rendimiento comience a disminuir. Las métricas de CloudWatch pueden confirmar esta situación para que pueda planificar el escalado vertical manual a una clase de instancia más grande.

Combine los siguientes valores de métricas de CloudWatch para averiguar si se acerca al límite de clases de instancia:

- NetworkThroughput: rendimiento de red que reciben y transmiten los clientes para cada instancia en el clúster de base de datos de Aurora. Este valor de rendimiento no incluye el tráfico de red entre las instancias del clúster de bases de datos y el volumen de clúster.
- StorageNetworkThroughput: rendimiento de red que recibe el subsistema de almacenamiento de Aurora y que cada instancia del clúster de bases de datos de Aurora envía al subsistema de almacenamiento de Aurora.

Sume la métrica NetworkThroughput a StorageNetworkThroughput para determinar el rendimiento de red recibido y enviado al subsistema de almacenamiento de Aurora por cada instancia del clúster de base de datos de Aurora. El límite de clases de instancia para su instancia debe ser mayor que la suma de estas dos métricas combinadas.

Puede utilizar las siguientes métricas para revisar detalles adicionales del tráfico de red de las aplicaciones cliente al enviar y recibir:

- NetworkReceiveThroughput: rendimiento de red recibido de los clientes por cada instancia del clúster de base de datos de Aurora PostgreSQL. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de y el volumen de clúster.
- NetworkTransmitThroughput: rendimiento de red enviado a los clientes por cada instancia del clúster de bases de datos de Aurora. Este desempeño no incluye el tráfico de red entre las instancias del clúster de bases de datos de y el volumen de clúster.

- StorageNetworkReceiveThroughput: rendimiento de red recibido del subsistema de almacenamiento de Aurora por cada instancia del clúster de bases de datos.
- StorageNetworkTransmitThroughput: rendimiento de red enviado al subsistema de almacenamiento de Aurora por cada instancia del clúster de bases de datos.

Sume todas estas métricas para evaluar cómo está el uso de la red con respecto al límite de las clases de instancias. El límite de las clases de instancias debe ser mayor que la suma de estas métricas combinadas.

Los límites de la red y el uso de la CPU para el almacenamiento son mutuos. Cuando aumenta el rendimiento de la red, también aumenta la utilización de la CPU. La supervisión del uso de la CPU y la red proporciona información sobre cómo y por qué se agotan los recursos.

Para ayudar a minimizar el uso de la red, puede considerar lo siguiente:

- Utilizar una clase de instancia mayor.
- Utilizar estrategias de partición `pg_partman`.
- Dividir las solicitudes de escritura en lotes para reducir las transacciones generales.
- Redirigir la carga de trabajo de solo lectura a una instancia de solo lectura.
- Eliminar los índices no utilizados.
- Comprobar si hay objetos sobrecargados y VACUUM. En el caso de una sobrecarga grave, utilice la extensión de PostgreSQL `pg_repack`. Para obtener más información acerca de `pg_repack`, consulte Reorganize tables in PostgreSQL databases with minimal locks (Reorganizar tablas en bases de datos PostgreSQL con los bloqueos mínimos).

Uso de la replicación lógica para realizar una actualización de la versión principal para Aurora PostgreSQL

Al usar la replicación lógica y la clonación rápida de Aurora, puede ejecutar una actualización de la versión principal, que utiliza la versión actual de la base de datos de Aurora PostgreSQL, mientras migra gradualmente los datos cambiantes a la base de datos de la nueva versión principal. Este proceso de actualización con un tiempo de inactividad bajo se denomina actualización azul/verde. La versión actual de la base de datos se denomina entorno “azul” y la nueva versión de la base de datos, entorno “verde”.

La clonación rápida de Aurora carga completamente los datos existentes mediante una instantánea de la base de datos de origen. La clonación rápida utiliza un protocolo de copia en escritura construido sobre la capa de almacenamiento de Aurora, que permite crear un clon de la base de datos en poco tiempo. Este método es muy eficaz cuando se actualiza a una base de datos de gran tamaño.

La replicación lógica en PostgreSQL rastrea y transfiere los cambios en los datos de la instancia inicial a una nueva instancia que se ejecuta en paralelo hasta que pase a la versión más reciente de PostgreSQL. La replicación lógica usa un modelo de publicación y suscripción. Para obtener más información acerca de la replicación lógica de Aurora PostgreSQL, consulte Replicación con Amazon Aurora PostgreSQL.

> **Tip**
>
> Puede minimizar el tiempo de inactividad necesario para la actualización de una versión principal mediante la característica de implementación azul/verde administrada de Amazon RDS. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Temas

- Requisitos
- Limitaciones
- Configuración y comprobación de valores de parámetros
- Actualización de Aurora PostgreSQL a una nueva versión principal
- Realización de tareas posteriores a la actualización

Requisitos

Debe cumplir los siguientes requisitos para llevar a cabo este proceso de actualización con bajo tiempo de inactividad:

- Debe tener permisos de rds_superuser.
- El clúster de base de datos de Aurora PostgreSQL que pretende actualizar debe ejecutar una versión compatible que pueda realizar actualizaciones de versiones importantes mediante la replicación lógica. Asegúrese de aplicar todas las actualizaciones y

revisiones de versiones secundarias al clúster de base de datos. La característica `aurora_volume_logical_start_lsn` que se utiliza en esta técnica es compatible con las siguientes versiones de Aurora PostgreSQL:

- Versión 15.2 y versiones posteriores a la 15
- Versión 14.3 y versiones posteriores a la 14
- Versión 13.6 y versiones posteriores a la 13
- Versión 12.10 y versiones posteriores a la 12
- Versión 11.15 y versiones posteriores a la 11
- Versión 10.20 y versiones posteriores a la 10

Para obtener más información sobre las funciones de `aurora_volume_logical_start_lsn`, consulte aurora_volume_logical_start_lsn.

- Todas las tablas deben tener una clave principal o incluir una columna de identidad de PostgreSQL.
- Configure el grupo de seguridad de su VPC para permitir el acceso entrante y saliente entre los dos clústeres de base de datos de Aurora PostgreSQL, tanto antiguos como nuevos. Puede conceder acceso a un rango específico de enrutamiento entre dominios sin clases (CIDR) o a otro grupo de seguridad de su VPC o de una VPC del mismo nivel. (La VPC del mismo nivel requiere una conexión del mismo nivel de VPC).

> **Note**
>
> Para obtener información detallada sobre los permisos necesarios para configurar y administrar un escenario de replicación lógica en ejecución, consulte la documentación básica de PostgreSQL.

Limitaciones

Al realizar una actualización con tiempo de inactividad bajo en el clúster de base de datos de Aurora PostgreSQL a una nueva versión principal, está utilizando la característica de replicación lógica nativa de PostgreSQL. Tiene las mismas capacidades y limitaciones que la replicación lógica de PostgreSQL. Para obtener más información, consulte Uso de la replicación lógica de PostgreSQL.

- Los comandos del lenguaje de definición de datos no se replican.

- La replicación no admite cambios de esquema en una base de datos activa. El esquema se vuelve a crear en su formato original durante el proceso de clonación. Si cambia el esquema después de la clonación, pero antes de completar la actualización, no se reflejará en la instancia actualizada.
- Los objetos grandes no se replican, pero se pueden almacenar datos en tablas normales.
- La replicación solo es compatible con tablas, incluidas las tablas particionadas. No se admite la replicación en otros tipos de relaciones, como vistas, vistas materializadas o tablas externas.
- Los datos de la secuencia no se replican y requieren una actualización manual después de la conmutación por error.

> **Note**
>
> Esta actualización no admite la creación automática de scripts. Hay que realizar todos los pasos de forma manual.

Configuración y comprobación de valores de parámetros

Antes de realizar la actualización, configure la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que funcione como un servidor de publicación. La instancia debe utilizar un grupo de parámetros de clúster de base de datos personalizado con la siguiente configuración:

- `rds.logical_replication`: establezca este parámetro en 1. El parámetro `rds.logical_replication` tiene el mismo propósito que el parámetro `wal_level` de un servidor PostgreSQL independiente y otros parámetros que controlan la administración de archivos de registro de escritura anticipada.
- `max_replication_slots`: establezca este parámetro en el número total de suscripciones que tiene previsto crear. Si utiliza AWS DMS, establezca este parámetro en el número de tareas de AWS DMS que tiene pensado utilizar para la captura de datos cambiados desde este clúster de base de datos.
- `max_wal_senders`: establezca el número de conexiones simultáneas, más algunas adicionales, para que estén disponibles para tareas de administración y sesiones nuevas. Si utiliza AWS DMS, el número de max_wal_senders debe ser igual al número de sesiones simultáneas más el número de tareas AWS DMS que pueden estar funcionando en un momento dado.
- `max_logical_replication_workers`: establezca el número de trabajadores de replicación lógica y trabajadores de sincronización de tablas que espera tener. Por lo general, es seguro establecer el número de trabajadores de replicación en el mismo valor que se utiliza para

max_wal_senders. Los trabajadores se toman del conjunto de procesos en segundo plano (max_worker_processes) asignado para el servidor.

- `max_worker_processes`: establezca el número de procesos en segundo plano para el servidor. Este número debe ser lo suficientemente grande como para asignar trabajadores a la replicación, los procesos auto-vacuum y otros procesos de mantenimiento que puedan llevarse a cabo simultáneamente.

Al actualizar a una versión más reciente de Aurora PostgreSQL, debe duplicar los parámetros que haya modificado en la versión anterior del grupo de parámetros. Estos parámetros se aplican a la versión actualizada. Puede consultar la tabla `pg_settings` para obtener una lista de las configuraciones de parámetros para poder volver a crearlos en el nuevo clúster de base de datos de Aurora PostgreSQL.

Por ejemplo, para obtener la configuración de los parámetros de replicación, ejecute la siguiente consulta:

```
SELECT name, setting FROM pg_settings WHERE name in
('rds.logical_replication', 'max_replication_slots',
'max_wal_senders', 'max_logical_replication_workers',
'max_worker_processes');
```

Actualización de Aurora PostgreSQL a una nueva versión principal

Para preparar el publicador (azul)

1. En el ejemplo siguiente, la instancia del escritor de origen (azul) es un clúster de base de datos de Aurora PostgreSQL que ejecuta la versión 11.15 de PostgreSQL. Este es el nodo de publicación de nuestro escenario de replicación. Para esta demostración, nuestra instancia de escritor de origen aloja una tabla de ejemplo que contiene una serie de valores:

   ```
   CREATE TABLE my_table (a int PRIMARY KEY);
   INSERT INTO my_table VALUES (generate_series(1,100));
   ```

2. Para crear una publicación en la instancia de origen, conéctese al nodo de escritura de la instancia con psql (la CLI de PostgreSQL o con el cliente que elija). Introduzca el siguiente comando en cada base de datos:

   ```
   CREATE PUBLICATION publication_name FOR ALL TABLES;
   ```

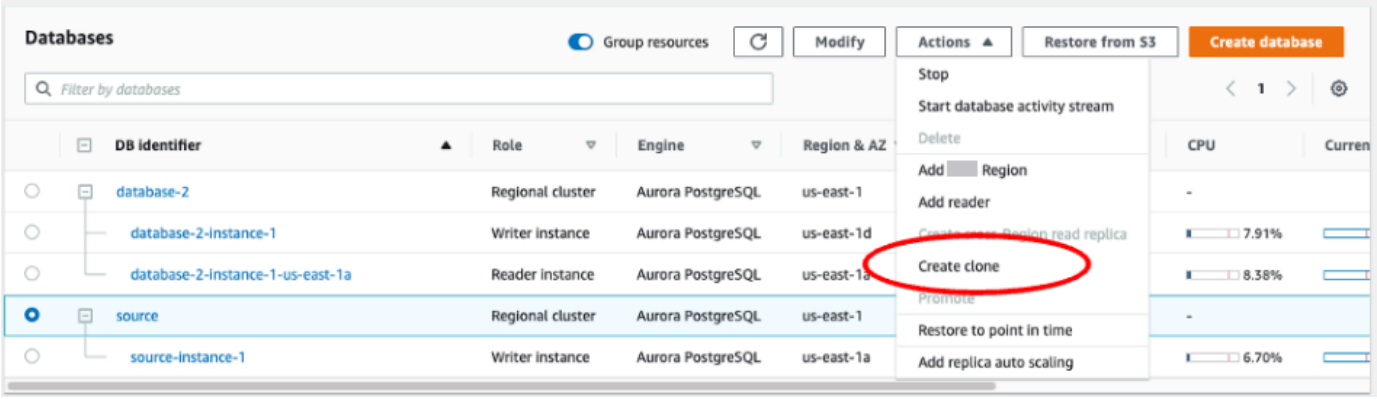
publication_name especifica el nombre de la publicación.

3. También debe crear una ranura de replicación en la instancia. El siguiente comando crea una ranura de replicación y carga el plugin de descodificación lógica `pgoutput`. El plugin cambia el contenido leído del registro de lectura anticipada (WAL) al protocolo de replicación lógica y filtra los datos de acuerdo con la especificación de la publicación.

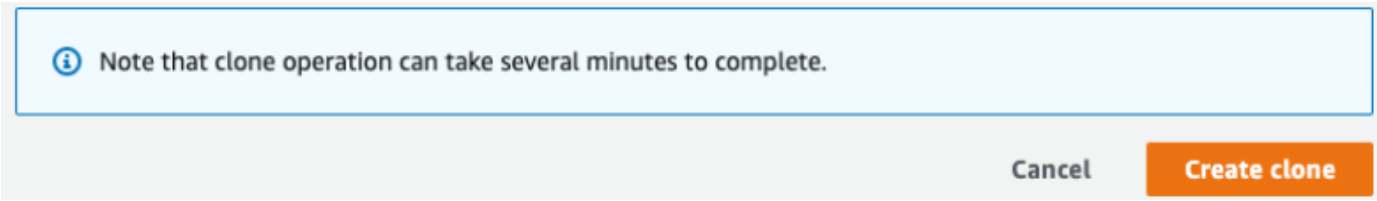
```
SELECT pg_create_logical_replication_slot('replication_slot_name', 'pgoutput');
```

Para clonar el publicador

1. Use la consola de Amazon RDS para crear un clon de la instancia de origen. Resalte el nombre de la instancia en la consola de Amazon RDS y, a continuación, elija Create clone (Crear clon) en el menú Actions (Acciones).



2. Introduzca un nombre único para la instancia. La mayoría de los ajustes son los valores predeterminados de la instancia de origen. Cuando haya realizado los cambios necesarios para la nueva instancia, seleccione Create clone (Crear clon).



3. Mientras se inicia la instancia de destino, la columna Status (Estado) del nodo de escritura muestra Creating (Creando) en la columna Status (Estado). Cuando la instancia esté lista, su estado cambiará a Available (Disponible).

Para preparar el clon para una actualización

1. El clon es la instancia “verde” del modelo de implementación. Es el host del nodo de suscripción de replicación. Cuando el nodo esté disponible, conéctese con psql y consulte el nuevo nodo de escritor para obtener el número de secuencia de registro (LSN). El LSN identifica el principio de un registro en el flujo de WAL.

   ```
   SELECT aurora_volume_logical_start_lsn();
   ```

2. En la respuesta de la consulta, encontrará el número LSN. Necesitará este número más adelante en el proceso, así que anótelo.

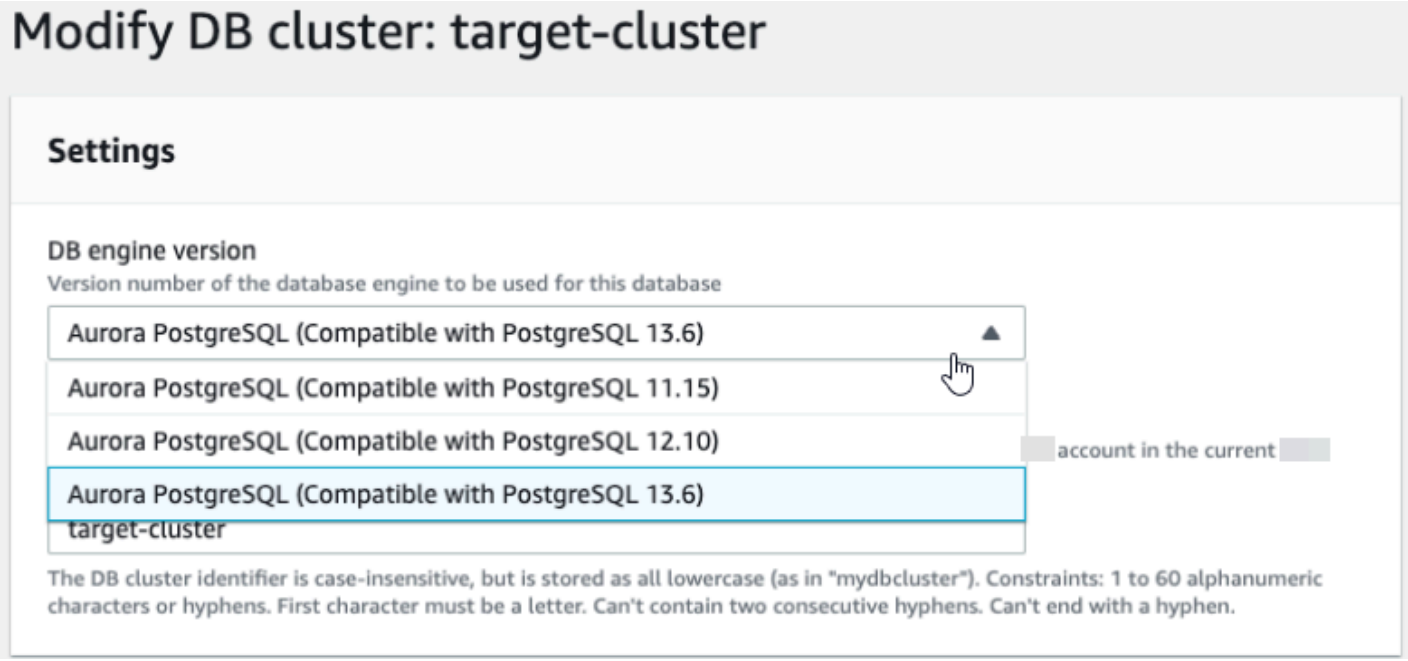
   ```
   postgres=> SELECT aurora_volume_logical_start_lsn();
   aurora_volume_logical_start_lsn
   ---------------
   0/402E2F0
   (1 row)
   ```

3. Antes de actualizar el clon, elimine la ranura de replicación del clon.

   ```
   SELECT pg_drop_replication_slot('replication_slot_name');
   ```

Para actualizar un clúster a una nueva versión principal

- Tras clonar el nodo del proveedor, utilice la consola de Amazon RDS para iniciar una actualización de la versión principal en el nodo de suscripción. Resalte el nombre de la instancia en la consola de RDS y seleccione el botón Modify (Modificar). Seleccione la versión actualizada y los grupos de parámetros actualizados y aplique la configuración inmediatamente para actualizar la instancia de destino.



- También puede utilizar la CLI para realizar una actualización:

```
aws rds modify-db-cluster –db-cluster-identifier $TARGET_Aurora_ID –engine-version
 13.6 –allow-major-version-upgrade –apply-immediately
```

Para preparar al suscriptor (verde)

1. Cuando el clon esté disponible tras la actualización, conéctese con psql y defina la suscripción. Para ello, debe especificar las siguientes opciones en el comando `CREATE SUBSCRIPTION`:

   - `subscription_name`: el nombre de la suscripción.
   - `admin_user_name`: el nombre de un usuario administrativo con permisos de rds_superuser.
   - `admin_user_password`: la contraseña asociada al usuario administrativo.
   - `source_instance_URL`: la URL de la instancia del servidor de publicaciones.
   - `database`: la base de datos a la que se conectará el servidor de suscripciones.
   - `publication_name`: el nombre del servidor de publicación.
   - `replication_slot_name`: el nombre del grupo de replicación.

```
CREATE SUBSCRIPTION subscription_name
CONNECTION 'postgres://admin_user_name:admin_user_password@source_instance_URL/
database' PUBLICATION publication_name
```

```
WITH (copy_data = false, create_slot = false, enabled = false, connect = true,
 slot_name = 'replication_slot_name');
```

2. Tras crear la suscripción, consulte la vista pg_replication_origin para recuperar el valor roname, que es el identificador del origen de la replicación. Cada instancia tiene un solo `roname`:

   ```
   SELECT * FROM pg_replication_origin;
   ```

   Por ejemplo:

   ```
   postgres=>
   SELECT * FROM pg_replication_origin;

   roident | roname
   ---------+----------
   1 | pg_24586
   ```

3. Introduzca el LSN que guardó de la consulta anterior del nodo de publicación y el `roname` devuelto desde [INSTANCIA] del nodo de suscripción en el comando. Este comando usa la función `pg_replication_origin_advance` para especificar el punto inicial de la secuencia de registro para la replicación.

   ```
   SELECT pg_replication_origin_advance('roname', 'log_sequence_number');
   ```

   `roname` es el identificador devuelto por la vista pg_replication_origin.

   `log_sequence_number` es el valor devuelto por la consulta anterior de la función `aurora_volume_logical_start_lsn`.

4. A continuación, utilice la cláusula `ALTER SUBSCRIPTION... ENABLE` para activar la replicación lógica.

   ```
   ALTER SUBSCRIPTION subscription_name ENABLE;
   ```

5. En este punto, puede confirmar que la replicación funciona. Añada un valor a la instancia de publicación y, a continuación, confirme que el valor se replica en el nodo de suscripción.

   A continuación, utilice el siguiente comando para supervisar el retardo de replicación en el nodo de publicación:

```
SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';
```

Por ejemplo:

```
postgres=> SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
 confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';

current_time                   | slot_name             | active | active_pid |
 diff_size | diff_bytes
-------------------------------+-----------------------+--------+------------
+-----------+------------
2022-04-13 15:11:00.243401+00  | replication_slot_name | t      | 21854      | 136
 bytes | 136
(1 row)
```

Puede supervisar el retardo de replicación mediante los valores `diff_size` y `diff_bytes`. Cuando estos valores lleguen a 0, la réplica estará funcionando al mismo ritmo que la instancia de base de datos de origen.

Realización de tareas posteriores a la actualización

Cuando finalice la actualización, el estado de la instancia aparecerá como Available (Disponible) en la columna Status (Estado) del panel de control de la consola. En la nueva instancia, recomendamos que haga lo siguiente:

- Redirija sus aplicaciones para que apunten al nodo de escritor.
- Añada nodos de lector para administrar el número de casos y ofrecer una alta disponibilidad en caso de que surja un problema con el nodo de escritor.
- En ocasiones, los clústeres de base de datos de Aurora PostgreSQL requieren actualizaciones del sistema operativo. Estas actualizaciones a veces pueden incluir una versión más reciente de la

biblioteca glibc. Durante estas actualizaciones, le recomendamos que siga las directrices que se describen en Intercalaciones admitidas en Aurora PostgreSQL.

- Actualice los permisos de usuario en la nueva instancia para garantizar el acceso.

Tras probar la aplicación y los datos en la nueva instancia, le recomendamos que realice una copia de seguridad final de la instancia inicial antes de eliminarla. Para obtener más información acerca del uso de la replicación lógica en un host de Aurora, consulte Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL.

Solución de problemas de almacenamiento en Aurora PostgreSQL

Si la cantidad de memoria de trabajo que necesitan las operaciones de ordenación o creación de índices supera la cantidad asignada por el parámetro `work_mem`, Aurora PostgreSQL escribe los datos que sobran en archivos de disco temporales. Cuando escribe los datos, Aurora PostgreSQL usa el mismo espacio de almacenamiento que para almacenar los registros de errores y mensajes, es decir, el almacenamiento local. Cada instancia de su clúster de base de datos de Aurora PostgreSQL tiene una cantidad de almacenamiento local disponible. La cantidad de almacenamiento se basa en su clase de instancia de base de datos. Para aumentar la cantidad de almacenamiento local, debe modificar la instancia para usar una clase de instancia de base de datos más grande. Para ver las especificaciones de clase de instancia de base de datos, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

Puede supervisar el espacio de almacenamiento local de su clúster de base de datos de Aurora PostgreSQL observando la métrica de Amazon CloudWatch para `FreeLocalStorage`. Esta métrica indica la cantidad de almacenamiento disponible en cada instancia de base de datos para las tablas y los registros temporales del clúster de base de datos de Aurora. Para obtener más información, consulte Supervisión de métricas de Amazon Aurora con Amazon CloudWatch.

Las operaciones de ordenación, indexación y agrupación comienzan en la memoria de trabajo, pero a menudo deben descargarse al almacenamiento local. Si su clúster de base de datos de Aurora PostgreSQL se queda sin almacenamiento local debido a este tipo de operaciones, puede resolver el problema mediante una de las siguientes acciones.

- Aumente la cantidad de memoria de trabajo. Esto reduce la necesidad de utilizar el almacenamiento local. De forma predeterminada, PostgreSQL asigna 4 MB para cada operación de ordenación, agrupación e indexado. Para comprobar el valor actual de la memoria de trabajo de la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL, conéctese a la instancia utilizando `psql` y ejecute el siguiente comando.

```
postgres=> SHOW work_mem;
work_mem
----------
 4MB
(1 row)
```

Puede aumentar la memoria de trabajo a nivel de sesión antes de ordenar, agrupar y realizar otras operaciones, como se indica a continuación.

```
SET work_mem TO '1 GB';
```

Para obtener más información sobre la memoria de trabajo, consulte Consumo de recursos en la documentación de PostgreSQL.

- Cambie el período de retención de registros para que los registros se almacenen durante períodos de tiempo más cortos. Para aprender a hacerlo, consulte Archivos de registro de bases de datos de Aurora PostgreSQL.

Para clústeres de base de datos de Aurora PostgreSQL de más de 40 TB, no utilice las clases de instancia db.t2, db.t3 o db.t4g. Recomendamos que las clases de instancia de base de datos T se utilicen solo para los servidores de desarrollo y de pruebas, o para otros servidores que no se utilicen para la producción. Para obtener más información, consulte Tipos de clase de instancia de base de datos.

Replicación con Amazon Aurora PostgreSQL

A continuación, encontrará información sobre la replicación con Amazon Aurora PostgreSQL, lo que incluye cómo supervisar y utilizar la replicación lógica.

Temas

- Uso de réplicas de Aurora
- Mejora de la disponibilidad de lectura de las réplicas de Aurora
- Monitoreo de replicación de Aurora PostgreSQL
- Información general sobre la replicación lógica de PostgreSQL con Aurora
- Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL
- Desactivación de la replicación lógica

- Supervisión de la memoria caché de escritura indirecta y de las ranuras lógicas para la replicación lógica de Aurora PostgreSQL
- Ejemplo: uso de la replicación lógica con clústeres de base de datos de Aurora PostgreSQL
- Ejemplo: replicación lógica mediante Aurora PostgreSQL y AWS Database Migration Service
- Configuración de la autenticación de IAM para conexiones de replicación lógica

Uso de réplicas de Aurora

Una réplica de Aurora es un punto de enlace independiente en un clúster de base de datos Aurora que se utilizan preferentemente para ajustar la escala de las operaciones de lectura e incrementar la disponibilidad. Un clúster de base de datos Aurora puede incluir hasta 15 réplicas de Aurora ubicadas en las zonas de disponibilidad de la región de AWS del clúster de base de datos Aurora.

El volumen del clúster de base de datos consta de varias copias de los datos del clúster de base de datos. No obstante, los datos del volumen del clúster se representan como un único volumen lógico para la instancia de base de datos de escritor principal y para las réplicas de Aurora del clúster de base de datos. Para obtener más información acerca de las réplicas de Aurora, consulte Réplicas de Aurora.

Las réplicas de Aurora funcionan bien para el escalado de lectura porque están totalmente dedicadas a las operaciones de lectura en el volumen del clúster. La instancia de base de datos de escritor administra operaciones de escritura. El volumen del clúster se comparte entre todas las instancias en su clúster de base de datos Aurora PostgreSQL. Por lo tanto, no se necesita trabajo adicional para replicar una copia de los datos de cada réplica Aurora.

Con Aurora PostgreSQL, cuando se elimina una réplica de Aurora, su punto de enlace de instancia se quita inmediatamente y la réplica de Aurora se quita del punto de enlace del lector. Si hay instrucciones que se ejecutan en la réplica de Aurora que se van a eliminar, hay un periodo de gracia de tres minutos. Las instrucciones existentes pueden finalizar correctamente durante el periodo de gracia. Cuando termina dicho periodo, se apaga la réplica de Aurora y se elimina.

Los clústeres de base de datos de Aurora PostgreSQL admiten réplicas de Aurora en diferentes regiones de AWS con la base de datos global de Aurora. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

> **Note**
>
> Con la característica de disponibilidad de lectura, si quiere reiniciar las réplicas de Aurora en el clúster de base de datos, debe hacerlo manualmente. Para los clústeres de base de datos creados antes de esta característica, al reiniciar la instancia de base de datos del escritor se reinician automáticamente las réplicas de Aurora. El reinicio automático restablece un punto de entrada que garantiza la coherencia de lectura/escritura en todo el clúster de base de datos.

Mejora de la disponibilidad de lectura de las réplicas de Aurora

Aurora PostgreSQL mejora la disponibilidad de lectura en el clúster de base de datos al atender continuamente las solicitudes de lectura cuando la instancia de base de datos del escritor se reinicia o cuando la réplica de Aurora no puede seguir el ritmo del tráfico de escritura.

La característica de disponibilidad de lectura está disponible de forma predeterminada en las siguientes versiones de Aurora PostgreSQL:

- Versión 16.1 y todas las versiones posteriores
- Versión 15.2 y versiones posteriores a la 15
- Versión 14.7 y versiones posteriores a la 14
- Versión 13.10 y versiones posteriores a la 13
- Versión 12.14 y versiones posteriores a la 12

La característica de disponibilidad de lectura es compatible con la base de datos global Aurora en las siguientes versiones:

- Versión 16.1 y todas las versiones posteriores
- Versión 15.4 y versiones posteriores a la 15
- Versión 14.9 y versiones posteriores a la 14
- Versión 13.12 y versiones posteriores a la 13
- Versión 12.16 y versiones posteriores a la 12

Para utilizar la característica de disponibilidad de lectura para un clúster de base de datos creado en una de estas versiones antes de este lanzamiento, reinicie la instancia de escritura del clúster de base de datos.

Al modificar los parámetros estáticos de su clúster de base de datos de Aurora PostgreSQL, debe reiniciar la instancia de escritor para que surtan efecto los cambios de parámetros. Por ejemplo, debe reiniciar la instancia del escritor al establecer el valor de `shared_buffers`. Con la característica de disponibilidad de lectura de las réplicas de Aurora, el clúster de base de datos mantiene la disponibilidad mejorada, lo que reduce el impacto al reiniciar la instancia de escritor. Las instancias del lector no se reinician y siguen respondiendo a las solicitudes de lectura. Para aplicar cambios en los parámetros estáticos, reinicie cada instancia individual del lector.

Una réplica de Aurora de un clúster de base de datos de Aurora PostgreSQL puede recuperarse de errores de replicación, como el reinicio del escritor, la conmutación por error, la replicación lenta y los problemas de red, ya que recupera rápidamente el estado de la base de datos en memoria una vez que se vuelve a conectar con el escritor. Este enfoque permite que las instancias de la réplica de Aurora sean coherentes con las actualizaciones de almacenamiento más recientes mientras la base de datos del cliente aún esté disponible.

Es posible que las transacciones en curso que entren en conflicto con la recuperación de la replicación reciban un error, pero el cliente puede volver a intentar estas transacciones una vez que los lectores alcancen al escritor.

Supervisión de réplicas de Aurora

Puede monitorizar las réplicas de Aurora cuando se recupere de una desconexión del escritor. Utilice las siguientes métricas para consultar la información más reciente sobre la instancia de lector y realizar un seguimiento de las transacciones de solo lectura en curso.

- La función `aurora_replica_status` se actualiza para que devuelva la información más actualizada de la instancia de lector cuando aún está conectada. La marca de tiempo de la última actualización en `aurora_replica_status` siempre está vacía para la fila correspondiente a la instancia de base de datos en la que se ejecuta la consulta. Esto significa que la instancia de lector tiene los datos más recientes.
- Cuando la réplica de Aurora se desconecta de la instancia de escritor y se vuelve a conectar, se emite el siguiente evento de base de datos:

  ```
  Read replica has been disconnected from the writer instance and
  reconnected.
  ```

- Cuando se cancela una consulta de solo lectura debido a un conflicto de recuperación, es posible que aparezca uno o más de los siguientes mensaje de error en el registro de errores de la base de datos:

  ```
  Canceling statement due to conflict with recovery.

  User query may not have access to page data to replica disconnect.

  User query might have tried to access a file that no longer exists.

  When the replica reconnects, you will be able to repeat your command.
  ```

Limitaciones

Las siguientes limitaciones se aplican a las réplicas de Aurora con la característica de disponibilidad de lectura mejorada:

- Las réplicas de Aurora del clúster de base de datos secundario se pueden reiniciar si los datos no se pueden transmitir desde la instancia de escritor durante la recuperación de la replicación.
- Las réplicas de Aurora no admiten la recuperación de la replicación en línea si ya hay una en curso y se reiniciará.
- Las réplicas de Aurora se reiniciarán cuando la instancia de base de datos se acerque al reinicio del ID de la transacción. Para obtener más información acerca del reinicio del ID de la transacción, consulte el tema sobre prevención de fallos del reinicio del ID de la transacción.
- Las réplicas de Aurora pueden reiniciarse cuando el proceso de replicación está bloqueado bajo determinadas circunstancias.

Monitoreo de replicación de Aurora PostgreSQL

El escalado de lectura y la alta disponibilidad dependen de un tiempo de retardo mínimo. Puede monitorizar el retardo de una réplica de Aurora con respecto a la instancia de base de datos de escritor del clúster de base de datos Aurora PostgreSQL mediante la monitorización de la métrica `ReplicaLag` de Amazon CloudWatch. Como las réplicas de Aurora leen desde el mismo volumen de clúster que la instancia de base de datos de escritor, la métrica `ReplicaLag` tiene un significado diferente para un clúster de base de datos Aurora PostgreSQL. La métrica `ReplicaLag` de una réplica de Aurora indica el retardo de la caché de página de la réplica de Aurora con respecto a la de la instancia de base de datos de escritor.

Para obtener más información acerca de la monitorización de instancias de RDS y de las métricas de CloudWatch, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Información general sobre la replicación lógica de PostgreSQL con Aurora

Al utilizar la función de replicación lógica de PostgreSQL con su clúster de base de datos de Aurora PostgreSQL, puede replicar y sincronizar tablas individuales en lugar de toda la instancia de base de datos. La replicación lógica usa un modelo de publicación y suscripción para replicar los cambios de una fuente a uno o más destinatarios. Para ello, usa registros de cambios del registro de escritura anticipada (WAL) de PostgreSQL. La fuente, o publicador, envía los datos WAL de las tablas especificadas a uno o más destinatarios (suscriptor), replicando así los cambios y manteniendo la tabla del suscriptor sincronizada con la tabla del publicador. El conjunto de cambios del publicador se identifica mediante una publicación. Los suscriptores obtienen los cambios mediante la creación de una suscripción que define la conexión con la base de datos del publicador y sus publicaciones. Un intervalo de replicación es el mecanismo que se utiliza en este esquema para realizar un seguimiento del progreso de una suscripción.

Para los clústeres de bases de datos de Aurora PostgreSQL, los registros WAL se guardan en el almacenamiento de Aurora. El clúster de base de datos de Aurora PostgreSQL que actúa como publicador en un escenario de replicación lógica lee los datos de WAL del almacenamiento de Aurora, los decodifica y los envía al suscriptor para que los cambios se puedan aplicar a la tabla de esa instancia. El publicador utiliza un decodificador lógico para decodificar los datos y que los suscriptores puedan utilizarlos. De forma predeterminada, los clústeres de bases de datos de Aurora PostgreSQL utilizan el complemento de PostgreSQL `pgoutput` nativo al enviar datos. Hay otros decodificadores lógicos disponibles. Por ejemplo, Aurora PostgreSQL también admite el complemento `wal2json` que convierte los datos WAL a JSON.

A partir de las versiones 14.5, 13.8, 12.12 y 11.17 de Aurora PostgreSQL, Aurora PostgreSQL amplía el proceso de replicación lógica de PostgreSQL con una memoria caché de escritura para mejorar el rendimiento. Los registros de transacciones de WAL se almacenan en caché localmente, en un búfer, para reducir la cantidad de E/S del disco, es decir, la lectura del almacenamiento de Aurora durante la decodificación lógica. La caché de escritura se usa de forma predeterminada cuando usa replicación lógica para el clúster de bases de datos de Aurora PostgreSQL. Aurora proporciona varias funciones que puede usar para administrar la caché. Para obtener más información, consulte Supervisión de la memoria caché de escritura indirecta de la replicación lógica en Aurora PostgreSQL.

La replicación lógica es compatible con todas las versiones de Aurora PostgreSQL disponibles actualmente. Para obtener información detallada sobre la versión, consulte las actualizaciones de Amazon Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

La replicación lógica es compatible con Babelfish para Aurora PostgreSQL a partir de las siguientes versiones:

- Versión 15.7 y posteriores
- Versión 16.3 y posteriores

> **Note**
>
> Además de la característica de replicación lógica nativa de PostgreSQL introducida en PostgreSQL 10, Aurora PostgreSQL también admite la extensión `pglogical`. Para obtener más información, consulte Uso de pglogical para sincronizar datos entre instancias.

Para obtener información adicional sobre la implementación de PostgreSQL en la replicación lógica, consulte la sección sobre replicación lógica y la sección sobre conceptos de descodificación lógica.

> **Note**
>
> PostgreSQL 16 ha agregado soporte para la decodificación lógica a partir de réplicas de lectura. Esta característica no se admite en Aurora PostgreSQL.

Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL

La configuración de la replicación lógica requiere privilegios de `rds_superuser`. El clúster de base de datos de Aurora PostgreSQL debe estar configurado para usar un grupo de parámetros de clúster de base de datos personalizado, de modo que pueda establecer los parámetros necesarios tal como se detalla en el procedimiento siguiente. Para obtener más información, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija el clúster de base de datos de Aurora PostgreSQL.
3. Haga clic en la pestaña Configuration (Configuración). En los detalles de la instancia, busque el enlace Grupo de parámetros con el Grupo de parámetros de clúster de base de datos para Tipo.
4. Elija el enlace para abrir los parámetros personalizados asociados al clúster de base de datos de Aurora PostgreSQL.
5. En el campo de búsqueda Parameters (Parámetros), escriba `rds` para buscar el parámetro `rds.logical_replication`. El valor predeterminado de este parámetro es `0`, lo que significa que está desactivado de forma predeterminada.
6. Elija Edit parameters (Editar parámetros) para acceder a los valores de las propiedades y, a continuación, elija 1 en el selector para activar la función. En función del uso previsto, es posible que también tenga que cambiar la configuración de los siguientes parámetros. Sin embargo, en muchos casos, los valores predeterminados son suficientes.

   - `max_replication_slots`: establezca este parámetro en un valor que sea al menos igual al número total planificado de publicaciones y suscripciones de replicación lógica. Si utiliza AWS DMS, este parámetro debe ser igual al menos a las tareas de captura de datos de cambios planificadas del clúster, además de las publicaciones y suscripciones de replicación lógica.
   - `max_wal_senders` y:`max_logical_replication_workers` establezca estos parámetros a un valor que sea al menos igual al número de ranuras de replicación lógica que desea queden activas o el número de tareas activas de AWS DMS para la captura de datos de cambios. Dejar inactiva una ranura de replicación lógica evita que vacuum elimine las tuplas obsoletas de las tablas, por lo que se recomienda supervisar las ranuras de replicación y eliminar las ranuras inactivas cuando sea necesario.
   - `max_worker_processes`: establezca este parámetro en un valor que sea al menos igual al total de los valores `max_logical_replication_workers`, `autovacuum_max_workers` y `max_parallel_workers`. En las clases de instancias de base de datos pequeñas, los procesos de trabajo en segundo plano pueden afectar a las cargas de trabajo de las aplicaciones, por lo que debe supervisarse el rendimiento de la base de datos si establece `max_worker_processes` en un valor superior al predeterminado. (El valor predeterminado es el resultado de `GREATEST(${DBInstanceVCPU*2},8}`, lo que significa que, de forma predeterminada, es 8 o dos veces el equivalente en CPU de la clase de instancia de base de datos, lo que sea mayor).

> **Note**
>
> Se pueden modificar los valores de los parámetros de un grupo de parámetros de base de datos creado por el cliente, pero no se pueden modificar los valores de los parámetros de un grupo de parámetros de base de datos predeterminado.

7. Seleccione Save changes (Guardar cambios).
8. Reinicie la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que los cambios surtan efecto. En la consola de Amazon RDS, elija la instancia de base de datos principal del clúster y elija Reboot (Reiniciar) en el menú Actions (Acciones).
9. Cuando la instancia esté disponible, puede comprobar que la replicación lógica esté activada de la siguiente manera.

   a. Use `psql` para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

   ```
   psql --host=your-db-cluster-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=postgres --password --dbname=labdb
   ```

   b. Compruebe que la replicación lógica esté habilitada mediante el siguiente comando.

   ```
   labdb=> SHOW rds.logical_replication;
    rds.logical_replication
   -------------------------
    on
   (1 row)
   ```

   c. Verifique que el parámetro `wal_level` esté establecido en `logical`.

   ```
   labdb=> SHOW wal_level;
     wal_level
   -----------
    logical
   (1 row)
   ```

Para ver un ejemplo del uso de la replicación lógica para mantener una tabla de base de datos sincronizada con los cambios de un clúster de base de datos de Aurora PostgreSQL de origen, consulte Ejemplo: uso de la replicación lógica con clústeres de base de datos de Aurora PostgreSQL.

Desactivación de la replicación lógica

Tras completar las tareas de replicación, debe detener el proceso de replicación, eliminar las ranuras de replicación y desactivar la replicación lógica. Antes de eliminar las ranuras, asegúrese de que ya no sean necesarias. Las ranuras de replicación activas no se pueden eliminar.

Desactivación de la replicación lógica

1. Elimine todas las ranuras de replicación.

   Para eliminar todas las ranuras de replicación, conéctese al publicador y ejecute el siguiente comando de SQL.

   ```
   SELECT pg_drop_replication_slot(slot_name)
     FROM pg_replication_slots
    WHERE slot_name IN (SELECT slot_name FROM pg_replication_slots);
   ```

   Las ranuras de replicación no pueden estar activas cuando ejecute este comando.
2. Modifique el grupo de parámetros de clúster de base de datos personalizado asociado al publicador tal como se detalla en Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL, pero establezca el parámetro `rds.logical_replication` en 0.

   Para obtener más información acerca de los grupos de parámetros personalizados, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.
3. Reinicie el clúster de base de datos de Aurora PostgreSQL del publicador para que se aplique el cambio en el parámetro `rds.logical_replication`.

Supervisión de la memoria caché de escritura indirecta y de las ranuras lógicas para la replicación lógica de Aurora PostgreSQL

Supervise la memoria caché de escritura indirecta de la replicación lógica y administre las ranuras lógicas para mejorar el rendimiento del clúster de base de datos de Aurora PostgreSQL. A

continuación, encontrará más información sobre la memoria caché de escritura indirecta y las ranuras lógicas.

Temas

- Supervisión de la memoria caché de escritura indirecta de la replicación lógica en Aurora PostgreSQL
- Administración de ranuras lógicas para Aurora PostgreSQL

Supervisión de la memoria caché de escritura indirecta de la replicación lógica en Aurora PostgreSQL

De forma predeterminada, las versiones 14.5, 13.8, 12.12 y 11.17 y posteriores de Aurora PostgreSQL utilizan una memoria caché de escritura para mejorar el rendimiento de la replicación lógica. Sin la memoria caché de escritura, Aurora PostgreSQL utiliza la capa de almacenamiento de Aurora en su implementación del proceso de replicación lógica nativo de PostgreSQL. Para ello, escribe los datos de WAL en el almacenamiento y, a continuación, los lee del almacenamiento para decodificarlos y enviarlos (replicarlos) a sus destinos (suscriptores). Esto puede provocar cuellos de botella durante la replicación lógica de los clústeres de bases de datos de Aurora PostgreSQL.

La memoria caché de escritura reduce al mínimo la dependencia de la capa de almacenamiento de Aurora. En lugar de escribir y leer de manera constante en esta capa, Aurora PostgreSQL utiliza un búfer para almacenar en caché el flujo WAL lógico de modo que pueda usarse durante el proceso de replicación, lo que reduce la necesidad de acceder al disco. Este búfer es la memoria caché nativa de PostgreSQL que utiliza la replicación lógica y que se identifica en los parámetros del clúster de base de datos de Aurora PostgreSQL como `rds.logical_wal_cache`.

Al utilizar la replicación lógica con el clúster de base de datos de Aurora PostgreSQL (para las versiones que admiten la caché de escritura), puede supervisar la tasa de aciertos de caché para comprobar cómo funciona en su caso de uso. Para ello, conéctese a la instancia de escritura del clúster de base de datos de Aurora PostgreSQL mediante la función de Aurora `psql` y, a continuación, utilice la función de Aurora `aurora_stat_logical_wal_cache`, como se muestra en el siguiente ejemplo.

```
SELECT * FROM aurora_stat_logical_wal_cache();
```

La función devuelve una salida como la siguiente:

```
name       | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
-----------+------------+-----------+------------+-----------+----------+--------------
test_slot1 | 79183      | 24        | 0          | 24        | 100.00%  | 2022-08-05
 17:39...
test_slot2 |            | 1         | 0          |  1        | 100.00%  | 2022-08-05
 17:34...
(2 rows)
```

Los valores de `last_reset_timestamp` se han acortado para facilitar la lectura. Para obtener más información acerca de esta función, consulte aurora_stat_logical_wal_cache.

Aurora PostgreSQL proporciona las dos funciones siguientes para supervisar la memoria caché de escritura.

- La función:`aurora_stat_logical_wal_cache` para obtener documentación de referencia, consulte aurora_stat_logical_wal_cache.
- La función:`aurora_stat_reset_wal_cache` para obtener documentación de referencia, consulte aurora_stat_reset_wal_cache.

Si descubre que el tamaño de la caché de WAL que se ha ajustado automáticamente no es suficiente para sus cargas de trabajo, puede cambiar el valor de `rds.logical_wal_cache` manualmente. Considere lo siguiente:

- Cuando el parámetro `rds.logical_replication` está deshabilitado, `rds.logical_wal_cache` se establece en cero (0).
- Cuando el parámetro `rds.logical_replication` está habilitado, `rds.logical_wal_cache` tiene un valor predeterminado de 16 MB.
- El parámetro `rds.logical_wal_cache` es estático y requiere un reinicio de la instancia de base de datos para que los cambios surtan efecto. Este parámetro se define en términos de bloques de 8 Kb. Tenga en cuenta que cualquier valor positivo inferior a 32 Kb se considera 32 Kb. Para obtener más información sobre `wal_buffers`, consulte Registro de escritura anticipada en la documentación de PostgreSQL.

Administración de ranuras lógicas para Aurora PostgreSQL

La actividad de streaming se captura en la vista `pg_replication_origin_status`. Para ver el contenido de esta vista, puede usar la función `pg_show_replication_origin_status()`, tal como se muestra a continuación:

```
SELECT * FROM pg_show_replication_origin_status();
```

Puede obtener una lista de las ranuras lógicas mediante la siguiente consulta SQL.

```
SELECT * FROM pg_replication_slots;
```

Para eliminar una ranura lógica, use `pg_drop_replication_slot` con el nombre de la ranura, como se muestra en el siguiente comando.

```
SELECT pg_drop_replication_slot('test_slot');
```

Ejemplo: uso de la replicación lógica con clústeres de base de datos de Aurora PostgreSQL

En el siguiente procedimiento se muestra cómo iniciar la replicación lógica entre dos clústeres de base de datos de Aurora PostgreSQL. Tanto el publicador como el suscriptor deben estar configurados para la replicación lógica, tal como se detalla en Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL.

El clúster de base de datos de Aurora PostgreSQL que es el publicador designado también debe permitir el acceso a la ranura de replicación. Para ello, modifique el grupo de seguridad asociado a la nube pública virtual (VPC) del clúster de base de datos de Aurora PostgreSQL en función del servicio Amazon VPC. Para permitir el acceso entrante, agregue el grupo de seguridad asociado a la VPC del suscriptor al grupo de seguridad del publicador. Para obtener más información, consulte Controlar el tráfico hacia los recursos mediante grupos de seguridad en la Guía del usuario de Amazon Virtual Private Cloud.

Una vez completados estos pasos preliminares, puede usar los comandos `CREATE PUBLICATION` de PostgreSQL en el publicador y `CREATE SUBSCRIPTION` en el suscriptor, tal como se detalla en el siguiente procedimiento.

Para iniciar la replicación lógica entre dos clústeres de base de datos de Aurora PostgreSQL.

En estos pasos se supone que los clústeres de base de datos de Aurora PostgreSQL tienen una instancia de escritor con una base de datos en la que crear las tablas de ejemplo.

1. En el clúster de base de datos de Aurora PostgreSQL de publicadores

   a. Cree una tabla con la siguiente instrucción SQL.

   ```
   CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
   ```

   b. Inserte datos en la base de datos de publicador mediante la siguiente instrucción SQL.

   ```
   INSERT INTO LogicalReplicationTest VALUES (generate_series(1,10000));
   ```

   c. Compruebe que los datos existen en la tabla mediante la siguiente instrucción SQL.

   ```
   SELECT count(*) FROM LogicalReplicationTest;
   ```

   d. Cree una publicación para esta tabla mediante la instrucción `CREATE PUBLICATION` que se indica a continuación.

   ```
   CREATE PUBLICATION testpub FOR TABLE LogicalReplicationTest;
   ```

2. En el clúster de base de datos Aurora PostgreSQL de suscriptor

   a. Cree la misma tabla `LogicalReplicationTest` en el suscriptor que creó en el publicador, de la siguiente manera.

   ```
   CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
   ```

   b. Compruebe que esta tabla esté vacía.

   ```
   SELECT count(*) FROM LogicalReplicationTest;
   ```

   c. Crea una suscripción para recibir los cambios del publicador. Debe utilizar los siguientes detalles sobre el clúster de base de datos Aurora PostgreSQL de publicador.

   - host: la instancia de base de datos de escritor del clúster de base de datos de Aurora pPostgreSQL del publicador.

- puerto: el puerto en el que la instancia de base de datos de escritor está a la escucha. El valor predeterminado de PostgreSQL es 5432.
- dbname: el nombre de la base de datos.

```
CREATE SUBSCRIPTION testsub CONNECTION
   'host=publisher-cluster-writer-endpoint port=5432 dbname=db-name user=user
 password=password'
   PUBLICATION testpub;
```

> **Note**
>
> Especifique una contraseña distinta de la que se muestra aquí como práctica recomendada de seguridad.

Una vez creada la suscripción, se crea una ranura de replicación lógica en el publicador.

d. Para comprobar en este ejemplo que se replican los datos iniciales en el suscriptor, use la siguiente instrucción SQL en la base de datos de suscriptor.

```
SELECT count(*) FROM LogicalReplicationTest;
```

Todo cambio adicional en el publicador se replicará en el suscriptor.

La replicación lógica afecta al rendimiento. Le recomendamos que desactive la replicación lógica una vez finalizadas las tareas de replicación.

Ejemplo: replicación lógica mediante Aurora PostgreSQL y AWS Database Migration Service

Puede usar el AWS Database Migration Service (AWS DMS) para replicar una base de datos o parte de una base de datos. Utilice AWS DMS para migrar sus datos de una base de datos de Aurora PostgreSQL a otra base de datos comercial o de código abierto. Para obtener más información acerca de AWS DMS, consulte la Guía del usuario de AWS Database Migration Service.

En el siguiente ejemplo, se muestra cómo configurar la reproducción lógica desde una base de datos de Aurora PostgreSQL como publicador y, luego, cómo usar AWS DMS para la migración. El

publicador y suscriptor usados en este ejemplo son los mismos que los creados en Ejemplo: uso de la replicación lógica con clústeres de base de datos de Aurora PostgreSQL.

Para configurar la reproducción lógica con AWS DMS, necesita detalles acerca de su publicador y suscriptor de Amazon RDS. Concretamente, necesita detalles acerca de la instancia de base de datos de escritor del publicador y la instancia de base de datos de suscriptor.

Obtenga la siguiente información para la instancia de base de datos de escritor del publicador:

- Identificador de la nube virtual privada (VPC)
- Grupo de subredes
- Zona de disponibilidad (AZ)
- Grupo de seguridad de VPC
- ID de instancia de base de datos

Obtenga la siguiente información para la instancia de base de datos de suscriptor:

- ID de instancia de base de datos
- Motor de origen

Para utilizar AWS DMS para la reproducción lógica con Aurora PostgreSQL

1. Prepare la base de datos de publicador para trabajar con AWS DMS.

   Para ello, tanto PostgreSQL 10.x como bases de datos posteriores requieren que aplique funciones contenedoras de AWS DMS a la base de datos de publicador. Para obtener detalles acerca de este paso y pasos posteriores, consulte las instrucciones en Uso de PostgreSQL versión 10.x y posterior como origen para AWS DMS en la guía del usuario de AWS Database Migration Service.
2. Inicie sesión en la Consola de administración de AWS y abra la consola de AWS DMS en https://console.aws.amazon.com/dms/v2. En la parte superior derecha, elija la misma región de AWS en la que se encuentran el publicador y el suscriptor.
3. Cree una instancia de replicación de AWS DMS.

   Elija valores que sean los mismos que para la instancia de base de datos de escritor de su publicador. Entre estos se incluyen los siguientes:

- En VPC, elija la misma VPC que para la instancia de base de datos de escritor.
- En Replication Subnet Group (Grupo de subredes de replicación), elija un grupo de subredes con los mismos valores que para la instancia de base de datos de escritor. Cree uno nuevo si es necesario.
- En la Availability zone (Zona de disponibilidad), elija la misma zona que para la instancia de base de datos de escritor.
- En el VPC Security Group (Grupo de seguridad de VPC), elija el mismo grupo que para la instancia de base de datos de escritor.

4. Cree un punto de enlace de AWS DMS para el origen.

   Especifique el publicador como punto de enlace de origen mediante los siguientes valores:

   - En Endpoint type (Tipo de punto de enlace), elija Source endpoint (Punto de enlace de origen).
   - Elija Select RDS DB Instance (Seleccionar instancia de base de datos de RDS).
   - En RDS Instance (Instancia RDS), elija el identificador de base de datos de la instancia de base de datos de escritor del publicador.
   - En Source engine (Motor de origen), elija postgres.

5. Cree un punto de enlace de AWS DMS para el destino.

   Especifique el suscriptor como punto de enlace de destino mediante los siguientes valores:

   - En Endpoint type (Tipo de punto de enlace), elija Target endpoint (Punto de enlace de destino).
   - Elija Select RDS DB Instance (Seleccionar instancia de base de datos de RDS).
   - En RDS Instance (Instancia RDS), elija el identificador de base de datos de la instancia de base de datos de suscriptor.
   - Elija un valor para Source engine (Motor de origen). Por ejemplo, si el suscriptor es una base de datos PostgreSQL de RDS, elija postgres. Si el suscriptor es una base de datos Aurora PostgreSQL, elija aurora-postgresql.

6. Cree una tarea de migración de bases de datos de AWS DMS.

   Debe usar una tarea de migración de bases de datos para especificar qué tablas de base de datos se van a migrar, asignar los datos mediante un esquema de destino y crear tablas

nuevas en la base de datos de destino. Como mínimo, use los siguientes valores para Task configuration (Configuración de tareas):

- En Replication instance (Instancia de replicación), elija la instancia de replicación que creó en un paso anterior.
- En Source database endpoint (Punto de enlace de base de datos de origen), elija el origen del publicador que creó en un paso anterior.
- En Target database endpoint (Punto de enlace de base de datos de destino), elija el destino del suscriptor que creó en un paso anterior.

El resto de los detalles de la tarea dependen de su proyecto de migración. Para obtener más información acerca de la especificación de todos los detalles de las tareas de DMS, consulte Trabajo con las tareas DMS de AWS en la guía del usuario de AWS Database Migration Service.

Una vez que AWS DMS cree la tarea, comenzará a migrar datos del publicador al suscriptor.

Configuración de la autenticación de IAM para conexiones de replicación lógica

A partir de las versiones 11 y posteriores de Aurora PostgreSQL, puede usar la autenticación AWS Identity and Access Management (IAM) para las conexiones de replicación. Esta característica mejora la seguridad al permitirle administrar el acceso a la base de datos mediante roles de IAM en lugar de contraseñas. Funciona por clúster y sigue el mismo modelo de seguridad que la autenticación de IAM estándar.

La autenticación de IAM para conexiones de replicación es una característica opcional. Para habilitarla, establezca el parámetro `rds.iam_auth_for_replication` en `1` en el grupo de parámetros del clúster de la base de datos. Como se trata de un parámetro dinámico, el clúster de base de datos no necesita reiniciarse, lo que le permite aprovechar la autenticación de IAM con las cargas de trabajo existentes sin tiempo de inactividad. Antes de habilitar esta característica, asegúrese de que cumple los Requisitos previos mostrados a continuación.

Requisitos previos

Para utilizar la autenticación de IAM para conexiones de replicación, debe cumplir todos los siguientes requisitos:

- El clúster de base de datos de Aurora PostgreSQL debe ser la versión 11 o posterior.

- En el clúster de base de datos de Aurora PostgreSQL del publicador:
  - Habilite la autenticación de bases de datos de IAM.

    Para obtener más información, consulte Activación y desactivación de la autenticación de bases de datos de IAM.
  - Habilite la replicación lógica estableciendo el parámetro `rds.logical_replication` en `1`.

    Para obtener más información, consulte Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL.

  En la replicación lógica, el publicador es el clúster de base de datos de Aurora PostgreSQL de origen que envía los datos a los clústeres de suscriptores. Para obtener más información, consulte Información general de la replicación lógica de PostgreSQL con Aurora.

> **Note**
>
> La autenticación de IAM y la replicación lógica deben estar habilitadas en el clúster de base de datos de Aurora PostgreSQL del publicador. Si ninguna de las dos está habilitada, no podrá utilizar la autenticación de IAM para conexiones de replicación.

Habilitación de la autenticación de IAM para las conexiones de replicación

Realice los siguientes pasos antes de habilitar la autenticación de IAM para la conexión de replicación.

1. Compruebe que el clúster de base de datos de Aurora PostgreSQL cumpla todos los requisitos previos para la autenticación de IAM con conexiones de replicación. Para obtener más información, consulte Requisitos previos.
2. Configure el parámetro `rds.iam_auth_for_replication` modificando el grupo de parámetros de clúster de base de datos:
   - Establezca el parámetro `rds.iam_auth_for_replication` como `1`. Este parámetro dinámico no requiere un reinicio.
3. Conéctese a la base de datos y conceda los roles necesarios al usuario de replicación:

   Los siguientes comandos SQL otorgan los roles necesarios para habilitar la autenticación de IAM para las conexiones de replicación:

```
-- Grant IAM authentication role
GRANT rds_iam TO replication_user_name;
-- Grant replication privileges
ALTER USER replication_user_name WITH REPLICATION;
```

Tras completar estos pasos, el usuario especificado debe usar la autenticación de IAM para las conexiones de replicación.

> **Important**
>
> Al habilitar la característica, los usuarios con los roles `rds_iam` y `rds_replication` deben usar la autenticación de IAM para las conexiones de replicación. Esto se aplica si los roles se asignan directamente al usuario o si se heredan a través de otros roles.

Desactivación de la autenticación de IAM para las conexiones de replicación

Puede desactivar la autenticación de IAM para conexiones de replicación mediante cualquiera de los siguientes métodos:

- Establecimiento del parámetro `rds.iam_auth_for_replication` en `0` en el grupo de parámetro de clúster de base de datos
- Otra opción, puede desactivar cualquiera de estas características en el clúster de base de datos de Aurora PostgreSQL:
  - Desactivación de la replicación lógica estableciendo el parámetro `rds.logical_replication` en `0`
  - Desactivación de la autenticación de IAM

Al desactivar la característica, las conexiones de replicación pueden usar contraseñas de bases de datos para la autenticación si están configuradas.

> **Note**
>
> Las conexiones de replicación para los usuarios sin el rol `rds_iam` pueden usar la autenticación por contraseña incluso cuando la característica está habilitada.

Limitaciones y consideraciones

Las limitaciones y consideraciones siguientes se aplican cuando se use la autenticación de IAM para las conexiones de replicación.

- La autenticación de IAM para las conexiones de replicación solo está disponible para las versiones 11 y posteriores de Aurora PostgreSQL.
- El publicador debe admitir la autenticación de IAM para las conexiones de replicación.
- De forma predeterminada, el token de autenticación de IAM caduca a los 15 minutos. Es posible que tenga que actualizar las conexiones de replicación de larga duración antes de que caduque el token.

Reenvío de escritura local en Aurora PostgreSQL

El reenvío de escritura local (en el clúster) permite a sus aplicaciones emitir transacciones de lectura/escritura directamente en una réplica de Aurora. A continuación, los comandos de escritura se reenvían a la instancia de base de datos del escritor para su confirmación. Puede utilizar el reenvío de escritura local para las aplicaciones que tengan escrituras ocasionales y requieran coherencia de lectura después de escritura, que es la capacidad de leer la última escritura de una transacción.

Sin el reenvío de escritura, sus aplicaciones deben dividir completamente todo el tráfico de lectura y escritura, manteniendo dos conjuntos de conexiones a bases de datos para enviar el tráfico al punto de conexión adecuado. Las réplicas de lectura reciben actualizaciones de la instancia del escritor de forma asincrónica. Además, dado que el retraso de replicación puede variar entre las réplicas de lectura, es difícil lograr una coherencia de lectura global en todas las réplicas. Debe realizar transacciones de cualquier lectura que requiera coherencia de lectura después de escritura en la instancia de base de datos del escritor. O bien, tendría que desarrollar una lógica de aplicación personalizada y compleja para aprovechar numerosas réplicas de lectura para la escalabilidad mientras se garantiza la coherencia.

Con el reenvío de escritura evita la necesidad de dividir esas transacciones o enviarlas exclusivamente a la instancia del escritor. Tampoco es necesario desarrollar una lógica de aplicación compleja para lograr una coherencia en la lectura después de la escritura.

El reenvío de escritura local está disponible en todas las regiones en las que Aurora PostgreSQL está disponible. Se admite en las siguientes versiones de Aurora PostgreSQL:

- Versión 16.4 y otras versiones 16 superiores

- Versión 15.8 y otras versiones 15 superiores
- Versión 14.13 y otras versiones 14 superiores

El reenvío de escritura local se utiliza para reenviar escrituras desde réplicas de la región. Para reenviar las escrituras desde una réplica global, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

Temas

- Limitaciones y consideraciones del reenvío de escritura local en Aurora PostgreSQL
- Configuración de Aurora PostgreSQL para el reenvío de escritura local
- Procedimiento del reenvío de escritura local en Aurora PostgreSQL
- Supervisión del reenvío de escritura local en Aurora PostgreSQL

Limitaciones y consideraciones del reenvío de escritura local en Aurora PostgreSQL

Las limitaciones siguientes se aplican actualmente al reenvío de escritura local en Aurora PostgreSQL:

- RDS Proxy no admite el reenvío de escritura local.
- Algunas instrucciones no están permitidas o pueden producir resultados obsoletos cuando se utilizan en Aurora PostgreSQL con reenvío de escritura. Además, no se admiten funciones ni procedimientos definidos por el usuario. Por ello, la configuración `EnableLocalWriteForwarding` está desactivada de forma predeterminada para los clústeres de base de datos. Antes de activarla, asegúrese de que el código de la aplicación no se vea afectado por ninguna de estas restricciones.
- Los siguientes tipos de instrucciones SQL no son compatibles con el reenvío de escritura:

> **Note**
>
> Puede utilizar estas declaraciones de forma implícita en su aplicación o inferirlas mediante el protocolo PostgreSQL. Por ejemplo, la gestión de excepciones de PL/SQL puede provocar el uso de SAVEPOINT, que no es una declaración admitida.

- ANALYZE
- CLUSTER
- COPY
- Cursores: los cursores no son compatibles, así que debe asegurarse de cerrarlos antes de utilizar el reenvío de escritura local.
- Instrucciones en lenguaje de definición de datos (DDL)
- GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL
- LISTEN / NOTIFY
- LOCK
- SAVEPOINT
- SELECT INTO
- SET CONSTRAINTS
- Actualizaciones de secuencias: nextval() y setval()
- TRUNCATE
- Comandos de confirmación en dos fases: PREPARE TRANSACTION, COMMIT PREPARED y ROLLBACK PREPARED
- Funciones y procedimientos definidos por el usuario.
- VACUUM

Puede considerar el uso de las siguientes instrucciones SQL con reenvío de escritura:

- Una instrucción DML puede constar de varias partes, como una instrucción INSERT ... SELECT o una instrucción DELETE ... WHERE. En este caso, la instrucción completa se reenvía a la instancia de base de datos del escritor y se ejecuta allí.
- Instrucciones de lenguaje de manipulación de datos (DML) como INSERT, DELETE y UPDATE.
- Instrucciones EXPLAIN con las instrucciones de esta lista.
- Instrucciones PREPARE y EXECUTE.
- Instrucciones SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }

Configuración de Aurora PostgreSQL para el reenvío de escritura local

En las siguientes secciones, puede habilitar el reenvío de escritura local para su clúster de base de datos PostgreSQL de Amazon Aurora, configurar los niveles de consistencia y administrar las transacciones con el reenvío de escritura.

Habilitación del reenvío de escritura local

De forma predeterminada, el reenvío de escritura local no está habilitado para los clústeres de base de datos de Aurora PostgreSQL. El reenvío de escritura local se habilita a nivel de clúster, no a nivel de instancia.

Consola

Si usa la Consola de administración de AWS, seleccione la casilla de verificación Activar el reenvío de escritura loca debajo de Reenvío de escritura de réplicas de lectura al crear o modificar un clúster de base de datos.

AWS CLI

Para habilitar el reenvío de escritura local con la AWS CLI, utilice la opción `--enable-local-write-forwarding`. Esta opción funciona cuando crea un nuevo clúster de basede datos mediante el comando `create-db-cluster`. También funciona cuando modifica un clúster de base de datos existente mediante el comando `modify-db-cluster`. Puede deshabilitar el reenvío de escritura local mediante la opción `--no-enable-local-write-forwarding` con estos mismos comandos de la CLI.

En el siguiente ejemplo se crea un clúster de base de datos de Aurora PostgreSQL con el reenvío de escritura local habilitado.

```
aws rds create-db-cluster \
--db-cluster-identifier write-forwarding-test-cluster \
--enable-local-write-forwarding \
--engine aurora-postgresql \
--engine-version 16.4 \
--master-username myuser \
--master-user-password mypassword \
--backup-retention 1
```

A continuación, crea instancias de base de datos del escritor y lector para poder utilizar el reenvío de escritura. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

API de RDS

Para habilitar el reenvío de escritura local mediante la API de Amazon RDS, establezca el parámetro `EnableLocalWriteForwarding` en `true`. Este parámetro funciona cuando crea un nuevo clúster de base de datos mediante la operación `CreateDBCluster`. También funciona cuando modifica un clúster de base de datos existente mediante la operación `ModifyDBCluster`. Puede deshabilitar el reenvío de escritura local estableciendo el parámetro `EnableLocalWriteForwarding` en `false`.

Habilitación del reenvío de escritura local para las sesiones de bases de datos

El parámetro `apg_write_forward.consistency_mode` es un parámetro de base de datos y un parámetro de clúster de base de datos que permite el reenvío de escritura. Puede especificar `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF` para el nivel de coherencia de lectura. Para obtener más información sobre los niveles de consistencia, consulte Coherencia y aislamiento del reenvío de escritura local en Aurora PostgreSQL.

Las siguientes reglas se aplican a este parámetro:

- El valor predeterminado es `SESSION`.
- El reenvío de escritura local solo está disponible si establece `apg_write_forward.consistency_mode` en `EVENTUAL`, `SESSION` o `GLOBAL`. Este parámetro solo es pertinente en instancias del lector de clústeres de bases de datos que tengan habilitado el reenvío de escritura local.
- Si se establece el valor en `OFF`, se deshabilita el reenvío de escritura local en la sesión.

Coherencia y aislamiento del reenvío de escritura local en Aurora PostgreSQL

Puede controlar cuál es el grado de coherencia de lectura en una réplica de lectura. Puede ajustar el nivel de coherencia de lectura para asegurarse de que todas las operaciones de escritura reenviadas desde la sesión estén visibles en la réplica de lectura antes de cualquier consulta posterior. También puede utilizar esta configuración para asegurarse de que las consultas de la réplica de lectura siempre vean las actualizaciones más recientes de la instancia de base de datos del escritor. Esto es así incluso para los presentados por otros periodos de sesiones u otros grupos temáticos. Para especificar este tipo de comportamiento para la aplicación, elija el valor adecuado para el parámetro de nivel de sesión `apg_write_forward.consistency_mode`. El parámetro

`apg_write_forward.consistency_mode` solo tiene efecto en réplicas de lectura donde está habilitado el reenvío de escritura local.

> **Note**
>
> Para el parámetro `apg_write_forward.consistency_mode`, puede especificar los valores `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. De forma predeterminada, el valor se establece en `SESSION`. Si se establece este valor en `OFF`, se deshabilita el reenvío de escritura.

A medida que aumenta el nivel de coherencia, la aplicación pasa más tiempo esperando que los cambios se propaguen a las réplicas de lectura. Puede buscar el equilibrio entre una latencia más baja y asegurarse de que los cambios realizados en otras ubicaciones estén completamente disponibles antes de que se ejecuten las consultas.

Con cada configuración del modo de coherencia disponible, el efecto es el siguiente:

- `SESSION`: una sesión en una réplica de lectura que utiliza el reenvío de escritura verá los resultados de todos los cambios realizados en esa sesión. Los cambios son visibles independientemente de si la transacción está confirmada. Si es necesario, la consulta espera a que los resultados de las operaciones de escritura reenviadas se repliquen en la instancia de base de datos del lector actual. No espera a que se actualicen los resultados de las operaciones de escritura realizadas en otras sesiones dentro del clúster de base de datos actual.
- `EVENTUAL`: una sesión en una réplica de lectura que utiliza el reenvío de escritura puede ver datos ligeramente obsoletos debido al retardo de replicación. Los resultados de las operaciones de escritura de la misma sesión no están visibles hasta que la operación de escritura se realice en la instancia de base de datos del escritor y se repliquen en la réplica de lectura. La consulta no espera a que los resultados actualizados estén disponibles. Por lo tanto, podría recuperar los datos antiguos o los datos actualizados, en función del momento de las instrucciones y la cantidad de retardo de replicación.
- `GLOBAL`: una sesión de una réplica de lectura puede ver los cambios realizados por esa sesión. También verá todos los cambios confirmados tanto de la instancia de base de datos del escritor de como de otras réplicas de lectura. Cada consulta puede esperar un tiempo, que variará en función de la cantidad de retardo de la sesión. La consulta continúa cuando la réplica de lectura está actualizada con todos los datos confirmados de la instancia de base de datos del escritor, a partir del momento en que comenzó la consulta.

> **Note**
>
> El modo de coherencia global afecta a la latencia de las consultas ejecutadas en una sesión. Esperará incluso cuando la sesión no haya enviado ninguna consulta de escritura.

- `OFF`: el reenvío de escritura local está deshabilitado.

En las sesiones que utilizan reenvío de escritura, solo puede utilizar los niveles de aislamiento `REPEATABLE READ` y `READ COMMITTED`. Sin embargo, no se admite el nivel de `SERIALIZABLE` aislamiento.

Para obtener más información sobre todos los parámetros relacionados con el reenvío de escritura, consulte Configuración de parámetros predeterminada para el reenvío de escritura.

Modos de acceso a transacciones con reenvío de escritura

Si el modo de acceso a la transacción está configurado en solo lectura, no se utiliza el reenvío de escritura local. Solo puede establecer el modo de acceso en lectura/escritura cuando esté conectado a un clúster de base de datos y a una sesión que tenga habilitado el reenvío de escritura local.

Para obtener más información sobre los modos de acceso a las transacciones, consulte SET TRANSACTION.

Procedimiento del reenvío de escritura local en Aurora PostgreSQL

En las siguientes secciones, puede comprobar si un clúster de base de datos tiene habilitado el reenvío de escritura local, ver las consideraciones de compatibilidad y ver los parámetros configurables y la configuración de la autenticación. Esta información le proporciona los detalles necesarios para utilizar la característica de reenvío de escritura local de Aurora PostgreSQL de forma eficaz.

> **Note**
>
> Cuando se reinicia una instancia del escritor de un clúster que utiliza el reenvío de escritura local, todas las transacciones y consultas activas reenviadas en las instancias del lector que utilizan el reenvío de escritura local se cierran automáticamente. Cuando la instancia del escritor vuelva a estar disponible, podrá volver a intentar estas transacciones.

Comprobación de si un clúster de base de datos tiene habilitado el reenvío de escritura local

Para determinar si puede utilizar el reenvío de escritura local en un clúster de base de datos, confirme que el clúster tenga el atributo `LocalWriteForwardingStatus` establecido en `enabled`.

En la Consola de administración de AWS, en la pestaña Configuración de la página de detalles del clúster, verá el estado Habilitado para Reenvío de escritura de réplica de lectura local.

Para ver el estado de la configuración de reenvío de escritura local de todos los clústeres, ejecute el siguiente comando de la AWS CLI.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
{
"LocalWriteForwardingStatus": "enabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-1"
},
{
"LocalWriteForwardingStatus": "disabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-2"
},
{
"LocalWriteForwardingStatus": "requested",
"DBClusterIdentifier": "test-global-cluster-2"
},
{
"LocalWriteForwardingStatus": "null",
"DBClusterIdentifier": "aurora-postgresql-v2-cluster"
}
]
```

Un clúster de base de datos puede tener los siguientes valores para `LocalWriteForwardingStatus`:

- `disabled`: el reenvío de escritura local está deshabilitado.
- `disabling`: el reenvío de escritura local está en proceso de deshabilitación.

- `enabled`: el reenvío de escritura local está habilitado.
- `enabling`: el reenvío de escritura local está en proceso de habilitación.
- `null`: el reenvío de escritura local no está disponible para este clúster de base de datos.
- `requested`: se ha solicitado el reenvío de escritura local, pero aún no está activo.

Configuración de parámetros predeterminada para el reenvío de escritura

Los grupos de parámetros del clúster de Aurora contienen nuevos ajustes para la característica de reenvío de escritura local. Como se trata de parámetros de clúster, todas las instancias de base de datos de cada clúster tienen los mismos valores para estas variables. Los detalles sobre estos parámetros se resumen en la tabla siguiente, con notas de uso después de la tabla.

Parámetro	Alcance	Tipo	Valor predeterminado	Valores válidos
`apg_write_forward.connect_timeout`	Sesión	segundos	30	0–2147483647
`apg_write_forward.consistency_mode`	Session	enum	Sesión	`SESSION`, `EVENTUAL`, `GLOBAL`, y `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Sesión	milisegundos	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Sesión	milisegundos	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Global	int	25	1–100

El parámetro `apg_write_forward.max_forwarding_connections_percent` es el límite superior en slots de conexiones de base de datos que se puede utilizar para gestionar las consultas reenviadas desde los lectores. Se expresa como un porcentaje de la

configuración `max_connections` de la instancia de base de datos del escritor. Por ejemplo, si `max_connections` es `800` y `apg_write_forward.max_forwarding_connections_percent` es `10`, el escritor permite un máximo de 80 sesiones reenviadas simultáneas. Estas conexiones provienen del mismo grupo de conexiones administrado por la configuración `max_connections`. Esta configuración solo se aplica a la instancia de base de datos del escritor cuando el clúster tiene habilitado el reenvío de escritura local.

Utilice la siguiente configuración para controlar las solicitudes de reenvío de escritura local:

- `apg_write_forward.consistency_mode`: un parámetro de nivel de sesión que controla el grado de coherencia de lectura en una réplica de lectura. Los valores válidos son `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. De forma predeterminada, el valor se establece en `SESSION`. Si se establece el valor en `OFF`, se deshabilita el reenvío de escritura local en la sesión. Para obtener más información sobre los niveles de consistencia, consulte Coherencia y aislamiento del reenvío de escritura local en Aurora PostgreSQL. Este parámetro solo es pertinente en instancias del lector que tengan habilitado el reenvío de escritura local.
- `apg_write_forward.connect_timeout`: el número máximo de segundos que espera la réplica de lectura al establecer una conexión con la instancia de base de datos del escritor antes de desistir. Un valor de `0` significa esperar indefinidamente.
- `apg_write_forward.idle_in_transaction_session_timeout`: el número de milisegundos que la instancia de base de datos del escritor espera a que se produzca actividad en una conexión que se reenvía desde una réplica de lectura que tiene una transacción abierta antes de cerrarla. Si la sesión permanece inactiva en la transacción al finalizar este período, Aurora la termina. Un valor de `0` deshabilita el tiempo de espera.
- `apg_write_forward.idle_session_timeout`: el número de milisegundos que la instancia de base de datos del escritor espera a que se produzca actividad en una conexión que se reenvía desde una réplica de lectura antes de cerrarla. Si la sesión permanece inactiva al finalizar este período, Aurora la termina. Un valor de `0` desactiva el tiempo de espera.

rdswriteforwarduser

`rdswriteforwarduser` es un usuario que utilizaremos para establecer una conexión entre la réplica de lectura y la instancia de base de datos del escritor.

> **Note**
>
> `rdswriteforwarduser` hereda sus privilegios CONNECT en las bases de datos de los clientes mediante el rol PUBLIC. Si se revocan los privilegios del rol PUBLIC, tendrá que conceder los privilegios CONNECT para las bases de datos a las que deba reenviar las escrituras.

Supervisión del reenvío de escritura local en Aurora PostgreSQL

En las siguientes secciones, puede supervisar el reenvío de escritura local en los clústeres de Aurora PostgreSQL, incluidas las métricas pertinentes de CloudWatch y los eventos de espera para realizar un seguimiento del rendimiento e identificar posibles problemas.

Métricas de Amazon CloudWatch y variables de estado de Aurora MySQL para el reenvío de escritura

Las siguientes métricas de Amazon CloudWatch se aplican a las instancias de base de datos del escritor cuando se utiliza el reenvío de escritura en una o más réplicas de lectura.

Métrica de CloudWatch	Unidades y descripción
`AuroraLocal ForwardingWriterDML Throughput`	Recuento por segundo Número de instrucciones DML reenviadas procesadas cada segundo por esta instancia de base de datos de escritor.
`AuroraLocalForwardingWriter OpenSessions`	Recuento Número de sesiones abiertas en esta instancia de base de datos de escritor que procesa las consultas reenviadas.
`AuroraLocal ForwardingWriterTotalSessions`	Recuento Número total de sesiones reenviadas en esta instancia de base de datos de escritor.

Las siguientes métricas de CloudWatch se aplican a cada réplica de lectura. Estas métricas se miden en cada instancia de base de datos del lector de un clúster de base de datos con el reenvío de escritura local habilitado.

Métrica de CloudWatch	Unidad y descripción
`AuroraForwardingReplicaCommitThroughput`	Recuento por segundo Número de confirmaciones en las sesiones reenviadas por esta réplica cada segundo.
`AuroraForwardingReplicaDMLLatency`	Milisegundos. Tiempo medio de respuesta en milisegundos de los DML reenviados durante la réplica.
`AuroraForwardingReplicaDMLThroughput`	Recuento por segundo Número de instrucciones DML reenviadas procesadas en esta réplica por segundo.
`AuroraForwardingReplicaErrorSessionsLimit`	Recuento Número de sesiones rechazadas por la instancia de base de datos del escritor porque se ha alcanzado el límite máximo de conexiones o el límite máximo de conexiones de reenvío de escritura.
`AuroraForwardingReplicaOpenSessions`	Recuento Número de sesiones que utilizan el reenvío de escritura en una instancia de réplica local.
`AuroraForwardingReplicaReadWaitLatency`	Milisegundos. Tiempo de espera medio en milisegundos que la réplica espera para ser coherente con el LSN de la instancia de base de datos de escritura. El grado en que la instancia de base de datos de lector espera depende de la configuración `apg_write_forward.consistency_mode`. Para obtener información sobre esta configuración, consulte the section called “Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL”.

Eventos de espera para el reenvío de escritura local en Aurora PostgreSQL

Amazon Aurora genera los siguientes eventos de espera cuando utiliza el reenvío de escritura con Aurora PostgreSQL.

Temas

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

El evento `IPC:AuroraWriteForwardConnect` se produce cuando un proceso de backend de la réplica de lectura espera a que se abra una conexión con la instancia de base de datos del escritor.

Causas probables del aumento del tiempo de espera

Este evento aumenta a medida que se incrementa el número de intentos de conexión desde una réplica de lectura en el nodo del escritor.

Acciones

Reduzca el número de conexiones simultáneas desde una réplica de lectura en el nodo del escritor.

IPC:AuroraWriteForwardConsistencyPoint

El evento `IPC:AuroraWriteForwardConsistencyPoint` describe cuánto tiempo esperará una consulta de un nodo en la réplica de lectura para que los resultados de las operaciones de escritura reenviadas se repliquen en la región actual. Este evento solo se genera si el parámetro de nivel de sesión `apg_write_forward.consistency_mode` se establece en uno de los siguientes valores:

- `SESSION`: las consultas de una réplica de lectura esperan los resultados de todos los cambios realizados en esa sesión.

- `GLOBAL`: las consultas de una réplica de lectura esperan los resultados de los cambios realizados en esa sesión, además de todos los cambios confirmados tanto de la instancia de base de datos del escritor como de la réplica de lectura.

Para obtener más información sobre la configuración del parámetro `apg_write_forward.consistency_mode`, consulte the section called “Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL”.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- Aumento del retraso de réplica, medido por la métrica `ReplicaLag` de Amazon CloudWatch. Para obtener más información sobre esta métrica, consulte Monitoreo de replicación de Aurora PostgreSQL.
- Aumento de la carga en la instancia de base de datos del escritor o en la réplica de lectura.

Acciones

Cambie el modo de coherencia según los requisitos de su aplicación.

IPC:AuroraWriteForwardExecute

El evento `IPC:AuroraWriteForwardExecute` se produce cuando un proceso de backend de la réplica de lectura está esperando a que una consulta reenviada se complete y se obtengan los resultados del nodo del escritor del clúster de base de datos.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- Obtención de una gran cantidad de filas del nodo del escritor.
- El aumento de la latencia de la red entre el nodo del escritor y la réplica de lectura incrementa el tiempo que tarda la réplica de lectura en recibir datos del nodo del escritor.
- El aumento de la carga en la réplica de lectura puede retrasar la transmisión de la solicitud de consulta desde la réplica de lectura hasta el nodo del escritor.

- El aumento de la carga en el nodo del escritor puede retrasar la transmisión de los datos desde el nodo del escritor hasta la réplica de lectura.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

- Optimice las consultas para recuperar solo los datos necesarios.
- Optimice las operaciones de lenguaje de manipulación de datos (DML) para modificar únicamente los datos necesarios.
- Si la réplica de lectura o el nodo del escritor están limitados por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

El evento `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` se produce cuando un proceso de backend de la réplica de lectura que utiliza el modo de coherencia GLOBAL está esperando para obtener el punto de coherencia global del nodo del escritor antes de ejecutar una consulta.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre la réplica de lectura y el nodo del escritor incrementa el tiempo que tarda la réplica de lectura en recibir datos del nodo del escritor.
- El aumento de la carga en la réplica de lectura puede retrasar la transmisión de la solicitud de consulta desde la réplica de lectura hasta el nodo del escritor.
- El aumento de la carga en el nodo del escritor puede retrasar la transmisión de los datos desde el nodo del escritor hasta la réplica de lectura.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

- Cambie el modo de coherencia según los requisitos de su aplicación.

- Si la réplica de lectura o el nodo del escritor están limitados por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardXactAbort

El evento `IPC:AuroraWriteForwardXactAbort` se produce cuando un proceso de backend de la réplica de lectura está esperando el resultado de una consulta de limpieza remota. Las consultas de limpieza se emiten para devolver el proceso al estado correspondiente después de cancelar una transacción de reenvío de escritura. Amazon Aurora las ejecuta porque ha detectado un error o porque un usuario ha emitido un comando `ABORT` explícito o ha cancelado una consulta en ejecución.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre la réplica de lectura y el nodo del escritor incrementa el tiempo que tarda la réplica de lectura en recibir datos del nodo del escritor.
- El aumento de la carga en la réplica de lectura puede retrasar la transmisión de la solicitud de consulta de limpieza desde la réplica de lectura al nodo del escritor.
- El aumento de la carga en el nodo del escritor puede retrasar la transmisión de los datos desde el nodo del escritor hasta la réplica de lectura.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

- Investigue por qué se ha cancelado la transacción.
- Si la réplica de lectura o la instancia de base de datos del escritor están limitadas por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardXactCommit

El evento `IPC:AuroraWriteForwardXactCommit` se produce cuando un proceso de backend de la réplica de lectura está esperando el resultado de un comando de transacción de confirmación reenviado.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre la réplica de lectura y el nodo del escritor incrementa el tiempo que tarda la réplica de lectura en recibir datos del nodo del escritor.
- El aumento de la carga en la réplica de lectura puede retrasar la transmisión de la solicitud de consulta desde la réplica de lectura hasta el nodo del escritor.
- El aumento de la carga en el nodo del escritor puede retrasar la transmisión de los datos desde el nodo del escritor hasta la réplica de lectura.

Acciones

Si la réplica de lectura o el nodo del escritor están limitados por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardXactStart

El evento `IPC:AuroraWriteForwardXactStart` se produce cuando un proceso de backend de la réplica de lectura está esperando el resultado de un comando de transacción de inicio reenviado.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre la réplica de lectura y el nodo del escritor incrementa el tiempo que tarda la réplica de lectura en recibir datos del nodo del escritor.
- El aumento de la carga en la réplica de lectura puede retrasar la transmisión de la solicitud de consulta desde la réplica de lectura hasta el nodo del escritor.
- El aumento de la carga en el nodo del escritor puede retrasar la transmisión de los datos desde el nodo del escritor hasta la réplica de lectura.

Acciones

Si la réplica de lectura o el nodo del escritor están limitados por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

Uso de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock

Puede usar un clúster de base de datos de Aurora PostgreSQL como base de conocimiento de Amazon Bedrock. Para obtener más información, consulte Creación de un almacén vectorial en Amazon Aurora. Una base de conocimiento toma automáticamente los datos de texto no estructurados almacenados en un bucket de Amazon S3, los convierte en fragmentos de texto y vectores y los almacena en una base de datos de PostgreSQL. Con las aplicaciones de IA generativa, puede utilizar Agentes para Amazon Bedrock para consultar los datos almacenados en la base de conocimientos y utilizar los resultados de esas consultas para aumentar las respuestas que proporcionan los modelos fundacionales. Este flujo de trabajo se denomina generación aumentada de recuperación (RAG). Para obtener más información sobre la RAG, consulte Generación aumentada de recuperación (RAG).

Para obtener información detallada sobre el uso de Aurora PostgreSQL para crear aplicaciones de IA generativa mediante RAG, consulte esta entrada de blog.

Temas

- Requisitos previos
- Preparación de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock
- Creación de una base de conocimiento en la consola de Bedrock
- Creación rápida de una base de conocimiento de Amazon Bedrock para Aurora PostgreSQL

Requisitos previos

Familiarícese con los siguientes requisitos previos para utilizar el clúster de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock. A un nivel superior, debe configurar los siguientes servicios para utilizarlos con Bedrock:

- Clúster de base de datos de Amazon Aurora PostgreSQL creado en cualquiera de las siguientes versiones:

- Versión 16.1 y todas las versiones posteriores
- Versión 15.4 y posteriores
- Versión 14.9 y posteriores
- Versión 13.12 y posteriores
- Versión 12.16 y posteriores

> **Note**
>
> Debe habilitar la extensión `pgvector` en la base de datos de destino y usar la versión 0.5.0 o superior. Para obtener más información, consulte pgvector v0.5.0 con indexación HNSW.

- API de datos de RDS
- Un usuario administrado en AWS Secrets Manager. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.

Preparación de Aurora PostgreSQL como base de conocimientos para Amazon Bedrock

Siga los pasos que se explican en las secciones siguientes a fin de preparar Aurora PostgreSQL para usarla como base de conocimiento de Amazon Bedrock.

Creación y configuración de Aurora PostgreSQL

Para configurar Amazon Bedrock con un clúster de base de datos de Aurora PostgreSQL, primero debe crear un clúster de base de datos de Aurora PostgreSQL y tomar nota de los campos importantes para la configuración con Amazon Bedrock. Para obtener más información sobre la creación de un clúster de base de datos de Aurora PostgreSQL, consulte Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él.

- Habilite la API de datos al crear el clúster de base de datos de Aurora PostgreSQL. Para obtener más información acerca de las versiones admitidas, consulte Uso de la API de datos de Amazon RDS.
- Anote los nombres de recurso de Amazon (ARN) del clúster de base de datos de Aurora PostgreSQL. Los necesitará para configurar el clúster de base de datos para el uso con Amazon Bedrock. Para obtener más información, consulte Nombres de recurso de Amazon (ARN).

Conexión a una base de datos e instalación de pgvector

Puede conectarse a Aurora PostgreSQL mediante cualquiera de las utilidades de conexión. Para obtener información más detallada sobre estas utilidades, consulte Conexión a un clúster de base de datos Amazon Aurora PostgreSQL. También puede usar el editor de consultas de la consola de RDS para ejecutar las consultas. Para utilizar el editor de consultas, necesita un clúster de base de datos de Aurora con la API de datos de RDS habilitada.

1. Inicie sesión en la base de datos con su usuario maestro y configure pgvector. Use el siguiente comando si la extensión no está instalada:

   ```
   CREATE EXTENSION IF NOT EXISTS vector;
   ```

   Utilice la `pgvector`0.5.0 y versiones superiores que admitan la indexación HNSW. Para obtener más información, consulte pgvector v0.5.0 con indexación HNSW.

2. Utilice el siguiente comando para comprobar qué versión de `pg_vector` está instalada:

   ```
   SELECT extversion FROM pg_extension WHERE extname='vector';
   ```

Configuración de objetos y privilegios de base de datos

1. Cree un esquema específico que Bedrock pueda usar para consultar los datos. Utilice el siguiente comando para crear un esquema:

   ```
   CREATE SCHEMA bedrock_integration;
   ```

2. Cree un nuevo rol que Bedrock pueda usar para consultar la base de datos. Utilice el siguiente comando para crear un nuevo rol:

   ```
   CREATE ROLE bedrock_user WITH PASSWORD 'password' LOGIN;
   ```

   > **Note**
   >
   > Anote esta contraseña, ya que la usará más tarde para crear una contraseña de Secrets Manager.

Si utiliza el cliente `psql`, use los siguientes comandos para crear un nuevo rol:

```
CREATE ROLE bedrock_user LOGIN;
\PASSWORD password;
```

3. Otorgue los permisos `bedrock_user` para administrar el esquema de `bedrock_integration`. Esto permitirá crear tablas o índices dentro del esquema.

```
GRANT ALL ON SCHEMA bedrock_integration to bedrock_user;
```

4. Inicie sesión como `bedrock_user` y cree una tabla en `bedrock_integration schema`.

```
CREATE TABLE bedrock_integration.bedrock_kb (id uuid PRIMARY KEY, embedding
 vector(n), chunks text, metadata json, custom_metadata jsonb);
```

Este comando creará la tabla `bedrock_kb` en el esquema `bedrock_integration` con integraciones de Titan.

Sustituya la n en el tipo de datos `vector(n)` por la dimensión adecuada para el modelo de incrustación que esté utilizando. Utilice las siguientes recomendaciones para seleccionar sus dimensiones:

- Para el modelo Titan versión 2, utilice `vector(1024)`, `vector(512)` o `vector (256)`. Para obtener más información, consulte Amazon Titan Embeddings Text.
- Para el modelo Titan versión 1.2, utilice `vector(1536)`. Para obtener más información, consulte Amazon Titan Multimodal Embeddings G1.
- Para el modelo Cohere Embed, utilice `vector(1024)`. Para obtener más información, consulte Cohere Embed models.
- Para Cohere Embed Multilingual versión 3, utilice `vector(1024)`.

Las cuatro primeras columnas son obligatorias. Para el manejo de los metadatos, Bedrock escribe los datos de los archivos de metadatos en la columna `custom_metadata`. Le recomendamos la creación de esta columna si planea usar metadatos y filtros. Si no crea una columna `custom_metadata`, agregue columnas individuales para cada atributo de metadatos de la tabla antes de iniciar la ingesta. Para obtener más información, consulte Configuración y personalización de las consultas y la generación de respuestas.

5. Siga estos pasos para crear los índices necesarios que Bedrock utiliza para consultar los datos:

   - Cree un índice con el operador de coseno que Bedrock pueda utilizar para consultar los datos.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops);
     ```

   - Le recomendamos que establezca el valor de `ef_construction` en 256 para la versión 0.6.0 o superior de `pgvector` que utilice la creación de índices en paralelo.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops) WITH (ef_construction=256);
     ```

   - Cree un índice que Bedrock pueda usar para consultar los datos de texto.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (to_tsvector('simple',
      chunks));
     ```

   - Si ha creado una columna para metadatos personalizados, cree un índice que Bedrock pueda utilizar para consultar los metadatos.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (custom_metadata);
     ```

Crear un secreto en Secrets Manager

Secrets Manager le permite almacenar sus credenciales de Aurora para que se puedan transmitir de forma segura a las aplicaciones. Si no ha elegido la opción de administrador de AWS Secrets Manager al crear el clúster de base de datos de Aurora PostgreSQL, puede crear un secreto ahora. Para obtener más información sobre cómo crear un secreto de base de datos AWS Secrets Manager, consulte AWS Secrets Manager database secret.

Creación de una base de conocimiento en la consola de Bedrock

Mientras prepara Aurora PostgreSQL para usarlo como almacén vectorial para una base de conocimiento, debe recopilar los siguientes detalles que debe proporcionar a la consola de Amazon Bedrock.

- ARN de clúster de base de datos de Amazon Aurora: el ARN del clúster de base de datos.
- ARN secreto: el ARN de la clave de AWS Secrets Manager para su clúster de base de datos.

- Nombre de base de datos: el nombre de la base de datos. Por ejemplo, puede usar el *postgres* de base de datos predeterminado.
- Nombre de la tabla: le recomendamos que proporcione un nombre cualificado del esquema al crear la tabla mediante un comando similar al siguiente.

```
CREATE TABLE bedrock_integration.bedrock_kb;
```

  Este comando creará la tabla `bedrock_kb` en el esquema `bedrock_integration`.
- Cuando cree la tabla, configúrela con las columnas y los tipos de datos especificados. Puede usar los nombres de columna que prefiera en lugar de los que aparecen en la tabla. Recuerde que debe anotar los nombres que haya elegido, ya que tendrá que consultarlos durante la configuración de la base de conocimiento.

Nombre de la columna	Tipo de datos:	Descripción
id	UUID clave principal	Contiene identificadores únicos para cada registro.
fragmentos	Texto	Contiene los fragmentos de texto sin procesar de los orígenes de datos.
Incrustación	Vector	Contiene las incrustaciones vectoriales de los orígenes de datos.
metadatos	JSON	Contiene los metadatos necesarios para llevar a cabo la atribución del origen y para permitir la ingesta y consulta de datos.
custom_metadata	JSONB	(Opcional) Define la columna de destino en la que Amazon Bedrock escribe los detalles de los metadatos de los orígenes de datos.

Con estos detalles, ya puede crear una base de conocimiento en la consola de Bedrock. Para obtener más información detallada sobre la configuración de un índice vectorial y la creación de una base de conocimiento, consulte Create a vector store in Amazon Aurora y Create a vector store in Amazon Aurora.

Tras añadir Aurora como base de conocimiento, ya puede incorporar sus orígenes de datos para realizar búsquedas y consultas. Para obtener más información, consulte Ingest your data sources into the Knowledge Base.

Creación rápida de una base de conocimiento de Amazon Bedrock para Aurora PostgreSQL

El flujo de trabajo de generación aumentada por recuperación (RAG) de Amazon Bedrock se basa en datos vectoriales almacenados en una base de datos de Aurora PostgreSQL para impulsar la recuperación de contenido. Anteriormente, configurar Aurora PostgreSQL como almacén de datos vectoriales para las bases de conocimiento de Bedrock era un proceso de varios pasos que requería numerosas acciones manuales en diferentes interfaces de usuario. Esto dificultaba que los científicos de datos y los desarrolladores pudieran aprovechar Aurora para sus proyectos de Bedrock.

Para mejorar la experiencia del usuario, AWS ha creado una nueva opción de creación rápida basada en CloudFormation que simplifica el proceso de configuración. La creación rápida de Aurora le permite aprovisionar un clúster de base de datos de Aurora PostgreSQL configurado previamente como almacén vectorial para sus bases de conocimiento de Amazon Bedrock con un solo clic.

Temas

- Regiones compatibles y versiones de Aurora PostgreSQL
- Comprensión del proceso de creación rápida
- Beneficios del uso de la creación rápida de Aurora
- Limitaciones del proceso de creación rápida de Aurora

Regiones compatibles y versiones de Aurora PostgreSQL

La opción de creación rápida de Aurora está disponible en todas las regiones de AWS que admiten las bases de conocimiento de Amazon Bedrock. De forma predeterminada, crea un clúster de base de datos de Aurora PostgreSQL con la versión 15.7. Para obtener más información sobre las regiones compatibles, consulte Supported models and regions for Amazon Bedrock Knowledge Bases.

Comprensión del proceso de creación rápida

El proceso de creación rápida aprovisiona automáticamente los siguientes recursos para configurar una base de datos de Amazon Aurora PostgreSQL como almacén de datos vectoriales para su base de conocimiento de Bedrock:

Un clúster de base de datos de Aurora PostgreSQL en su cuenta, configurado con los ajustes predeterminados.

- Las ACU (unidades de capacidad de Aurora) están configuradas entre 0 y 16. Esto permite que su almacén vectorial se reduzca verticalmente hasta cero cuando no se use, lo que ahorra costos de computación. Las ACU se pueden ajustar posteriormente en la consola de Amazon RDS.
- Índice HNSW (mundo pequeño navegable jerárquico) que utiliza la distancia euclidiana como medida de similitud para las incrustaciones vectoriales de Bedrock almacenadas en Aurora.
- La instancia de base de datos es una instancia de versión 2 sin servidor.
- El clúster está asociado a la VPC y las subredes predeterminadas, y tiene habilitada la API de datos de RDS.
- AWS Secrets Manager administra las credenciales de administrador del clúster.

Además de los ajustes predeterminados, se configuran los siguientes ajustes para usted. A medida que avance en el proceso, verá pantallas que explican el flujo de trabajo.

- Alimentar el clúster de Aurora con los objetos de base de datos necesarios:
  - Cree la extensión, el esquema, el rol y las tablas de pgvector necesarios para la base de conocimiento de Bedrock.
  - Registre un usuario de base de datos con privilegios limitados para que Bedrock interactúe con el clúster.
- Aparecerá un banner de progreso durante todo el proceso de aprovisionamiento de recursos, que le permitirá realizar un seguimiento del estado de los siguientes subeventos:
  - Creación de clústeres de Aurora
  - Alimentación del clúster de Aurora
  - Creación de bases de conocimiento

  El banner permanece visible hasta que la base de conocimiento se haya creado por completo, incluso si sale y vuelve a la página.

- Puede hacer clic en `View details` en el banner de progreso para ver el estado de cada paso. Para obtener más información sobre los eventos durante la creación de la base de conocimiento, elija el enlace de CloudFormation en la pantalla de visualización de detalles. Cuando se haya completado el proceso, podrá empezar a utilizarse la nueva base de conocimiento de Bedrock.
- Los ID de pila de todos los recursos de creación rápida se encuentran en las etiquetas de la base de conocimiento de Bedrock, por si necesita consultarlos.

Una base de conocimiento de Bedrock, con la configuración del clúster de Aurora recién aprovisionado al crear el almacén de vectores.

Beneficios del uso de la creación rápida de Aurora

- El proceso de creación rápida basado en CloudFormation reduce considerablemente el tiempo y la complejidad que supone utilizar Aurora como almacén de vectores.
- Aurora ofrece un rendimiento excelente, escalabilidad vectorial y ventajas económicas, además de permitir escalar hasta cero cargos de computación cuando no se usa.
- El proceso de creación rápida agiliza la experiencia integral, lo que le permite crear y configurar fácilmente sus bases de conocimiento de Bedrock con Aurora.
- Los clientes pueden crear una plantilla de CloudFormation para personalizar el aprovisionamiento con sus propias configuraciones.

Limitaciones del proceso de creación rápida de Aurora

- Con la opción de creación rápida de Aurora, el clúster de base de datos se aprovisiona con las configuraciones predeterminadas. Sin embargo, es posible que esta configuración predeterminada no cumpla con sus requisitos específicos o su caso de uso previsto. La creación rápida no ofrece opciones para modificar las configuraciones durante el proceso de aprovisionamiento. Las configuraciones se definen automáticamente para agilizar la experiencia de implementación. Si necesita personalizar la configuración del clúster de base de datos de Aurora, puede hacerlo después de la implementación inicial mediante la creación rápida en la consola de Amazon RDS.
- Si bien el flujo de creación rápida simplifica el proceso de configuración, el tiempo necesario para crear el clúster de base de datos de Aurora sigue siendo de aproximadamente diez minutos, igual que en una implementación manual. Esto se debe al tiempo necesario para aprovisionar la infraestructura de Aurora.

- La opción de creación rápida está diseñada para experimentar y realizar una configuración rápida. Es posible que los recursos creados mediante la creación rápida no sean adecuados para su uso en producción y no pueda migrarlos directamente a un entorno de producción en su VPC.

Integración de Amazon Aurora PostgreSQL con otros servicios de AWS

Amazon Aurora se integra con otros servicios de AWS con el fin de permitirle ampliar su clúster de base de datos de Aurora PostgreSQL para usar funcionalidades adicionales en la nube de AWS. El clúster de base de datos de Aurora PostgreSQL puede usar los servicios de AWS para hacer lo siguiente:

- Recopilar, ver y evaluar rápidamente el rendimiento de las instancias de base de datos Aurora PostgreSQL con Performance Insights de Amazon RDS. Performance Insights amplía las características de monitorización existentes de Amazon RDS para ilustrar el desempeño de la base de datos y le ayuda a analizar cualquier problema que le afecte. Con el panel de Performance Insights, puede visualizar la carga de la base de datos y filtrarla por esperas, instrucciones SQL, hosts o usuarios. Para obtener más información acerca de Performance Insights, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.
- Puede configurar un clúster de base de datos de Aurora PostgreSQL para publicar datos de registro en Amazon CloudWatch Logs. CloudWatch Logs ofrece un almacenamiento de larga duración para sus registros. Con CloudWatch Logs, puede realizar análisis en tiempo real de los datos de registro y utilizar CloudWatch para crear alarmas y ver métricas. Para obtener más información, consulte Publicación de registros de Aurora PostgreSQL en Amazon CloudWatch Logs.
- Importe datos de un bucket de Amazon S3 a un clúster de base de datos de Aurora PostgreSQL o exporte los datos de un clúster de base de datos de Aurora PostgreSQL a un bucket de Amazon S3. Para obtener más información, consulte Importación de datos de Amazon S3 en un clúster de base de datos Aurora PostgreSQL y Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3.
- Agregue predicciones basadas en machine learning a las aplicaciones de base de datos mediante el lenguaje SQL. El machine learning de Aurora es una integración sumamente optimizada entre los servicios de base de datos de Aurora y el machine learning (ML) de AWS: IA de SageMaker

y Amazon Comprehend. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora PostgreSQL.

- Invoque funciones de AWS Lambda desde un clúster de base de datos de Aurora PostgreSQL. Para ello, utilice la extensión de PostgreSQL `aws_lambda` proporcionada con Aurora PostgreSQL. Para obtener más información, consulte Invocación de una función de AWS Lambda desde un clúster de base de datos de Aurora PostgreSQL.
- Integre consultas de Amazon Redshift y Aurora PostgreSQL. Para obtener más información, consulte Introducción al uso de consultas federadas en PostgreSQL en la Guía para desarrolladores de bases de datos Amazon Redshift.

Importación de datos de Amazon S3 en un clúster de base de datos Aurora PostgreSQL

Puede importar los datos que se hayan almacenado mediante Amazon Simple Storage Service a una tabla en una instancia de clúster de base de datos Aurora PostgreSQL. Para ello, primero debe instalar la extensión de Aurora PostgreSQL `aws_s3`. Esta extensión proporciona las funciones que se utilizan para importar datos de un bucket de Amazon S3. Un bucket es un contenedor de objetos o archivos de Amazon S3. Los datos pueden estar en un archivo de valores separados por comas (CSV), un archivo de texto o un archivo comprimido (gzip). A continuación, aprenderá a instalar la extensión y a importar datos de Amazon S3 en una tabla.

Para hacer la importación de Simple Storage Service (Amazon S3) hacia , la base de datos debe ejecutar la versión de PostgreSQL 10.7 o superior. Aurora PostgreSQL.

Si no tiene datos almacenados en Amazon S3, primero debe crear un bucket y almacenar los datos. Para obtener más información, consulte los siguientes temas en la guía del usuario de Amazon Simple Storage Service.

- Crear un bucket
- Añadir un objeto a un bucket.

Se admite la importación entre cuentas desde Amazon S3. Para obtener más información, consulte Concesión de permisos entre cuentas en la Guía del usuario de Amazon Simple Storage Service.

Puede utilizar la clave administrada por el cliente para el cifrado al importar datos desde S3. Para obtener más información, consulte Claves de KMS almacenadas en AWS KMS en la Guía del usuario de Amazon Simple Storage Service.

> **Note**
>
> La importación de datos desde Amazon S3 no se admite para Aurora Serverless v1. Se admite para Aurora Serverless v2.

Temas

- Instalación de la extensión aws_s3
- Información general sobre la importación de datos desde los datos de Amazon S3
- Configuración del acceso a un bucket de Amazon S3
- Importación de datos de Amazon S3 a un clúster de base de datos Aurora PostgreSQL
- Referencia de funciones

Instalación de la extensión aws_s3

Antes de poder usar Amazon S3 con su clúster de base de datos de Aurora PostgreSQL, , debe instalar la extensión `aws_s3`. Esta extensión proporciona funciones para importar datos desde Amazon S3. También proporciona funciones para exportar datos desde una instancia de un clúster de base de datos de Aurora PostgreSQL a un bucket de Amazon S3. Para obtener más información, consulte Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3. La extensión `aws_s3` depende de algunas de las funciones de ayuda en la extensión de `aws_commons`, que se instala automáticamente cuando es necesario.

Para instalar la extensión de **`aws_s3`**

1. Utilice psql (o pgAdmin) para conectarse a la instancia de escritor del clúster de base de datos de Aurora PostgreSQL como usuario que tiene privilegios de `rds_superuser`. Si mantuvo el nombre predeterminado durante el proceso de configuración, conéctese como `postgres`.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Para instalar la extensión, ejecute el siguiente comando:

   ```
   postgres=> CREATE EXTENSION aws_s3 CASCADE;
   NOTICE: installing required extension "aws_commons"
   CREATE EXTENSION
   ```

3. Para comprobar que la extensión está instalada, puede usar el metacomando psql `\dx`.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Ya están disponibles las funciones para importar datos de Amazon S3 y para exportar datos a Amazon S3.

Información general sobre la importación de datos desde los datos de Amazon S3

Para importar datos de S3 a Aurora PostgreSQL, lleve a cabo el siguiente procedimiento:

Primero, reúna los detalles que necesita proporcionar a la función. Entre ellos se incluye el nombre de la tabla en la instancia del clúster de base de datos de Aurora PostgreSQL, y el nombre del bucket, la ruta del archivo, el tipo de archivo y la Región de AWS donde se almacenan los datos de Amazon S3. Para obtener más información, consulte el tema para ver un objeto en la guía del usuario de Amazon Simple Storage Service.

> **Note**
>
> Actualmente no se admite la importación de datos multiparte desde Amazon S3.

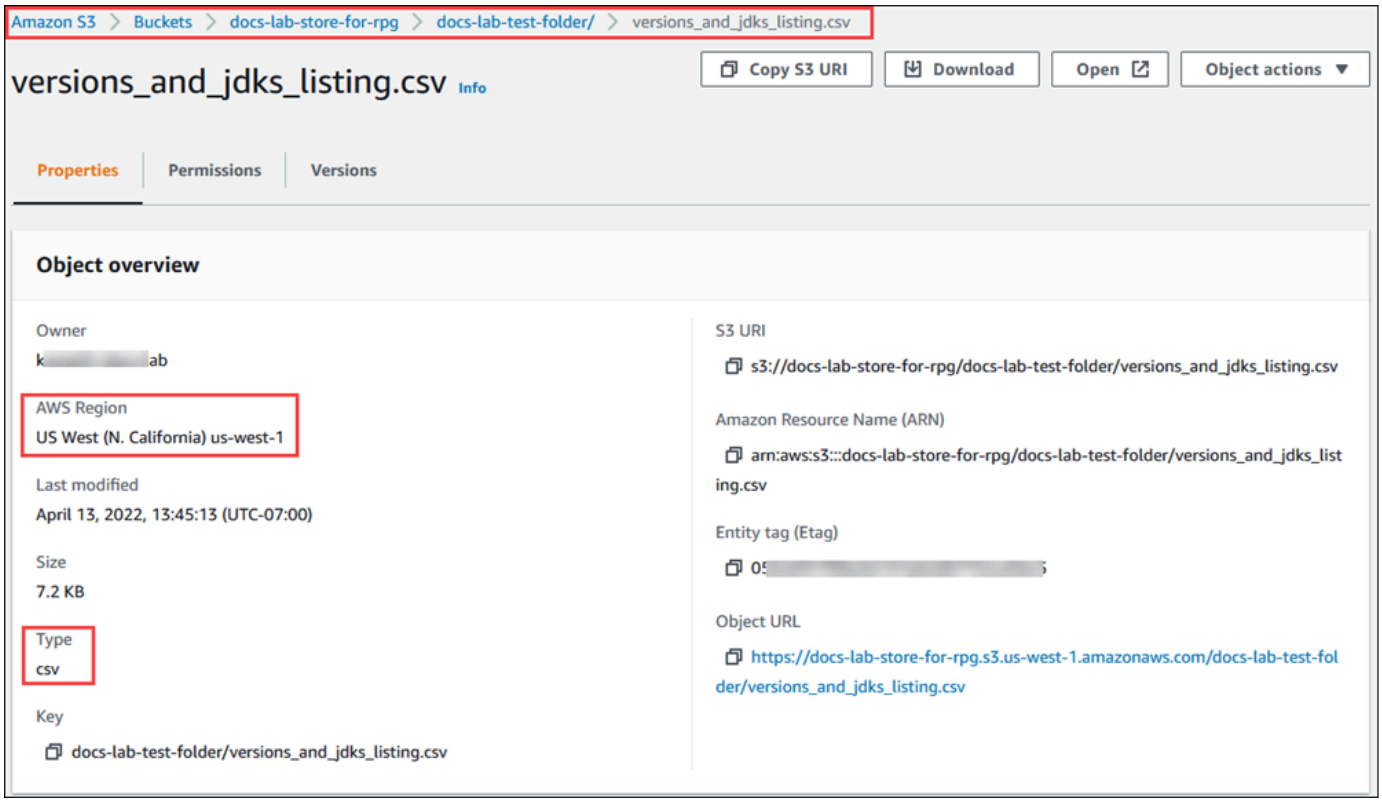
1. Obtenga el nombre de la tabla en la que la función `aws_s3.table_import_from_s3` va a importar los datos. A modo de ejemplo, el siguiente comando crea una tabla `t1` que se puede utilizar en pasos posteriores.

```
postgres=> CREATE TABLE t1
    (col1 varchar(80),
    col2 varchar(80),
    col3 varchar(80));
```

2. Obtenga información sobre el bucket de Amazon S3 y los datos que se van a importar. Para ello, abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/ y elija Buckets. Busque

el bucket que contiene sus datos en la lista. Elija el bucket, abra la página de información general de objetos y, a continuación, Properties (Propiedades).

Anote el nombre del bucket, la ruta, la Región de AWS y el tipo de archivo. Necesitará el nombre de recurso de Amazon (ARN) más adelante para configurar el acceso a Amazon S3 a través de un rol de IAM. Para obtener más información, consulte Configuración del acceso a un bucket de Amazon S3. En la siguiente imagen se muestra un ejemplo.



3. Para verificar la ruta a los datos en el bucket de Amazon S3, utilice el comando de AWS CLI `aws s3 cp`. Si la información es correcta, este comando descarga una copia del archivo de Amazon S3.

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. Configure los permisos de clúster de base de datos de Aurora PostgreSQL para permitir el acceso al archivo en el bucket de Amazon S3. Para ello, utilice un rol de AWS Identity and Access Management (IAM) o las credenciales de seguridad. Para obtener más información, consulte Configuración del acceso a un bucket de Amazon S3.
5. Proporcione la ruta y otros detalles del objeto de Amazon S3 recopilados (consulte el paso 2) para la función `create_s3_uri` para construir un objeto URI de Amazon S3. Para obtener

más información sobre esta función, consulte aws_commons.create_s3_uri. A continuación se muestra un ejemplo de cómo construir este objeto durante una sesión de psql.

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
   'us-west-1'
) AS s3_uri \gset
```

En el paso siguiente, pase este objeto (`aws_commons._s3_uri_1`) a la función `aws_s3.table_import_from_s3` para importar los datos a la tabla.

6. Invoque la función `aws_s3.table_import_from_s3` para importar los datos de Amazon S3 a la tabla. Para obtener información de referencia, consulte aws_s3.table_import_from_s3. Para ver ejemplos, consulta Importación de datos de Amazon S3 a un clúster de base de datos Aurora PostgreSQL.

Configuración del acceso a un bucket de Amazon S3

Para importar datos de un archivo de Amazon S3, conceda permiso del clúster de base de datos de Aurora PostgreSQL para obtener acceso al bucket de Amazon S3 en el que se encuentra el archivo. Puede proporcionar acceso a un bucket de Amazon S3 de una de las dos formas siguientes, tal y como se describe en los siguientes temas.

Temas

- Uso de un rol de IAM para obtener acceso a un bucket de Amazon S3
- Uso de credenciales de seguridad para obtener acceso a un bucket de Amazon S3
- Solución de errores de acceso a Amazon S3

Uso de un rol de IAM para obtener acceso a un bucket de Amazon S3

Antes de cargar los datos de un archivo de Amazon S3, conceda permiso al clúster de base de datos de Aurora PostgreSQL para obtener acceso al bucket de Amazon S3 en el que se encuentra el archivo. De esta forma, no tiene que facilitar ni administrar información adicional de credenciales en la llamada a la función aws_s3.table_import_from_s3.

Para ello, cree una política de IAM que proporcione acceso al bucket de Amazon S3. Cree un rol de IAM y conecte la política a dicho rol. A continuación, asigne el rol de IAM al clúster de base de datos.

> **Note**
>
> No se puede asociar un rol de IAM a un clúster de base de datos de Aurora Serverless v1, por lo que no se aplican los siguientes pasos.

Para dar a un clúster de base de datos de Aurora PostgreSQL acceso a Simple Storage Service (Amazon S3) a través de un rol de IAM, lleve a cabo el siguiente procedimiento:

1. Cree una política de IAM.

   Esta política concede los permisos de bucket y objeto que permiten que el clúster de base de datos de Aurora PostgreSQL tenga acceso a Amazon S3.

   Incluya las siguientes acciones requeridas en la política para permitir la transferencia de archivos de un bucket de Amazon S3 a Aurora PostgreSQL:

   - `s3:GetObject`
   - `s3:ListBucket`

   Incluya los siguientes recursos en la política para identificar el bucket de Amazon S3 y los objetos incluidos en este. A continuación se muestra el formato de nombre de recurso de Amazon (ARN) para obtener acceso a Amazon S3.

   - arn:aws:s3:::*amzn-s3-demo-bucket*
   - arn:aws:s3:::*amzn-s3-demo-bucket*/*

   Para obtener información adicional sobre cómo crear una política de IAM para Aurora PostgreSQL, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM. Consulte también el Tutorial: Crear y asociar su primera política administrada por el cliente en la Guía del usuario de IAM.

   El siguiente comando de la AWS CLI crea una política de IAM denominada `rds-s3-import-policy` con estas opciones. Otorga acceso a un bucket denominado *amzn-s3-demo-bucket*.

> **Note**
>
> Anote el Nombre de recurso de Amazon (ARN) de la política que devolvió este comando. Al asociar la política a un rol de IAM, se necesita el ARN para realizar un paso posterior.

Example

Para Linux, macOS o Unix:

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

Para Windows:

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
```

```
            "s3:ListBucket"
          ],
          "Effect": "Allow",
          "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
          ]
        }
      ]
    }'
```

2. Crear un rol de IAM.

   Haga esto para que Aurora PostgreSQL pueda asumir este rol de IAM para obtener acceso a los buckets de Amazon S3. Para obtener más información, vea Crear un rol para delegar permisos a un usuario de IAM en Guía del usuario de IAM.

   Le recomendamos que utilice las claves de contexto de condición globales de `aws:SourceArn` y `aws:SourceAccount` en las políticas basadas en recursos para limitar los permisos del servicio a un recurso específico. Esta es la forma más eficaz de protegerse contra el problema del suplente confuso.

   Si utiliza claves de contexto de condición globales y el valor `aws:SourceArn` contiene el ID de cuenta, el valor `aws:SourceAccount` y la cuenta en el valor `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utiliza en la misma instrucción de política.

   - Use `aws:SourceArn` si quiere acceso entre servicios para un único recurso.
   - Use `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

   En la política, asegúrese de utilizar la clave de contexto de condición global `aws:SourceArn` con el ARN completo del recurso. En el siguiente ejemplo se muestra cómo se usa el comando de la AWS CLI para crear un rol denominado `rds-s3-import-role`.

   Example

   Para Linux, macOS o Unix:

```
aws iam create-role \
   --role-name rds-s3-import-role \
```

```
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
             }
       }
     ]
   }'
```

Para Windows:

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
             }
       }
     ]
   }'
```

3. Asocie la política de IAM que creó al rol de IAM creado.

   El siguiente comando AWS CLI adjunta la política creada en el paso anterior al rol denominado `rds-s3-import-role`. Sustituya *`your-policy-arn`* por el ARN de la política que ha anotado en un paso anterior.

   Example

   Para Linux, macOS o Unix:

   ```
   aws iam attach-role-policy \
      --policy-arn your-policy-arn \
      --role-name rds-s3-import-role
   ```

   Para Windows:

   ```
   aws iam attach-role-policy ^
      --policy-arn your-policy-arn ^
      --role-name rds-s3-import-role
   ```

4. Añada el rol de IAM al clúster de base de datos.

   Para ello, utilice la Consola de administración de AWS o la AWS CLI, tal y como se describe a continuación.

Consola

Para añadir un rol de IAM para un clúster de base de datos de PostgreSQL utilizando la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione el nombre de clúster de base de datos de PostgreSQL para mostrar sus detalles.
3. En la pestaña Connectivity & security (Conectividad y seguridad), en la sección Manage IAM roles (Administrar roles de IAM), elija el rol que desee agregar en la instancia Add IAM roles to this clúster .
4. En Feature Feature (Característica), elija s3Import.
5. Seleccione Add role (Añadir rol).

AWS CLI

Para añadir un rol de IAM para un clúster de base de datos de PostgreSQL mediante la CLI, realice el siguiente procedimiento:

- Utilice el siguiente comando para añadir el rol al clúster de base de datos de PostgreSQL denominado `my-db-cluster`. Sustituya *`your-role-arn`* por el ARN del rol que ha anotado en el paso anterior. Utilice `s3Import` para el valor de la opción `--feature-name`.

  Example

  Para Linux, macOS o Unix:

  ```
  aws rds add-role-to-db-cluster \
     --db-cluster-identifier my-db-cluster \
     --feature-name s3Import \
     --role-arn your-role-arn   \
     --region your-region
  ```

  Para Windows:

  ```
  aws rds add-role-to-db-cluster ^
     --db-cluster-identifier my-db-cluster ^
     --feature-name s3Import ^
     --role-arn your-role-arn ^
     --region your-region
  ```

API de RDS

Para agregar un rol de IAM para un clúster de base de datos de PostgreSQL mediante la API de Amazon RDS, llame a la operación AddRoleToDBCluster .

Uso de credenciales de seguridad para obtener acceso a un bucket de Amazon S3

Si lo prefiere, puede utilizar credenciales de seguridad para proporcionar acceso a un bucket de Amazon S3, en lugar de proporcionar acceso con un rol de IAM. Para ello, especifique el parámetro `credentials` en la llamada a la función aws_s3.table_import_from_s3.

El parámetro `credentials` es una estructura de tipo `aws_commons._aws_credentials_1`, que contiene credenciales de AWS. Utilice la función aws_commons.create_aws_credentials

para establecer la clave de acceso y la clave secreta en una estructura `aws_commons._aws_credentials_1`, como se muestra a continuación.

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

Tras crear la estructura `aws_commons._aws_credentials_1` , utilice la función aws_s3.table_import_from_s3 con el parámetro `credentials` para importar los datos, tal y como se muestra a continuación.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

O bien puede incluir la llamada a la función aws_commons.create_aws_credentials insertada dentro de la llamada a la función `aws_s3.table_import_from_s3`.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Solución de errores de acceso a Amazon S3

Si tiene problemas de conexión al intentar importar los datos de Amazon S3, consulte las recomendaciones que se indican a continuación:

- Solución de problemas de identidades y accesos en Amazon Aurora
- Solución de problemas de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service
- Solución de problemas de Amazon S3 e IAM en la Guía del usuario de IAM

Importación de datos de Amazon S3 a un clúster de base de datos Aurora PostgreSQL

Para importar datos desde su bucket de Amazon S3, utilice la función `table_import_from_s3` de la extensión aws_s3. Para obtener información de referencia, consulte aws_s3.table_import_from_s3.

> **Note**
>
> En los siguientes ejemplos se utiliza el método de rol de IAM para permitir el acceso al bucket de Amazon S3. Por tanto, no hay parámetros de credenciales en las llamadas a la función `aws_s3.table_import_from_s3`.

A continuación se muestra un ejemplo típico.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

Los parámetros son los siguientes:

- `t1`: nombre de la tabla en el clúster de base de datos de PostgreSQL en la que desea copiar los datos.
- `''`: lista opcional de columnas en la tabla de la base de datos. Puede utilizar este parámetro para indicar qué columnas de los datos de S3 van en las columnas de la tabla. Si no se especifica ninguna columna, se copian en la tabla todas las columnas. Para obtener un ejemplo de uso de una lista de columnas, consulte Importación de un archivo de Amazon S3 que utiliza un delimitador personalizado.
- `(format csv)`: argumentos de COPY de PostgreSQL. El proceso de copia utiliza los argumentos y el formato del comando COPY de PostgreSQL para importar los datos. Las opciones de formato incluyen un valor separado por comas (CSV), como se muestra en este ejemplo, texto y binario. El valor predeterminado es texto.
- `s3_uri`: una estructura que contiene la información que identifica el archivo de Amazon S3. Para ver un ejemplo de cómo utilizar la función aws_commons.create_s3_uri para crear una estructura `s3_uri`, consulte Información general sobre la importación de datos desde los datos de Amazon S3.

Para obtener más información acerca de esta función, consulte aws_s3.table_import_from_s3.

La función `aws_s3.table_import_from_s3` devuelve texto. Para especificar otros tipos de archivos que se van a importar desde un bucket de Amazon S3, consulte uno de los siguientes ejemplos.

> **Note**
>
> Si importa 0 bytes, se producirá un error.

Temas

- Importación de un archivo de Amazon S3 que utiliza un delimitador personalizado
- Importación de un archivo comprimido (gzip) de Amazon S3
- Importación de un archivo de Amazon S3 codificado

Importación de un archivo de Amazon S3 que utiliza un delimitador personalizado

En el siguiente ejemplo se muestra cómo importar un archivo que utiliza un delimitador personalizado. También se muestra cómo controlar dónde colocar los datos en la tabla de la base de datos usando el parámetro `column_list` de la función aws_s3.table_import_from_s3.

En este ejemplo, supongamos que la siguiente información está organizada en columnas delimitadas por barras verticales en el archivo de Amazon S3.

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
4|foo4|bar4|elephant4
...
```

Para importar un archivo que utiliza un delimitador personalizado

1. Cree una tabla en la base de datos para los datos importados.

   ```
   postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
   ```

2. Utilice el siguiente formulario de la función aws_s3.table_import_from_s3 para importar datos desde el archivo de Amazon S3.

Puede incluir la llamada a la función aws_commons.create_s3_uri insertada dentro de la llamada a la función `aws_s3.table_import_from_s3` para especificar el archivo.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   'test',
   'a,b,d,e',
   'DELIMITER ''|''',
   aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
east-2')
);
```

Los datos se encuentran ahora en la tabla en las siguientes columnas.

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Importación de un archivo comprimido (gzip) de Amazon S3

El siguiente ejemplo muestra cómo importar un archivo comprimido con gzip desde Amazon S3. El archivo que se importa debe tener los siguientes metadatos de Simple Storage Service (Amazon S3):

- Clave: `Content-Encoding`
- Valor: `gzip`

Si carga el archivo con la Consola de administración de AWS, el sistema suele aplicar los metadatos. Para obtener información sobre cómo cargar archivos en Simple Storage Service (Amazon S3) con la Consola de administración de AWS, la AWS CLI o la API, consulte Carga de objetos en la Guía del usuario de Amazon Simple Storage Service.

Para obtener más información acerca de los metadatos de Simple Storage Service (Amazon S3) y detalles acerca de los metadatos proporcionados por el sistema, consulte Edición de metadatos de objeto en la consola de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

Importe el archivo gzip en su clúster de base de datos Aurora PostgreSQL como se muestra a continuación.

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

Importación de un archivo de Amazon S3 codificado

El siguiente ejemplo muestra cómo importar un archivo desde Amazon S3 que tenga codificación Windows-1252.

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

Referencia de funciones

Funciones

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri
- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

Importa datos de Amazon S3 en una tabla Aurora PostgreSQL. La extensión `aws_s3` proporciona la función `aws_s3.table_import_from_s3`. El valor de devolución es texto.

Sintaxis

Los parámetros obligatorios son `table_name`, `column_list` y `options`. Estos identifican la tabla de la base de datos y especifican cómo se copian los datos en la tabla.

Asimismo, puede utilizar los siguientes parámetros:

- El parámetro `s3_info` especifica el archivo Amazon S3 que se va a importar. Cuando utilice este parámetro, se proporciona acceso a Amazon S3 mediante un rol de IAM para el clúster de base de datos de PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1
)
```

- El parámetro `credentials` especifica las credenciales para acceder a Amazon S3. Cuando utilice este parámetro, no utilice un rol de IAM.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

Parámetros

table_name

Cadena de texto obligatoria que contiene el nombre de la tabla de la base de datos de PostgreSQL a la que importar los datos.

column_list

Cadena de texto obligatoria que contiene una lista opcional de las columnas de la tabla de la base de datos de PostgreSQL en la que se copiarán los datos. Si la cadena está vacía, se utilizan todas las columnas de la tabla. Para ver un ejemplo, consulte Importación de un archivo de Amazon S3 que utiliza un delimitador personalizado.

options

Cadena de texto obligatoria que contiene argumentos para el comando `COPY` de PostgreSQL. Estos argumentos especifican cómo se copian los datos en la tabla PostgreSQL. Para obtener más detalles, consulte la documentación de COPY de PostgreSQL.

s3_info

Tipo compuesto `aws_commons._s3_uri_1` que contiene la siguiente información sobre el objeto de S3:

- `bucket`: el nombre del bucket de Amazon S3 que contiene el archivo.
- `file_path` –: la ruta de Amazon S3 del archivo.
- `region`: la región de AWS en la que se encuentra el archivo. Para ver una lista de los nombres de regiones de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

credenciales

Tipo compuesto `aws_commons._aws_credentials_1` que contiene las siguientes credenciales para usar en la operación de importación:

- Clave de acceso
- Clave secreta
- Token de sesión

Para obtener información sobre la creación de una estructura compuesta `aws_commons._aws_credentials_1`, consulte aws_commons.create_aws_credentials.

Sintaxis alternativa

Como ayuda en las pruebas, puede utilizar un conjunto de parámetros expandido en lugar de los parámetros `s3_info` y `credentials`. A continuación, se incluyen variaciones de sintaxis adicionales para la función:`aws_s3.table_import_from_s3`

- En lugar de utilizar el parámetro `s3_info` para identificar un archivo de Amazon S3, utilice la combinación de los parámetros `bucket`, `file_path` y `region`. Con esta forma de la función, se facilita acceso a Amazon S3 mediante un rol de IAM en la instancia de base de datos de PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text
)
```

- En lugar de utilizar el parámetro `credentials` para especificar el acceso a Amazon S3, utilice la combinación de parámetros `access_key`, `session_key` y `session_token`.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text,
   access_key text,
   secret_key text,
   session_token text
)
```

Parámetros alternativos

bucket

Cadena de texto que incluye el nombre del bucket de Amazon S3 que contiene el archivo.

file_path

Cadena de texto que contiene la ruta de Amazon S3 del archivo.

region

Una cadena de texto que identifique la ubicación de Región de AWS del archivo. Para ver una lista de los nombres de Región de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

access_key

Cadena de texto que contiene la clave de acceso que se va a utilizar para la operación de importación. El valor predeterminado es NULL.

secret_key

Cadena de texto que contiene la clave secreta que se va a usar para la operación de importación. El valor predeterminado es NULL.

session_token

(Opcional) Cadena de texto que contiene la clave de la sesión que se va a utilizar para la operación de importación. El valor predeterminado es NULL.

aws_commons.create_s3_uri

Crea una estructura `aws_commons._s3_uri_1` para contener la información de archivos de Amazon S3. Utilice los resultados de la función `aws_commons.create_s3_uri` en el parámetro `s3_info` de la función aws_s3.table_import_from_s3.

Sintaxis

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parámetros

bucket

Cadena de texto obligatoria que contiene el nombre del bucket de Amazon S3 del archivo.

file_path

Cadena de texto requerida que contiene la ruta de Amazon S3 del archivo.

region

Cadena de texto obligatoria que contiene la Región de AWS en la que se encuentra el archivo. Para ver una lista de los nombres de Región de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

aws_commons.create_aws_credentials

Establece una clave de acceso y una clave secreta en una estructura `aws_commons._aws_credentials_1`. Utilice los resultados de la función `aws_commons.create_aws_credentials` en el parámetro `credentials` de la función aws_s3.table_import_from_s3.

Sintaxis

```
aws_commons.create_aws_credentials(
   access_key text,
   secret_key text,
   session_token text
```

```
)
```

Parámetros

access_key

Cadena de texto obligatoria que contiene la clave de acceso que se va a utilizar para importar un archivo de Amazon S3. El valor predeterminado es NULL.

secret_key

Cadena de texto obligatoria que contiene la clave secreta que se va a utilizar para importar un archivo de Amazon S3. El valor predeterminado es NULL.

session_token

Cadena de texto opcional que contiene el token de la sesión que se va a utilizar para importar un archivo de Amazon S3. El valor predeterminado es NULL. Si facilita un `session_token` opcional, puede usar credenciales temporales.

Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3

Puede consultar datos de una instancia de clúster de base de datos de Aurora PostgreSQL y exportarlos directamente a archivos almacenados en un bucket de Amazon S3. Para ello, primero debe instalar la extensión de Aurora PostgreSQL `aws_s3`. Esta extensión le proporciona las funciones que utiliza para exportar los resultados de las consultas a Amazon S3. A continuación, puede averiguar cómo instalar la extensión y cómo exportar datos de Amazon S3.

Solo se puede exportar desde una instancia de base de datos aprovisionada o de Aurora Serverless v2. Estos pasos no se admiten para Aurora Serverless v1.

> **Note**
>
> No se ha agregado compatibilidad con la exportación entre cuentas a Amazon S3.

Todas las versiones disponibles actualmente de Aurora PostgreSQL admiten la exportación de datos a Amazon Simple Storage Service. Para obtener información detallada sobre la versión, consulte las actualizaciones de Amazon Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

Si no tienes un bucket configurado para la exportación, consulta los siguientes temas: Guía del usuario de Amazon Simple Storage Service.

- Configuración de Amazon S3
- Crear un bucket

De forma predeterminada, los datos exportados desde Aurora PostgreSQL a Amazon S3 utilizan cifrado del servidor con Clave administrada de AWS. De forma alternativa, puede utilizar una clave administrada por el cliente que ya haya creado. Si utiliza cifrado de buckets, el bucket de Amazon S3 debe cifrarse con una clave AWS Key Management Service (AWS KMS) (SSE-KMS). En la actualidad, no se admiten buckets cifrados con claves administradas de Amazon S3 (SSE-S3).

> **Note**
>
> Puede guardar datos de instantáneas de base de datos y de base de datos en Amazon S3 mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS. Para obtener más información, consulte Exportación de datos de instantánea del clúster de bases de datos a Amazon S3.

Temas

- Instalación de la extensión aws_s3
- Información general de la exportación de datos a Amazon S3
- Especificación de la ruta del archivo de Amazon S3 a exportar
- Configuración del acceso a un bucket de Amazon S3
- Exportación de datos de consulta mediante la función aws_s3.query_export_to_s3
- Referencia de funciones
- Solución de errores de acceso a Amazon S3

Instalación de la extensión aws_s3

Antes de poder usar Amazon Simple Storage Service con su clúster de base de datos de Aurora PostgreSQL, , debe instalar la extensión `aws_s3`. Esta extensión proporciona funciones para exportar datos desde la instancia de escritura de un clúster de base de datos de Aurora PostgreSQL a un bucket de Amazon S3. También proporciona funciones para importar datos desde Amazon S3.

Para obtener más información, consulte Importación de datos de Amazon S3 en un clúster de base de datos Aurora PostgreSQL. La extensión `aws_s3` depende de algunas de las funciones de ayuda en la extensión de `aws_commons`, que se instala automáticamente cuando es necesario.

Para instalar la extensión de **`aws_s3`**

1. Utilice psql (o pgAdmin) para conectarse a la instancia de escritor del clúster de base de datos de Aurora PostgreSQL como usuario que tiene privilegios de `rds_superuser`. Si mantuvo el nombre predeterminado durante el proceso de configuración, conéctese como `postgres`.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Para instalar la extensión, ejecute el siguiente comando:

   ```
   postgres=> CREATE EXTENSION aws_s3 CASCADE;
   NOTICE: installing required extension "aws_commons"
   CREATE EXTENSION
   ```

3. Para comprobar que la extensión está instalada, puede usar el metacomando psql `\dx`.

   ```
   postgres=> \dx
          List of installed extensions
       Name     | Version |   Schema   |                 Description
   -------------+---------+------------+---------------------------------------------
    aws_commons | 1.2     | public     | Common data types across AWS services
    aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
    plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
   (3 rows)
   ```

Ya están disponibles las funciones para importar datos de Amazon S3 y para exportar datos a Amazon S3.

Confirme que su versión de Aurora PostgreSQL admite exportaciones a Amazon S3.

Puede comprobar que su versión de Aurora PostgreSQL admite la exportación a Amazon S3 mediante el comando `describe-db-engine-versions`. El siguiente ejemplo comprueba si la versión 10.14 se puede exportar a Amazon S3.

```
aws rds describe-db-engine-versions --region us-east-1 \
```

```
--engine aurora-postgresql --engine-version 10.14 | grep s3Export
```

Si en la salida se recoge la cadena de texto `"s3Export"`, el motor admite las exportaciones de Amazon S3 . Si no es así, el motor no las admite.

Información general de la exportación de datos a Amazon S3

Para exportar datos almacenados en una base de datos de Aurora PostgreSQL a un bucket de Amazon S3, utilice el siguiente procedimiento.

Para exportar Aurora PostgreSQL datos a S3

1. Identifique la ruta de archivo de Amazon S3 que se va a utilizar para exportar datos. Para obtener más información sobre este proceso, consulte Especificación de la ruta del archivo de Amazon S3 a exportar.
2. Conceda permiso para acceder al bucket de Amazon S3.

   Para exportar datos a un archivo de Amazon S3, conceda permiso al clúster de base de datos de Aurora PostgreSQL para obtener acceso al bucket de Amazon S3 que la exportación usará para el almacenamiento. Esto incluye los siguientes pasos:

   1. Cree una política de IAM que proporcione acceso al bucket de Amazon S3 al que se desea exportar.
   2. Cree un rol de IAM.
   3. Asocie la política que ha creado al rol que ha creado.
   4. Agregue este rol de IAM al clúster de base de datos.

   Para obtener más información sobre este proceso, consulte Configuración del acceso a un bucket de Amazon S3.
3. Identifique una consulta de base de datos para obtener los datos. Exporte los datos de consulta llamando a la función `aws_s3.query_export_to_s3`.

   Después de completar las tareas de preparación anteriores, utilice la función aws_s3.query_export_to_s3 para exportar los resultados de la consulta a Amazon S3. Para obtener más información sobre este proceso, consulte Exportación de datos de consulta mediante la función aws_s3.query_export_to_s3.

Especificación de la ruta del archivo de Amazon S3 a exportar

Especifique la siguiente información para identificar la ubicación de Amazon S3 a la que desea exportar los datos:

- Nombre de bucket: un bucket es un contenedor para objetos o archivos de Amazon S3.

  Para obtener más información sobre cómo almacenar datos con Amazon S3, consulte Crear un bucket y Trabajar con objetos en la Guía del usuario de Amazon Simple Storage Service.
- Ruta del archivo: la ruta del archivo identifica dónde se almacena la exportación en el bucket de Amazon S3. La ruta del archivo consta de lo siguiente:
  - Un prefijo de ruta opcional que identifica una ruta de carpeta virtual.
  - Un prefijo de archivo que identifica uno o varios archivos que se van a almacenar. Las exportaciones más grandes se almacenan en varios archivos, cada uno con un tamaño máximo de aproximadamente 6 GB. Los nombres de archivo adicionales tienen el mismo prefijo de archivo, pero con `_part`*XX* anexado. *XX* representa 2, luego 3, y así sucesivamente.

  Por ejemplo, una ruta de archivo con una carpeta `exports` y un prefijo de archivo `query-1-export` es `/exports/query-1-export`.
- Región de AWS (opcional): la región de AWS donde se encuentra el bucket de Amazon S3. Si no especifica un valor de región de AWS, Aurora guarda sus archivos en Amazon S3, en la misma región de AWS que el clúster de base de datos de exportación.

  > **Note**
  >
  > Actualmente, la región de AWS debe ser la misma región que la del clúster de base de datos e de exportación.

  Para ver una lista de los nombres de regiones de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

Para mantener la información del archivo de Amazon S3 acerca de dónde se va a almacenar la exportación, puede utilizar la función aws_commons.create_s3_uri para crear una estructura compuesta `aws_commons._s3_uri_1` de la siguiente manera.

```
psql=> SELECT aws_commons.create_s3_uri(
```

```
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Más adelante, proporcione este valor `s3_uri_1` como un parámetro en la llamada a la función aws_s3.query_export_to_s3. Para ver ejemplos, consulte Exportación de datos de consulta mediante la función aws_s3.query_export_to_s3.

Configuración del acceso a un bucket de Amazon S3

Para exportar datos a Amazon S3, conceda permiso al clúster de base de datos de PostgreSQL para acceder al bucket de Amazon S3 al que irán los archivos.

Para ello, siga el procedimiento que se indica a continuación.

Para proporcionar al clúster de base de datos de PostgreSQL acceso a Amazon S3 a través de un rol de IAM

1. Cree una política de IAM.

   Esta política concede los permisos de bucket y objeto que permiten al clúster de base de datos de PostgreSQL acceder a Amazon S3.

   Como parte de la creación de esta política, realice los siguientes pasos:

   a. Incluya las siguientes acciones necesarias en la política para permitir la transferencia de archivos del clúster de base de datos de PostgreSQL a un bucket de Amazon S3:

      - `s3:PutObject`
      - `s3:AbortMultipartUpload`

   b. Incluye el nombre de recurso de Amazon (ARN) que identifica el bucket de Amazon S3 y los objetos del bucket. El formato del ARN para acceder a Amazon S3 es: `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Para obtener información adicional sobre cómo crear una política de IAM para Aurora PostgreSQL, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM. Consulte también el Tutorial: Crear y asociar su primera política administrada por el cliente en la Guía del usuario de IAM.

El siguiente comando de la AWS CLI crea una política de IAM denominada `rds-s3-export-policy` con estas opciones. Otorga acceso a un bucket denominado *`amzn-s3-demo-bucket`*.

> **Warning**
>
> Le recomendamos que configure la base de datos en una VPC privada que tenga políticas de punto de enlace configuradas para acceder a buckets específicos. Para obtener más información, consulte Uso de políticas de punto de enlace para Amazon S3 en la Guía del usuario de Amazon VPC.
> Recomendamos encarecidamente que no cree una política con acceso a todos los recursos. Este acceso puede representar una amenaza para la seguridad de los datos. Si crea una política que da acceso `S3:PutObject` a todos los recursos mediante `"Resource":"*"`, un usuario con privilegios de exportación puede exportar datos a todos los buckets de su cuenta. Además, el usuario puede exportar datos a cualquier bucket en el que se pueda escribir públicamente dentro de su región de AWS.

Después de crear la política, anote el nombre de recurso de Amazon (ARN) de la política. Cuando asocia la política a un rol de IAM, necesita el ARN para realizar un paso posterior.

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
         "Action": [
           "s3:PutObject*",
           "s3:ListBucket",
           "s3:GetObject*",
           "s3:DeleteObject*",
           "s3:GetBucketLocation",
           "s3:AbortMultipartUpload"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
```

```
    }'
```

2. Cree un rol de IAM.

   Realiza este paso para que Aurora PostgreSQL pueda asumir este rol de IAM en su nombre para obtener acceso a los buckets de Amazon S3. Para obtener más información, consulte Creación de un rol para delegar permisos a un usuario de IAM en la Guía del usuario de IAM.

   Le recomendamos que utilice las claves de contexto de condición globales de `aws:SourceArn` y `aws:SourceAccount` en las políticas basadas en recursos para limitar los permisos del servicio a un recurso específico. Esta es la forma más eficaz de protegerse contra el problema del suplente confuso.

   Si utiliza claves de contexto de condición globales y el valor `aws:SourceArn` contiene el ID de cuenta, el valor `aws:SourceAccount` y la cuenta en el valor `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utiliza en la misma instrucción de política.

   - Use `aws:SourceArn` si quiere acceso entre servicios para un único recurso.
   - Use `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

   En la política, asegúrese de utilizar la clave de contexto de condición global `aws:SourceArn` con el ARN completo del recurso. En el siguiente ejemplo se muestra cómo se usa el comando de la AWS CLI para crear un rol denominado `rds-s3-export-role`.

   Example

   Para Linux, macOS o Unix:

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
```

```
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

Para Windows:

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

3. Asocie la política de IAM que creó al rol de IAM creado.

   El siguiente comando de la AWS CLI asocia la política creada anteriormente al rol denominado `rds-s3-export-role.`. Sustituya *`your-policy-arn`* por el ARN de la política que ha anotado en un paso anterior.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

4. Añada el rol de IAM al clúster de base de datos. Para ello, utilice la Consola de administración de AWS o la AWS CLI, tal y como se describe a continuación.

Consola

Para añadir un rol de IAM para un clúster de base de datos de PostgreSQL utilizando la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione el nombre de clúster de base de datos de PostgreSQL para mostrar sus detalles.
3. En la pestaña Connectivity & security (Conectividad y seguridad), en la sección Manage IAM roles (Administrar roles de IAM), elija el rol que desee añadir en Add IAM roles to this instance (Añadir roles de IAM a esta instancia).
4. En Feature (Característica), elija s3Export.
5. Seleccione Add role (Añadir rol).

AWS CLI

Para añadir un rol de IAM para un clúster de base de datos de PostgreSQL mediante la CLI, realice el siguiente procedimiento:

- Utilice el siguiente comando para añadir el rol al clúster de base de datos de PostgreSQL denominado `my-db-cluster`. Sustituya *your-role-arn* por el ARN del rol que ha anotado en el paso anterior. Utilice `s3Export` para el valor de la opción `--feature-name`.

  Example

  Para Linux, macOS o Unix:

```
aws rds add-role-to-db-cluster \
   --db-cluster-identifier my-db-cluster \
   --feature-name s3Export \
   --role-arn your-role-arn   \
   --region your-region
```

Para Windows:

```
aws rds add-role-to-db-cluster ^
   --db-cluster-identifier my-db-cluster ^
   --feature-name s3Export ^
   --role-arn your-role-arn ^
   --region your-region
```

Exportación de datos de consulta mediante la función aws_s3.query_export_to_s3

Exporte sus datos de PostgreSQL a Amazon S3 llamando a la función aws_s3.query_export_to_s3.

Temas

- Requisitos previos
- Llamar a aws_s3.query_export_to_s3
- Exportación a un archivo CSV que utiliza un delimitador personalizado
- Exportación a un archivo binario con codificación

Requisitos previos

Antes de utilizar la función `aws_s3.query_export_to_s3`, asegúrese de completar los siguientes requisitos previos:

- Instale las extensiones de PostgreSQL necesarias como se describe en Información general de la exportación de datos a Amazon S3.
- Determine a dónde exportar los datos en Amazon S3 como se describe en Especificación de la ruta del archivo de Amazon S3 a exportar.
- Tenga cuidado de que la instancia de base de datos tenga acceso de exportación a Amazon S3 según se describe en Configuración del acceso a un bucket de Amazon S3.

Los ejemplos siguientes utilizan una tabla de base de datos llamada `sample_table`. Estos ejemplos exportan los datos a un bucket llamado *`amzn-s3-demo-bucket`*. La tabla y los datos de ejemplo se crean con las siguientes instrucciones SQL en psql.

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
```

```
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

Llamar a aws_s3.query_export_to_s3

A continuación, se muestran las formas básicas de llamar a la función aws_s3.query_export_to_s3.

En estos ejemplos se utiliza la variable `s3_uri_1` para identificar una estructura que contiene la información que identifica el archivo de Amazon S3. Utilice la función aws_commons.create_s3_uri para crear la estructura.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Aunque los parámetros varían para las dos llamadas a funciones siguientes `aws_s3.query_export_to_s3`, los resultados son los mismos para estos ejemplos. Todas las filas de la tabla `sample_table` se exportan a un bucket llamado *amzn-s3-demo-bucket*.

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');

psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

Los parámetros se describen de la siguiente manera:

- `'SELECT * FROM sample_table'`: el primer parámetro es una cadena de texto requerida que contiene una consulta SQL. El motor de PostgreSQL ejecuta esta consulta. Los resultados de la consulta se copian en el bucket de S3 identificado en otros parámetros.
- `:'s3_uri_1'`: este parámetro es una estructura que identifica el archivo de Amazon S3. En este ejemplo se utiliza una variable para identificar la estructura creada anteriormente. En su lugar, puede crear la estructura incluyendo la llamada a la función `aws_commons.create_s3_uri` insertada dentro de la llamada a la función `aws_s3.query_export_to_s3` de la siguiente manera.

  ```
  SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
     aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
  ```

```
);
```

- `options :='format text'`: el parámetro `options` es una cadena de texto opcional que contiene argumentos `COPY` de PostgreSQL. El proceso de copia utiliza los argumentos y el formato del comando COPY de PostgreSQL.

Si el archivo especificado no existe en el bucket de Amazon S3, se crea. Si el archivo ya existe, se sobrescribe. La sintaxis para acceder a los datos exportados en Amazon S3 es la siguiente.

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

Las exportaciones más grandes se almacenan en varios archivos, cada uno con un tamaño máximo de aproximadamente 6 GB. Los nombres de archivo adicionales tienen el mismo prefijo de archivo, pero con `_part`*XX* anexado. *XX* representa 2, luego 3, y así sucesivamente. Por ejemplo, supongamos que especifica la ruta donde almacena los archivos de datos como sigue.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

Si la exportación tiene que crear tres archivos de datos, el bucket de Amazon S3 contiene los siguientes archivos de datos.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

Para obtener la referencia completa de esta función y formas adicionales de llamarla, consulte aws_s3.query_export_to_s3. Para obtener más información sobre el acceso a archivos en Amazon S3, consulte Ver un objeto en la Guía del usuario de Amazon Simple Storage Service.

Exportación a un archivo CSV que utiliza un delimitador personalizado

En el ejemplo siguiente se muestra cómo llamar a la función aws_s3.query_export_to_s3 para exportar datos a un archivo que utiliza un delimitador personalizado. En el ejemplo se utilizan argumentos del comando COPY de PostgreSQL para especificar el formato de valor separado por comas (CSV) y un delimitador de dos puntos (:).

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

Exportación a un archivo binario con codificación

En el ejemplo siguiente se muestra cómo llamar a la función aws_s3.query_export_to_s3 para exportar datos a un archivo binario que tiene codificación Windows-1253.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

Referencia de funciones

Funciones

- aws_s3.query_export_to_s3
- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

Exporta un resultado de consulta PostgreSQL a un bucket de Amazon S3. La extensión `aws_s3` proporciona la función `aws_s3.query_export_to_s3`.

Los parámetros obligatorios son `query` y `s3_info`. Definen la consulta que se va a exportar e identifican el bucket de Amazon S3 al que se va a exportar. Un parámetro opcional llamado `options` proporciona la definición de varios parámetros de exportación. Para obtener ejemplos sobre el uso de la función `aws_s3.query_export_to_s3`, consulte Exportación de datos de consulta mediante la función aws_s3.query_export_to_s3.

Sintaxis

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
)
```

Parámetros de entrada

consulta

Cadena de texto necesaria que contiene una consulta SQL que ejecuta el motor de PostgreSQL. Los resultados de esta consulta se copian en un bucket de S3 identificado en el parámetro `s3_info`.

s3_info

Tipo compuesto `aws_commons._s3_uri_1` que contiene la siguiente información sobre el objeto de S3:

- `bucket`: el nombre del bucket de Amazon S3 que contiene el archivo.
- `file_path` –: la ruta de Amazon S3 del archivo.
- `region`: la región de AWS en la que se encuentra el bucket. Para ver una lista de los nombres de regiones de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

  Actualmente, este valor debe ser la misma región de AWS que la del clúster de base de datos e de exportación. El valor predeterminado es la región de AWS del clúster de base de datos e de exportación.

Para crear una estructura compuesta `aws_commons._s3_uri_1`, consulte la función aws_commons.create_s3_uri.

options

Cadena de texto opcional que contiene argumentos para el comando `COPY` de PostgreSQL. Estos argumentos especifican cómo se copian los datos cuando se exportan. Para obtener más detalles, consulte la documentación de COPY de PostgreSQL.

kms_key text

Una cadena de texto opcional que contiene la clave KMS administrada por el cliente del bucket de S3 al que se exportan los datos.

Parámetros de entrada alternativos

Como ayuda en las pruebas, puede utilizar un conjunto de parámetros expandido en lugar del parámetro `s3_info`. A continuación, se incluyen otras variaciones de la sintaxis de la función `aws_s3.query_export_to_s3`.

En lugar de utilizar el parámetro `s3_info` para identificar un archivo de Amazon S3, utilice la combinación de los parámetros `bucket`, `file_path` y `region`.

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

consulta

Cadena de texto necesaria que contiene una consulta SQL que ejecuta el motor de PostgreSQL. Los resultados de esta consulta se copian en un bucket de S3 identificado en el parámetro `s3_info`.

bucket

Cadena de texto obligatoria que incluye el nombre del bucket de Amazon S3 que contiene el archivo.

file_path

Cadena de texto requerida que contiene la ruta de Amazon S3 del archivo.

region

Cadena de texto opcional que contiene la región de AWS en la que se encuentra el bucket. Para ver una lista de los nombres de regiones de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

Actualmente, este valor debe ser la misma región de AWS que la del clúster de base de datos e de exportación. El valor predeterminado es la región de AWS del clúster de base de datos e de exportación.

options

Cadena de texto opcional que contiene argumentos para el comando `COPY` de PostgreSQL. Estos argumentos especifican cómo se copian los datos cuando se exportan. Para obtener más detalles, consulte la documentación de COPY de PostgreSQL.

kms_key text

Una cadena de texto opcional que contiene la clave KMS administrada por el cliente del bucket de S3 al que se exportan los datos.

Parámetros de salida

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

Número de filas de tabla que se cargaron correctamente a Amazon S3 para la consulta dada.

files_uploaded

El número de archivos cargados en Amazon S3. Los archivos se crean en tamaños de aproximadamente 6 GB. Cada archivo adicional creado tiene `_part`*XX* anexado al nombre. *XX* representa 2, luego 3, y así sucesivamente según sea necesario.

bytes_uploaded

El número total de bytes cargados a Amazon S3.

Ejemplos

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Crea una estructura `aws_commons._s3_uri_1` para contener la información de archivos de Amazon S3. Debe utilizar los resultados de la función `aws_commons.create_s3_uri` en el parámetro `s3_info` de la función aws_s3.query_export_to_s3. Para ver un ejemplo de uso de la

función `aws_commons.create_s3_uri`, consulte Especificación de la ruta del archivo de Amazon S3 a exportar.

Sintaxis

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parámetros de entrada

bucket

Cadena de texto obligatoria que contiene el nombre del bucket de Amazon S3 del archivo.

file_path

Cadena de texto requerida que contiene la ruta de Amazon S3 del archivo.

region

Cadena de texto obligatoria que contiene la región de AWS en la que se encuentra el archivo. Para ver una lista de los nombres de regiones de AWS y los valores asociados, consulte Regiones y zonas de disponibilidad.

Solución de errores de acceso a Amazon S3

Si se producen problemas de conexión al intentar exportar los datos a Amazon S3, confirme primero que las reglas de acceso saliente del grupo de seguridad de la VPC asociado a la instancia de base de datos permitan la conectividad de red. En concreto, el grupo de seguridad debe tener una regla que permita que la instancia de base de datos envíe tráfico TCP al puerto 443 y a cualquier dirección IPv4 (0.0.0.0/0). Para obtener más información, consulte Proporcionar acceso al clúster de base de datos en la VPC mediante la creación de un grupo de seguridad.

También consulte las recomendaciones siguientes:

- Solución de problemas de identidades y accesos en Amazon Aurora
- Solución de problemas de Amazon S3 en la Guía del usuario de Amazon Simple Storage Service
- Solución de problemas de Amazon S3 e IAM en la Guía del usuario de IAM

Invocación de una función de AWS Lambda desde un clúster de base de datos de Aurora PostgreSQL

AWS Lambda es un servicio de computación controlado por eventos que permite ejecutar código sin aprovisionar ni administrar servidores. Está disponible para su uso con muchos servicios de AWS, incluidos Aurora PostgreSQL. Por ejemplo, puede utilizar funciones de Lambda para procesar notificaciones de eventos desde una base de datos o para cargar datos desde archivos cada vez que se carga un nuevo archivo en Simple Storage Service (Amazon S3). Para obtener más información sobre Lambda, consulte ¿Qué es AWS Lambda? en la Guía para desarrolladores de AWS Lambda.

> **Note**
>
> La invocación de funciones AWS Lambda se admite en Aurora PostgreSQL 11.9 y versiones posteriores (incluida Aurora Serverless v2).

La configuración de Aurora PostgreSQL para trabajar con las funciones de Lambda es un proceso de varios pasos que incluye AWS Lambda, IAM, su VPC y su clúster de base de datos de Aurora PostgreSQL. A continuación, se muestran resúmenes de los pasos necesarios.

Para obtener más información acerca de las funciones de Lambda, consulte Introducción a Lambda y Conceptos básicos de AWS Lambda en la Guía para desarrolladores de AWS Lambda.

Temas

- Paso 1: configure el clúster de base de datos de Aurora PostgreSQL para conexiones salientes a AWS Lambda.
- Paso 2: configure IAM para su clúster de base de datos de Aurora PostgreSQL y AWS Lambda.
- Paso 3: instale la extensión de aws_lambda para un clúster de base de datos de Aurora PostgreSQL
- Paso 4: utilice las funciones auxiliares de Lambda con su clúster de base de datos de Aurora PostgreSQL (Opcional)
- Paso 5: invoque una función de Lambda desde su clúster de base de datos de Aurora PostgreSQL
- Paso 6: Conceder permiso a otros usuarios para invocar las funciones de Lambda
- Ejemplos: invoque las funciones de Lambda desde su clúster de base de datos de Aurora PostgreSQL
- Mensajes de error de la función de Lambda

- Referencia de parámetros y funciones de AWS Lambda

Paso 1: configure el clúster de base de datos de Aurora PostgreSQL para conexiones salientes a AWS Lambda.

Las funciones de Lambda siempre se ejecutan dentro de una Amazon VPC propiedad del servicio de AWS Lambda. Lambda aplica acceso a la red y reglas de seguridad a esta VPC y mantiene y supervisa la VPC automáticamente. Su clúster de base de datos de Aurora PostgreSQL envía tráfico de red a la VPC del servicio de Lambda. La manera en que se configura esto depende de si la instancia de base de datos principal del clúster de Aurora DB es pública o privada.

- Clúster de base de datos de Aurora PostgreSQL público: una instancia principal de base de datos del clúster de base de datos es pública si se encuentra en una subred pública de la VPC y si la propiedad “PubliclyAccessible” de la instancia es `true`. Para encontrar el valor de esta propiedad, puede utilizar el comando describe-db-instances de AWS CLI. O, si lo desea, puede utilizar Consola de administración de AWS para abrir la pestaña Connectivity & security (Conectividad y seguridad) y verificar que Publicly accessible (Acceso público) sea Yes (Sí). Para comprobar que la instancia está en la subred pública de la VPC, puede utilizar la Consola de administración de AWS o la AWS CLI.

  Para configurar el acceso a Lambda, utilice la Consola de administración de AWS o la AWS CLI para crear una regla de salida en el grupo de seguridad de la VPC. La regla de salida especifica que TCP puede utilizar el puerto 443 para enviar paquetes a cualquier dirección IPv4 (0.0.0.0/0).
- Clúster de base de datos privado de Aurora PostgreSQL: en este caso, la propiedad “PubliclyAccessible” de la instancia es `false` o está en una subred privada. Para permitir el funcionamiento de la instancia con Lambda, puede utilizar una puerta de enlace de traducción de direcciones de red (NAT). Para obtener más información, consulte Puerta de enlace NAT. O bien, puede configurar su VPC con un punto de conexión de VPC para Lambda. Para obtener más información, consulte Puntos de enlace de la VPC en la Guía del usuario de Amazon VPC. El punto de conexión responde a las llamadas hechas por su clúster de base de datos de Aurora PostgreSQL a las funciones de Lambda.

La VPC ahora puede interactuar con la VPC de AWS Lambda en el ámbito de red. A continuación, debe configurar los permisos mediante IAM.

Paso 2: configure IAM para su clúster de base de datos de Aurora PostgreSQL y AWS Lambda.

La invocación de funciones de Lambda desde su clúster de base de datos de Aurora PostgreSQL requiere ciertos privilegios. Para configurar los privilegios necesarios, recomendamos crear una política de IAM que permita invocar funciones de Lambda, asignarla a un rol y, a continuación, aplicar el rol a su clúster de base de datos. Este enfoque da al clúster de base de datos privilegios para invocar la función de Lambda especificada en su nombre. En los pasos siguientes se muestra cómo hacer esto con AWS CLI.

Para configurar los permisos de IAM para utilizar su clúster con Lambda, lleve a cabo el siguiente procedimiento.

1. Utilice el comando create-policy de la AWS CLI para crear una política de IAM que permita a su clúster de base de datos de Aurora PostgreSQL invocar la función de Lambda especificada. (El ID de instrucción [Sid] es una descripción opcional de la instrucción de política y no afecta al uso). Esta política proporciona a su clúster de base de datos de Aurora los permisos mínimos necesarios para invocar la función de Lambda especificada.

   ```
   aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
       "Version": "2012-10-17",
       "Statement": [
           {
           "Sid": "AllowAccessToExampleFunction",
           "Effect": "Allow",
           "Action": "lambda:InvokeFunction",
           "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
           }
       ]
   }'
   ```

   También puede utilizar la política predefinida de `AWSLambdaRole` que le permite invocar cualquiera de las funciones de Lambda. Para obtener más información, consulte Políticas de IAM basadas en identidades para Lambda.

2. Utilice el comando de la AWS CLI create-role para crear un rol de IAM que la política pueda asumir en tiempo de ejecución.

   ```
   aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
       "Version": "2012-10-17",
   ```

```
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Aplique la política al rol mediante el comando attach-role-policy de AWS CLI.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Aplique el rol a su clúster de base de datos de Aurora PostgreSQL mediante el comando add-role-to-db-cluster de la AWS CLI. En este último paso se permite a los usuarios de bases de datos de su clúster de base de datos invocar funciones de Lambda.

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Con la VPC y las configuraciones de IAM completadas, ahora puede instalar la extensión de `aws_lambda`. (Tenga en cuenta que puede instalar la extensión en cualquier momento, pero hasta que no configure la compatibilidad con VPC y los privilegios de IAM correctos, la extensión de `aws_lambda` no agrega nada a las capacidades de su clúster de base de datos de Aurora PostgreSQL.

Paso 3: instale la extensión de **`aws_lambda`** para un clúster de base de datos de Aurora PostgreSQL

Para utilizar AWS Lambda con su clúster de base de datos de Aurora PostgreSQL, agregue la extensión de PostgreSQL de `aws_lambda` a su clúster de base de datos de Aurora PostgreSQL. Esta extensión proporciona a su clúster de base de datos de Aurora PostgreSQL la capacidad de llamar a funciones de Lambda desde PostgreSQL.

Para instalar la extensión de **aws_lambda** en su clúster de base de datos de Aurora PostgreSQL

Utilice la línea de comandos psql de PostgreSQL o la herramienta pgAdmin para conectarse a su clúster de base de datos de Aurora PostgreSQL .

1. Conéctese a su clúster de base de datos de Aurora PostgreSQL como usuario con privilegios de rds_superuser. El valor predeterminado de usuario de postgres se muestra en el ejemplo.

```
psql -h cluster-instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p
 5432
```

2. Instale la extensión de aws_lambda. La extensión de aws_commons también es necesaria. Proporciona funciones auxiliares para aws_lambda y muchas otras extensiones de Aurora para PostgreSQL. Si aún no está en su clúster de base de datos de Aurora PostgreSQL , se instala con aws_lambda como se muestra a continuación.

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

La extensión de aws_lambda se instala en su instancia de base de datos principal del clúster de base de datos de Aurora PostgreSQL Ahora puede crear estructuras de conveniencia para invocar las funciones de Lambda.

Paso 4: utilice las funciones auxiliares de Lambda con su clúster de base de datos de Aurora PostgreSQL (Opcional)

Puede utilizar las funciones auxiliares en la extensión de aws_commons para preparar entidades que puede invocar con más facilidad desde PostgreSQL. Para ello, debe tener la siguiente información sobre las funciones de Lambda:

- Nombre de la función: el nombre, el nombre de recurso de Amazon (ARN), la versión o el alias de la función de Lambda. La política de IAM creada en Paso 2: configure IAM para su clúster y Lambda requiere el ARN, por lo que recomendamos utilizar el ARN de su función.
- Región de AWS: (Opcional) la región de AWS en la que se encuentra la función de Lambda si no se encuentra en la misma región que su clúster de base de datos de Aurora PostgreSQL.

Para mantener la información del nombre de la función Lambda, utilice la función aws_commons.create_lambda_function_arn. Esta función auxiliar crea una estructura compuesta de `aws_commons._lambda_function_arn_1` con los detalles necesarios para la función de invocación. A continuación, encontrará tres enfoques alternativos para configurar esta estructura compuesta.

```
SELECT aws_commons.create_lambda_function_arn(
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

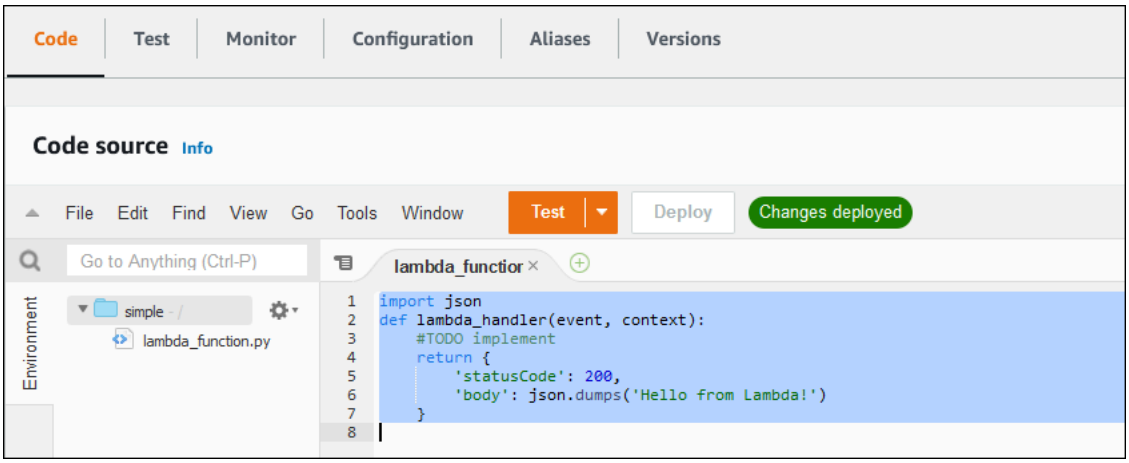
Cualquiera de estos valores se puede utilizar en las llamadas a la función aws_lambda.invoke. Para ver ejemplos, consulte Paso 5: invoque una función de Lambda desde su clúster de base de datos de Aurora PostgreSQL.

Paso 5: invoque una función de Lambda desde su clúster de base de datos de Aurora PostgreSQL

La función `aws_lambda.invoke` se comporta de forma sincrónica o asíncrona, según `invocation_type`. Las dos alternativas para este parámetro son `RequestResponse` (el valor predeterminado) y `Event`, como se muestra a continuación.

- **`RequestResponse`**: este tipo de invocación es sincrónico. Es el comportamiento predeterminado cuando la llamada se hace sin especificar un tipo de invocación. La carga de respuesta incluye los resultados de la función `aws_lambda.invoke`. Utilice este tipo de invocación cuando el flujo de trabajo requiera recibir los resultados de la función de Lambda antes de continuar.
- **`Event`**: este tipo de invocación es asíncrono. La respuesta no incluye una carga que contenga resultados. Utilice este tipo de invocación cuando el flujo de trabajo no necesite un resultado de la función de Lambda para continuar con el procesamiento.

Como simple prueba de la configuración, puede conectarse a la instancia de base de datos mediante `psql` e invocar una función de ejemplo desde la línea de comandos. Supongamos que tiene una de las funciones básicas configuradas en su servicio Lambda, como la sencilla función de Python que se muestra en la siguiente captura de pantalla.



Para invocar una función de ejemplo

1. Conéctese a la instancia de base de datos principal con `psql` o pgAdmin.

   ```
   psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. Invoque la función mediante su ARN.

   ```
   SELECT * from
    aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
   region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
    Postgres!"}'::json );
   ```

   La respuesta tiene el siguiente aspecto.

   ```
   status_code |                         payload                          |
    executed_version | log_result
   -------------+----------------------------------------------------------
   +------------------+------------
            200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
        |
   (1 row)
   ```

Si el intento de invocación no se lleva a cabo correctamente, consulte Mensajes de error de la función de Lambda .

Paso 6: Conceder permiso a otros usuarios para invocar las funciones de Lambda

En este punto de los procedimientos, solo usted como `rds_superuser` puede invocar las funciones de Lambda. Para permitir que otros usuarios puedan invocar cualquier función que haya creado usted, deberá otorgarles permiso.

Para otorgar permiso para invocar una función de Lambda

1. Conéctese a la instancia de base de datos principal con `psql` o pgAdmin.

   ```
   psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. Ejecute los siguientes comandos SQL:

   ```
   postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
   GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
   ```

Ejemplos: invoque las funciones de Lambda desde su clúster de base de datos de Aurora PostgreSQL

A continuación, puede encontrar varios ejemplos de llamada a la función de aws_lambda.invoke. La mayoría de ejemplos utilizan la estructura compuesta `aws_lambda_arn_1` que se crea en Paso 4: utilice las funciones auxiliares de Lambda con su clúster de base de datos de Aurora PostgreSQL (Opcional) para simplificar la transferencia de los detalles de la función. Para obtener un ejemplo de invocación asincrónica, consulte Ejemplo: invocación asincrónica (Event) de funciones de Lambda. El resto de ejemplos enumerados utilizan la invocación sincrónica.

Para obtener más información acerca de los tipos de invocación de Lambda, consulte Invocación de funciones de Lambda en la Guía para desarrolladores de AWS Lambda. Para obtener más información acerca de `aws_lambda_arn_1`, consulte aws_commons.create_lambda_function_arn.

Lista de ejemplos

- Ejemplo: invocación sincrónica (RequestResponse) de funciones de Lambda
- Ejemplo: invocación asincrónica (Event) de funciones de Lambda
- Ejemplo: captura del registro de ejecución de Lambda en una respuesta de función

- Ejemplo: inclusión del contexto del cliente en una función Lambda
- Ejemplo: invocación de una versión específica de una función de Lambda

Ejemplo: invocación sincrónica (RequestResponse) de funciones de Lambda

Lo que sigue son dos ejemplos de una invocación de función de Lambda sincrónica. Los resultados de estas llamadas de funciones de `aws_lambda.invoke` son iguales.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

Los parámetros se describen de la siguiente manera:

- `:'aws_lambda_arn_1'`: este parámetro identifica la estructura compuesta creada en Paso 4: utilice las funciones auxiliares de Lambda con su clúster de base de datos de Aurora PostgreSQL (Opcional), con la función auxiliar de `aws_commons.create_lambda_function_arn`. También puede crear esta estructura en línea dentro de su llamada de `aws_lambda.invoke` de la siguiente manera.

  ```
  SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
   'aws-region'),
  '{"body": "Hello from Postgres!"}'::json
  );
  ```

- `'{"body": "Hello from PostgreSQL!"}'::json` – La carga útil JSON que se va a pasar a la función Lambda.
- `'RequestResponse'` – El tipo de invocación Lambda.

Ejemplo: invocación asincrónica (Event) de funciones de Lambda

Lo que sigue es un ejemplo de una invocación de función asincrónica Lambda. El tipo de invocación `Event` programa la invocación de la función Lambda con la carga útil de entrada especificada y regresa inmediatamente. Utilice el tipo de invocación `Event` en ciertos flujos de trabajo que no dependen de los resultados de la función Lambda.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

Ejemplo: captura del registro de ejecución de Lambda en una respuesta de función

Puede incluir los últimos 4 KB del registro de ejecución en la respuesta de función mediante el parámetro `log_type` en su llamada a funciones de `aws_lambda.invoke`. De forma predeterminada, este parámetro se establece en `None`, pero puede especificar `Tail` para capturar los resultados del registro de ejecución de Lambda en la respuesta, como se muestra a continuación.

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Establezca el parámetro aws_lambda.invoke de la función `log_type` en `Tail` para incluir el registro de ejecución en la respuesta. El valor predeterminado para el parámetro `log_type` es `None`.

El `log_result` que se devuelve es una cadena codificada `base64`. Puede decodificar el contenido utilizando una combinación de las funciones `decode` y `convert_from` PostgreSQL.

Para obtener más información acerca de `log_type`, consulte aws_lambda.invoke.

Ejemplo: inclusión del contexto del cliente en una función Lambda

La función `aws_lambda.invoke` tiene un parámetro `context` que puede utilizar para pasar la información por separado de la carga, como se muestra a continuación.

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Para incluir el contexto del cliente, utilice un objeto JSON para el parámetro aws_lambda.invoke de la función `context`.

Para obtener más información sobre los parámetros de `context`, consulte la referencia de aws_lambda.invoke.

Ejemplo: invocación de una versión específica de una función de Lambda

Se puede especificar una versión concreta de una función de Lambda mediante el parámetro `qualifier` con la llamada de `aws_lambda.invoke`. A continuación, encontrará información sobre el ejemplo que hace esto mediante '*`custom_version`*' como alias de la versión.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

Además, puede proporcionar un calificador de función de Lambda con los detalles del nombre de función en su lugar de la siguiente manera.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

Para obtener más información acerca de `qualifier` y otros parámetros, consulte la referencia de aws_lambda.invoke.

Mensajes de error de la función de Lambda

En la siguiente lista encontrará información sobre los mensajes de error, con posibles causas y soluciones.

- Problemas de configuración de la VPC

  Los problemas de configuración de la VPC pueden generar los siguientes mensajes de error al intentar conectarse:

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  CONTEXT:  SQL function "invoke" statement 1
  ```

  Una causa común de este error es configurar erróneamente el grupo de seguridad de la VPC. Asegúrese de tener abierta una regla de salida para TCP en el puerto 443 de su grupo de seguridad de la VPC para que la VPC pueda conectarse a la VPC de Lambda.
- Falta de permisos necesarios para invocar funciones de Lambda

  Si ve alguno de los siguientes mensajes de error, significa que el usuario (rol) que invoca la función no tiene los permisos adecuados.

```
ERROR:  permission denied for schema aws_lambda
```

```
ERROR:  permission denied for function invoke
```

Se deben otorgar permisos específicos a un usuario (rol) para que pueda invocar funciones de Lambda. Para obtener más información, consulte Paso 6: Conceder permiso a otros usuarios para invocar las funciones de Lambda.

- Gestión incorrecta de errores en las funciones de Lambda

  Si una función Lambda lanza una excepción durante el procesamiento de la solicitud, `aws_lambda.invoke` se produce un error de PostgreSQL como el siguiente.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
ERROR:  lambda invocation failed
DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
 "Unhandled", details: "<Error details string>".
```

  Asegúrese de controlar los errores en las funciones de Lambda o en la aplicación de PostgreSQL.

Referencia de parámetros y funciones de AWS Lambda

A continuación, se presenta la referencia de las funciones y parámetros que se pueden utilizar para invocar Lambda con Aurora PostgreSQL.

Funciones y parámetros

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- Parámetros de aws_lambda

aws_lambda.invoke

Ejecuta una Lambda función destinada a un clúster deAurora PostgreSQL base de datos .

Para obtener más detalles acerca de la invocación de funciones de Lambda, consulte también Invoke en la guía para desarrolladores de AWS Lambda.

Sintaxis

JSON

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
```

```
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

Parámetros de entrada

function_name

El nombre de identificación de la función Lambda. El valor puede ser el nombre de la función, un ARN o un ARN parcial. Para obtener una lista de los formatos posibles, consulte los formatos de nombres de función de Lambda en la guía para desarrolladores de AWS Lambda.

payload

La entrada de la función Lambda. El formato puede ser JSON o JSONB. Para obtener más información, consulte la documentación de PostgreSQL sobre Tipos de JSON.

region

(Opcional) La región Lambda de la función. De forma predeterminada, Aurora resuelve la región de AWS desde el ARN completo en `function_name` o utiliza la región de instancia de base de datos de Aurora PostgreSQL. Si este valor de región entra en conflicto con el proporcionado en el ARN `function_name`, se genera un error.

invocation_type

Tipo de invocación de la función Lambda. El valor distingue entre mayúsculas y minúsculas. Entre los valores posibles se incluyen:

- `RequestResponse` – El valor de tiempo de espera predeterminado. Este tipo de invocación para una función Lambda es sincrónica y devuelve una carga útil de respuesta en el resultado.

Utilice el tipo de invocación de `RequestResponse` cuando el flujo de trabajo dependa de recibir el resultado de la función Lambda inmediatamente.

- `Event` – Este tipo de invocación para una función Lambda es asincrónica y regresa inmediatamente sin una carga útil devuelta. Utilice el tipo de invocación `Event` cuando no necesite resultados de la función Lambda antes de que el flujo de trabajo avance.
- `DryRun` – Este tipo de invocación prueba el acceso sin ejecutar la función Lambda.

log_type

El tipo de registro Lambda que se va a devolver en el parámetro `log_result` de salida. El valor distingue entre mayúsculas y minúsculas. Entre los valores posibles se incluyen:

- Final – El parámetro de salida `log_result` devuelto incluirá los últimos 4 KB del registro de ejecución.
- Ninguno – No se devuelve ninguna información de registro Lambda.

context

Contexto del cliente en formato JSON o JSONB. Los campos que se van a utilizar incluyen `custom` y `env`.

Calificador

Un calificador que identifica la versión de una función Lambda que se va a invocar. Si este valor entra en conflicto con uno proporcionado en el ARN `function_name`, se genera un error.

Parámetros de salida

status_code

Un código de respuesta de estado HTTP. Para obtener más información, consulte los elementos de respuesta de invocación de Lambda en la guía para desarrolladores de AWS Lambda.

payload

La información devuelta de la función Lambda que se ejecutó. El formato está en JSON o JSONB.

executed_version

La versión de la función Lambda que se ejecutó.

log_result

La información del registro de ejecución devuelta si el valor `log_type` es `Tail` cuando se invocó la función Lambda. El resultado contiene los últimos 4 KB del registro de ejecución codificado en Base64.

aws_commons.create_lambda_function_arn

Crea una estructura `aws_commons._lambda_function_arn_1` para contener la información del nombre de función Lambda. Puede utilizar los resultados de la función `aws_commons.create_lambda_function_arn` en el parámetro `function_name` de la función aws_lambda.invoke.

Sintaxis

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
    RETURNS aws_commons._lambda_function_arn_1
```

Parámetros de entrada

function_name

Una cadena de texto obligatoria que contiene el nombre de la función Lambda. El valor puede ser un nombre de función, un ARN parcial o un ARN completo.

region

Una cadena de texto opcional que contiene la región de AWS en la que se encuentra la función de Lambda. Para ver una lista de los nombres de regiones de y los valores asociados, consulte Regiones y zonas de disponibilidad.

Parámetros de aws_lambda

En esta tabla verá los parámetros asociados a la función `aws_lambda`.

Parámetro	Descripción
`aws_lambda.connect_timeout_ms`	Se trata de un parámetro dinámico y establece el tiempo máximo de espera durante la conexión a AWS Lambda. El valor predeterminado es `1000`. Los valores permitidos para este parámetro son de 1 a 900 000.
`aws_lambda.request_timeout_ms`	Se trata de un parámetro dinámico y establece el tiempo máximo de espera a la respuesta de AWS Lambda. El valor predeterminado es `3000`. Los valores permitidos para este parámetro son de 1 a 900 000.
`aws_lambda.endpoint_override`	Especifica el punto de conexión que se puede utilizar para conectarse a AWS Lambda. Una cadena vacía selecciona el punto de conexión de AWS Lambda predeterminado para la región. Debe reiniciar la base de datos para que se aplique el cambio en este parámetro estático.

Publicación de registros de Aurora PostgreSQL en Amazon CloudWatch Logs

Puede configurar el clúster de bases de datos de Aurora PostgreSQL para exportar datos de registro a Registros de Amazon CloudWatch de forma regular. Al hacerlo, los eventos del registro de PostgreSQL del clúster de base de datos de Aurora PostgreSQL se publican automáticamente en Amazon CloudWatch, como Registros de Amazon CloudWatch. En CloudWatch, puede encontrar los datos de registro exportados en un grupo de registropara el clúster de base de datos de Aurora PostgreSQL. El grupo de registro contiene uno o más flujos de registro que contienen los eventos del registro de PostgreSQL de cada instancia del clúster.

La publicación de registros en Registros de CloudWatch permite mantener los registros de PostgreSQL en un almacenamiento de larga duración. Con los datos de registro disponibles en Registros de CloudWatch, puede evaluar y mejorar las operaciones de su clúster. También puede utilizar CloudWatch para crear alarmas y visualizar métricas. Para obtener más información, consulte Monitoreo de eventos de registro en Amazon CloudWatch.

> **Note**
>
> La publicación de sus registros de PostgreSQL en Registros de CloudWatch consume almacenamiento y usted incurre en cargos por dicho almacenamiento. Asegúrese de eliminar los Registros de CloudWatch que ya no necesite.

La desactivación de la opción de exportación de registros para un clúster de base de datos de Aurora PostgreSQL existente no afecta a ningún dato que ya esté guardado en Registros de CloudWatch. Los registros existentes permanecen disponibles en Registros de CloudWatch en función de la configuración de retención de registros. Para obtener más información sobre Registros de CloudWatch, consulte el tema que explica qué es Registros de CloudWatch.

Aurora PostgreSQL admite la publicación de registros en Registros de CloudWatch en las siguientes versiones:

- Versión 16.1 y todas las versiones posteriores
- Versión 15.2 y versiones posteriores a la 15
- Versión 14.3 y versiones posteriores a la 14
- Versión 13.3 y versiones posteriores a la 13
- Versión 12.8 y versiones posteriores a la 12
- Versión 11.9 y versiones posteriores a la 11

Para obtener información sobre cómo activar la opción para publicar registros en Registros de CloudWatch, supervisar los eventos de registro en Registros de CloudWatch y analizar los registros con Información de registros de Amazon CloudWatch, consulte los siguientes temas.

Temas

- Activación de la opción de publicación de registros en Amazon CloudWatch
- Monitoreo de eventos de registro en Amazon CloudWatch
- Análisis de los registros de PostgreSQL mediante CloudWatch Logs Insights

Activación de la opción de publicación de registros en Amazon CloudWatch

Para publicar el registro de PostgreSQL del clúster de base de datos de Aurora PostgreSQL en Registros de CloudWatch, elija la opción Log export (Exportación del registro) para el clúster. Cuando

cree el clúster de base de datos de Aurora PostgreSQL, puede elegir la configuración de exportación de registros. O bien, puede modificar el clúster más adelante. Cuando modifica un clúster existente, los registros de PostgreSQL de cada instancia se publican en el clúster de CloudWatch a partir de ese momento. Para Aurora PostgreSQL, el registro de PostgreSQL (`postgresql.log`) es el único registro que se publica en Amazon CloudWatch.

Puede utilizar la Consola de administración de AWS, la AWS CLI o la API de RDS para activar la función de exportación de registros para el clúster de base de datos de Aurora PostgreSQL.

Consola

Para empezar a publicar los registros de PostgreSQL del clúster de base de datos de Aurora PostgreSQL en Registros de CloudWatch, elija la opción de exportación de registros.

Para activar la función de exportación de registros desde la consola

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster de base de datos de Aurora PostgreSQL cuyos datos de registro desea publicar en Registros de CloudWatch.
4. Elija Modify.
5. En la sección Log exports (Exportaciones de registros), elija Postgresql log (Registro de Postgresql).
6. Elija Continue (Continuar), seguido de Modify Cluster (Modificar clúster) en la página de resumen.

AWS CLI

Puede activar la opción de exportación de registros para empezar a publicar registros de Aurora PostgreSQL en Registros de Amazon CloudWatch con la AWS CLI. Para ello, ejecute el comando modify-db-cluster de la AWS CLI con las siguientes opciones:

- `--db-cluster-identifier`: identificador de clúster de base de datos.
- `--cloudwatch-logs-export-configuration`: el parámetro de configuración de los tipos de registros que se va a establecer para exportar a CloudWatch Logs para el clúster de base de datos.

También puede publicar registros de Aurora PostgreSQL si ejecuta uno de los siguientes comandos de la AWS CLI:

- create-db-cluster
- restore-db-clúster-from-s3
- restore-db-clúster-from-snapshot
- restore-db-clúster-to-point-in-time

Ejecute uno de estos comandos de la AWS CLI con las siguientes opciones:

- `--db-cluster-identifier`: identificador de clúster de base de datos.
- `--engine`: el motor de base de datos.
- `--enable-cloudwatch-logs-exports`: el ajuste de configuración para los tipos de registros que habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

Podrían ser necesarias otras opciones en función del comando de la AWS CLI que se ejecute.

Example

El siguiente comando crea un clúster de base de datos de Aurora PostgreSQL para publicar archivos de registro en CloudWatch Logs.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --engine aurora-postgresql \
    --enable-cloudwatch-logs-exports postgresql
```

Para Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --engine aurora-postgresql ^
    --enable-cloudwatch-logs-exports postgresql
```

Example

El siguiente comando modifica un clúster de base de datos de Aurora PostgreSQL existente para publicar archivos de registro en CloudWatch Logs. El valor `--cloudwatch-logs-export-configuration` es un objeto JSON. La clave de este objeto es `EnableLogTypes` y el valor es `postgresql` e `instance`.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","instance"]}'
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --cloudwatch-logs-export-configuration '{\"EnableLogTypes\":[\"postgresql\",
\"instance\"]}'
```

> **Note**
>
> Al utilizar el símbolo del sistema de Windows, asegúrese de aplicar escape con comillas dobles (") en código JSON al ponerlas como prefijo con una barra invertida (\).

Example

El siguiente ejemplo modifica un clúster de base de datos de Aurora PostgreSQL existente para desactivar la publicación de archivos de registro en CloudWatch Logs. El valor `--cloudwatch-logs-export-configuration` es un objeto JSON. La clave de este objeto es `DisableLogTypes` y el valor es `postgresql` e `instance`.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":
["postgresql","instance"]}'
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"postgresql\",
\"instance\"]}"
```

> **Note**
>
> Al utilizar el símbolo del sistema de Windows, debe aplicar escape con comillas dobles (") en código JSON al ponerlas como prefijo con una barra invertida (\).

API de RDS

Puede activar la opción de exportación de registros para empezar a publicar registros de Aurora PostgreSQL con la API de RDS. Para ello, ejecute la operación ModifyDBCluster con las siguientes opciones:

- `DBClusterIdentifier`: El identificador de clúster de base de datos.
- `CloudwatchLogsExportConfiguration`: el parámetro de configuración para los tipos de registros que se va a habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

También puede publicar logs de Aurora PostgreSQL con la API de RDS mediante la ejecución de una de las siguientes operaciones de API de RDS:

- CreateDBCluster
- RestoreDBclústerFromS3
- RestoreDBclústerFromSnapshot
- RestoreDBclústerToPointInTime

Ejecute la acción de la API de RDS con los siguientes parámetros:

- `DBClusterIdentifier`: identificador de clúster de base de datos.
- `Engine`: el motor de base de datos.
- `EnableCloudwatchLogsExports`: el ajuste de configuración para los tipos de registros que habilitar para exportar a CloudWatch Logs para el clúster de base de datos.

Podrían ser necesarios otros parámetros en función del comando de la AWS CLI que ejecute.

Monitoreo de eventos de registro en Amazon CloudWatch

Con los eventos de registro de Aurora PostgreSQL publicados y disponibles como Registros de Amazon CloudWatch, puede ver y supervisar los eventos mediante Amazon CloudWatch. Para obtener más información sobre el monitoreo, consulte Ver datos de registro enviados a CloudWatch Logs.

Cuando se activa la exportación de registros, se crea un nuevo grupo de registro de forma automática con el prefijo `/aws/rds/cluster/` con el nombre de su Aurora PostgreSQL y el tipo de registro, como en el siguiente patrón.

```
/aws/rds/cluster/your-cluster-name/postgresql
```

Por ejemplo, supongamos que un clúster de base de datos de Aurora PostgreSQL con el nombre `docs-lab-apg-small`exporta su registro a Registros de Amazon CloudWatch. A continuación se muestra el nombre del grupo de registro en Amazon CloudWatch.

```
/aws/rds/cluster/docs-lab-apg-small/postgresql
```

Si ya existe un grupo de registro con el nombre especificado, Aurora utilizará dicho grupo de registro para exportar los datos de registros para el clúster de base de datos Aurora. Cada instancia de base de datos del clúster de base de datos de Aurora PostgreSQL carga su registro de PostgreSQL en el grupo de registro como un flujo de registro distinto. Puede examinar el grupo de registro y sus flujos de registro mediante las diversas herramientas gráficas y analíticas disponibles en Amazon CloudWatch.

Por ejemplo, puede buscar información en los eventos de registro desde el clúster de base de datos de Aurora PostgreSQL y filtrar eventos con la consola de Registros de CloudWatch, la AWS CLI o la API de Registros de CloudWatch. Para obtener más información, consulte el tema sobre la búsqueda y el filtrado de datos de registros en la Guía del usuario de Registros de Amazon CloudWatch.

De forma predeterminada, los grupos de registro nuevos se crean mediante Never expire (Nunca caduca) para su periodo de retención. Puede utilizar la consola de CloudWatch Logs, la AWS CLI o la API de CloudWatch Logs para modificar el periodo de retención de registros. Para obtener más información, consulte el tema sobre cómo cambiar la retención de datos de registro en Registros de CloudWatch en la Guía de usuario de Registros de Amazon CloudWatch.

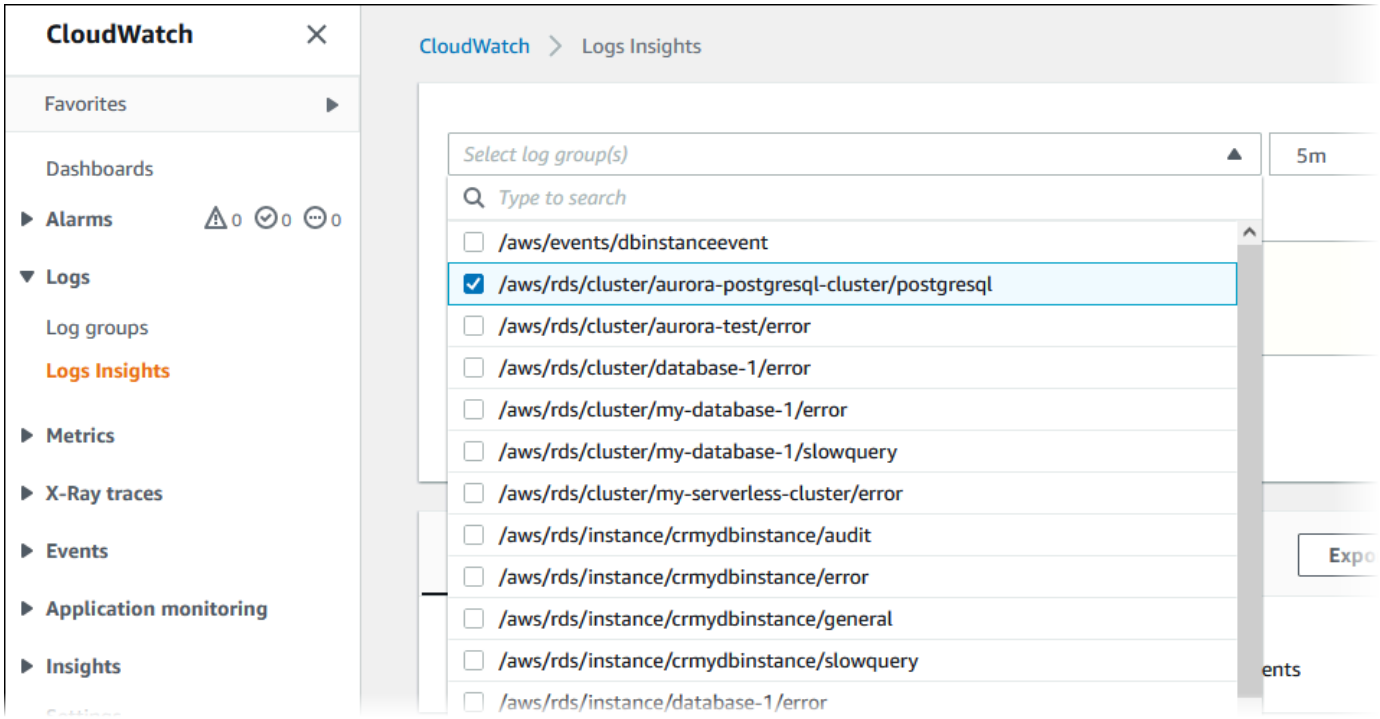
> **Tip**
>
> Puede utilizar la configuración automática, como AWS CloudFormation, para crear grupos de registro con periodos de retención de registro predefinidos, filtros de métricas y permisos de acceso.

Análisis de los registros de PostgreSQL mediante CloudWatch Logs Insights

Con los registros de PostgreSQL de su clúster de base de datos de Aurora PostgreSQL publicados como Registros de CloudWatch, puede usar CloudWatch Logs Insights para buscar y analizar de forma interactiva los datos de registro en Registros de Amazon CloudWatch. CloudWatch Logs Insights incluye un lenguaje de consulta, consultas de muestra y otras herramientas para analizar los datos de registro, de modo que pueda identificar posibles problemas y verificar las correcciones. Para obtener más información, consulte el tema sobre el análisis de los datos de registro con CloudWatch Logs Insights en la Guía del usuario de Registros de Amazon CloudWatch.

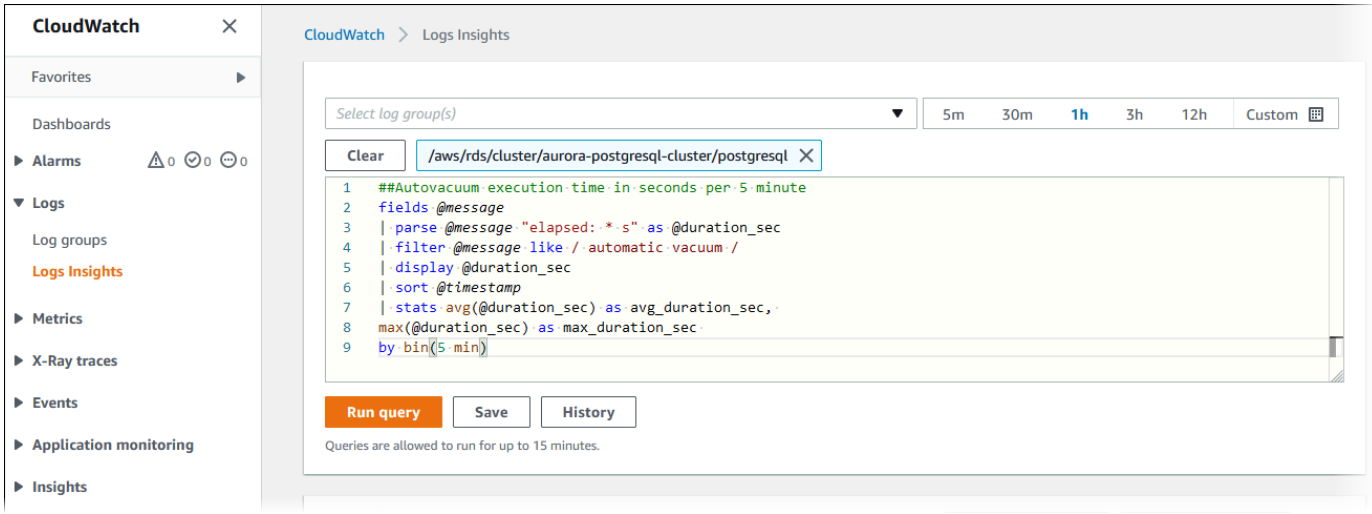
Para analizar los registros de PostgreSQL con CloudWatch Logs Insights

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, abra Logs (Registros) y elija Insights (Información).
3. En Select log group(s) (Seleccionar grupos de registro), seleccione el grupo de registro para el clúster de base de datos de Aurora PostgreSQL.

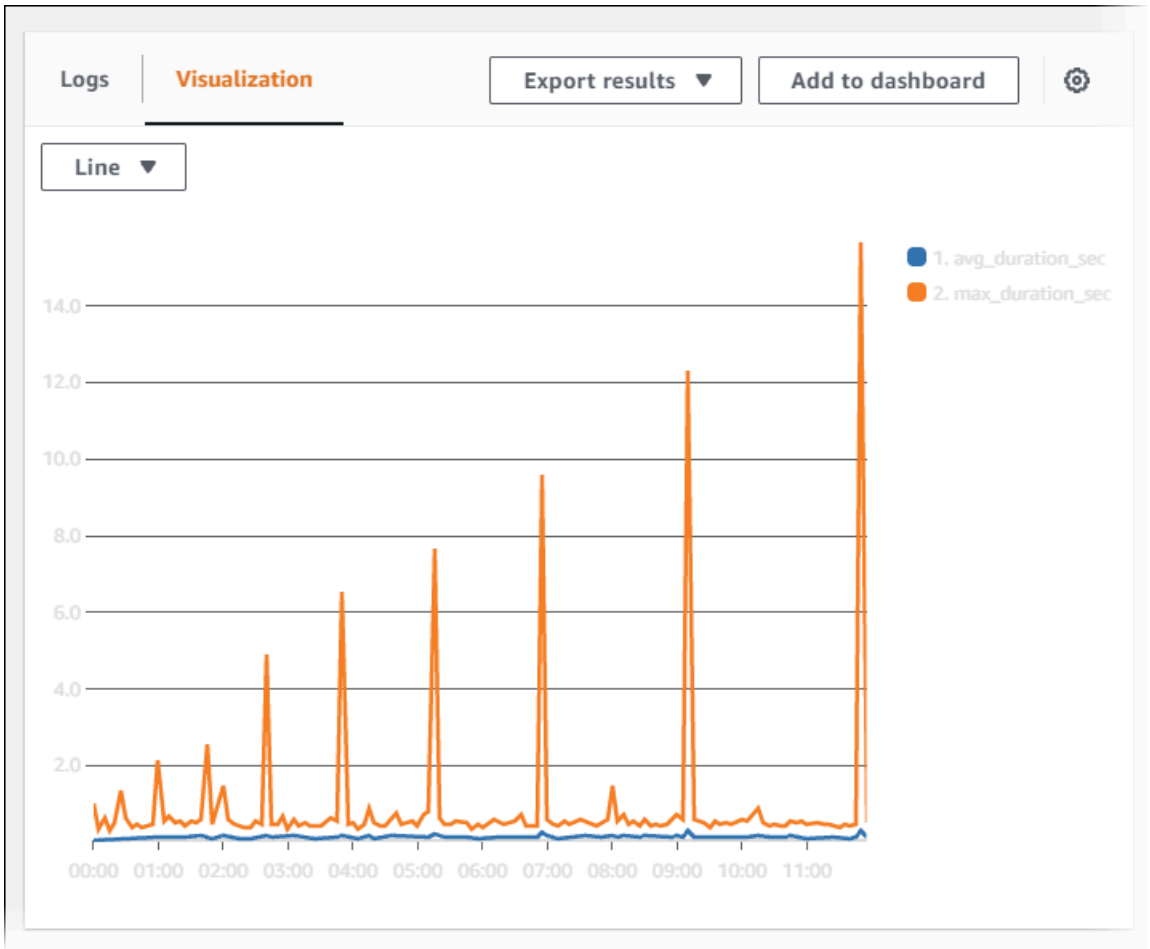


4. En el editor de consultas, elimine la consulta que se muestra actualmente y, a continuación, ingrese lo siguiente y elija Run query (Ejecutar consulta).

```
##Autovacuum execution time in seconds per 5 minute
fields @message
| parse @message "elapsed: * s" as @duration_sec
| filter @message like / automatic vacuum /
| display @duration_sec
| sort @timestamp
| stats avg(@duration_sec) as avg_duration_sec,
max(@duration_sec) as max_duration_sec
by bin(5 min)
```

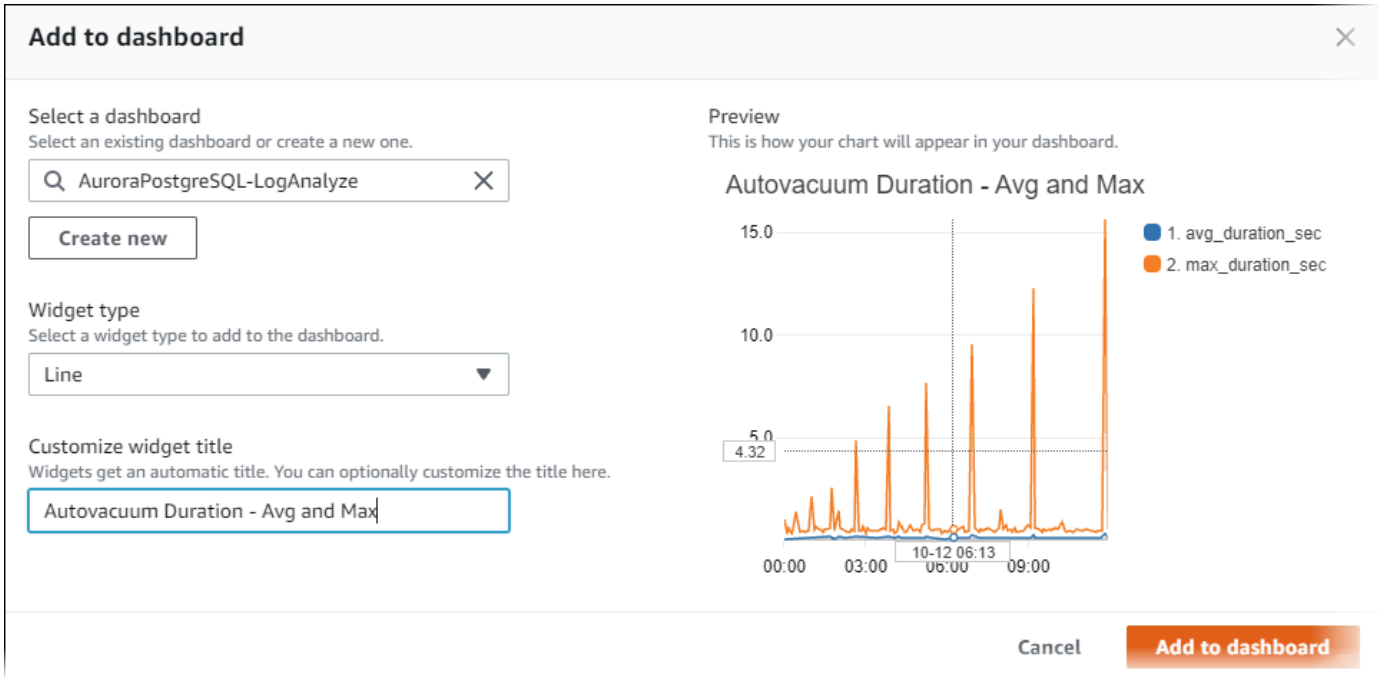


5. Elija la pestaña Visualization (Visualización).



6. Elija Add to dashboard (Añadir a panel).
7. En Select a dashboard (Seleccionar un panel), seleccione un panel o ingrese un nombre para crear un nuevo panel.

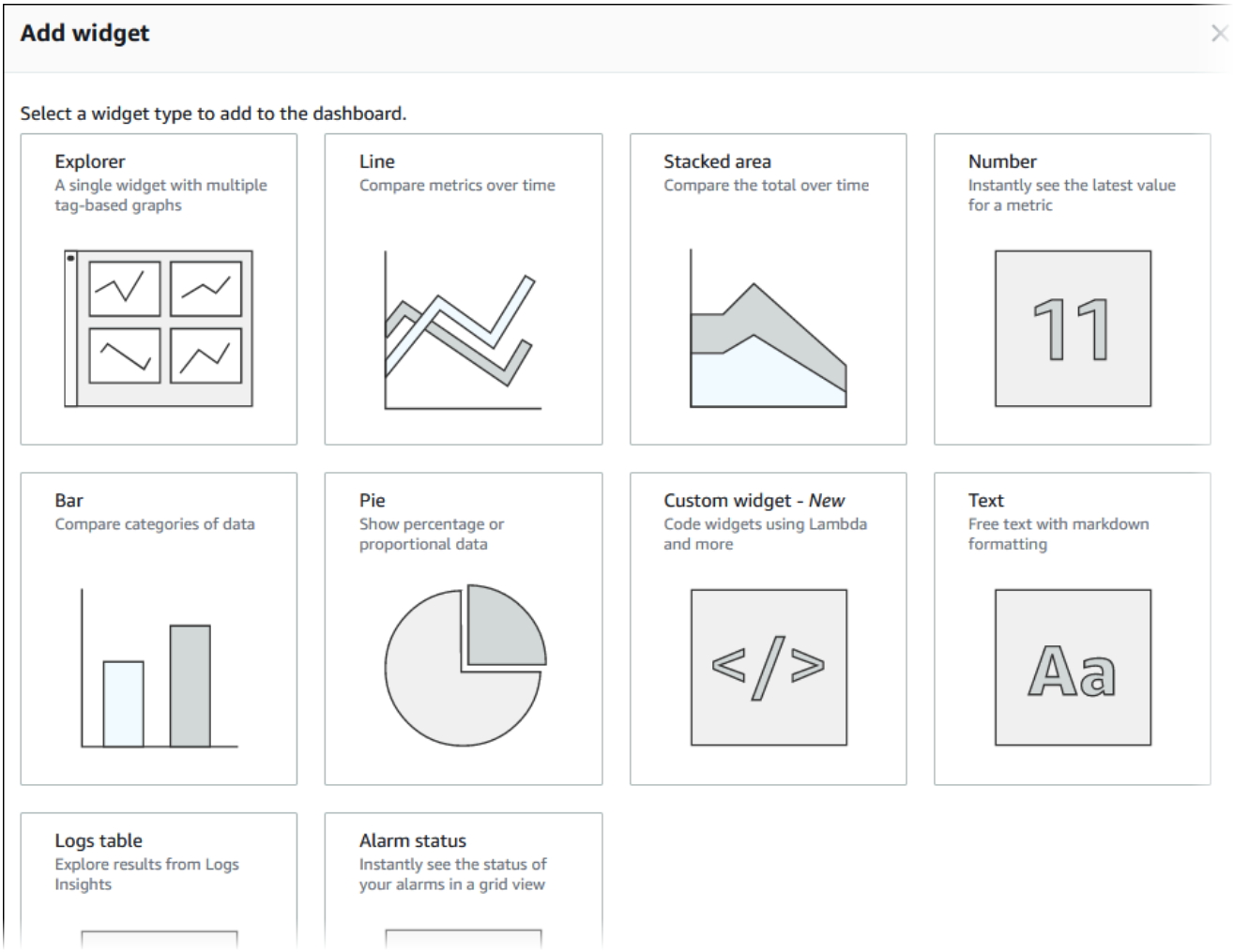
8. En Widget type (Tipo de widget), elija un tipo de widget para la visualización.



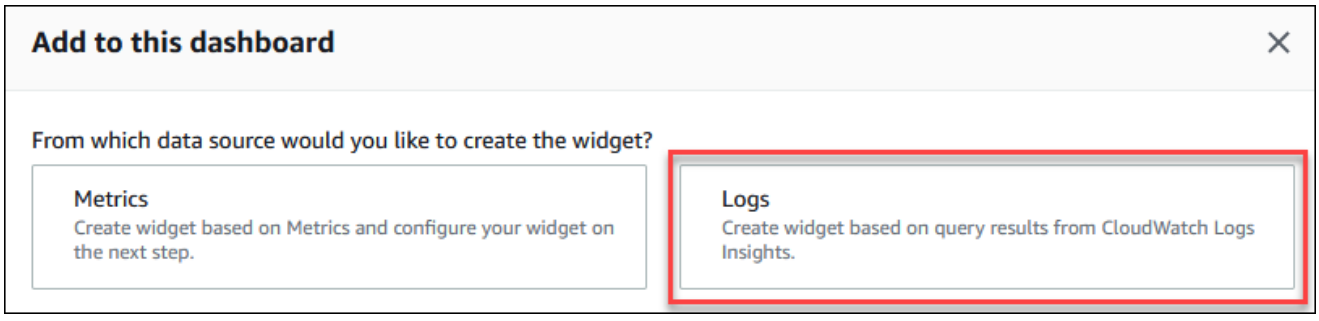
9. (Opcional) Agregue más widgets según los resultados de la consulta de registro.

   a. Elija Add widget (Añadir widget).

   b. Elija un tipo de widget, como Line (Línea).



c. En la ventana Add to this dashboard (Agregar a este panel), elija Logs (Registros).

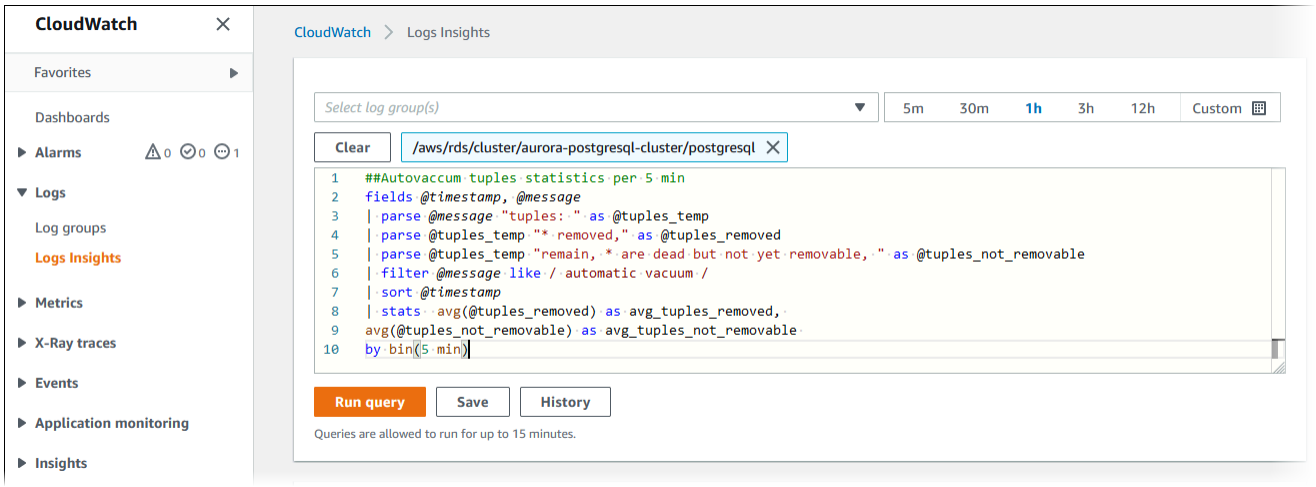


d. En Select log group(s) (Seleccionar grupos de registros), seleccione el grupo de registros para el clúster de base de datos.

e. En el editor de consultas, elimine la consulta que se muestra actualmente y, a continuación, ingrese lo siguiente y elija Run query (Ejecutar consulta).

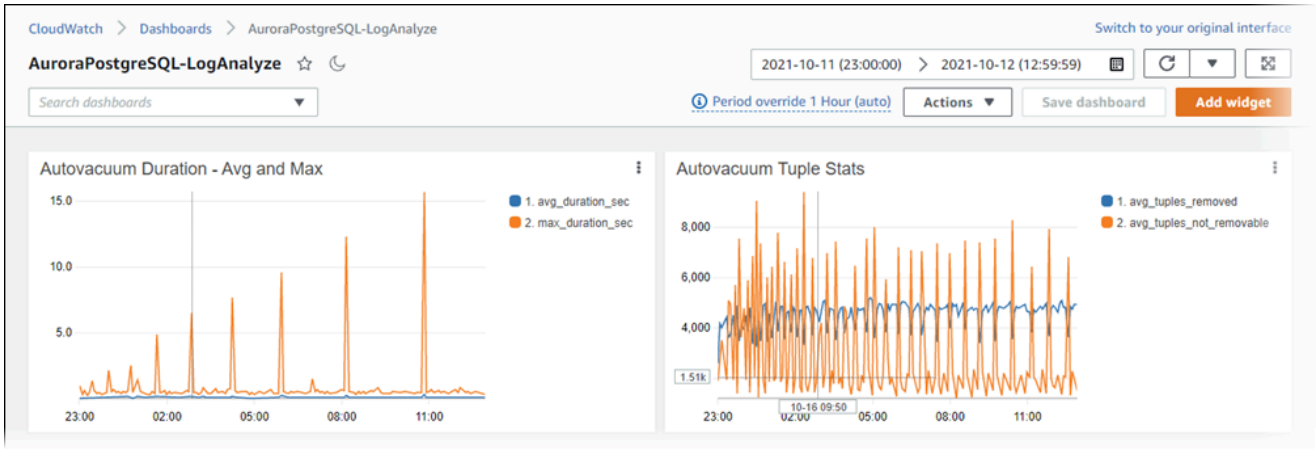
```
##Autovacuum tuples statistics per 5 min
fields @timestamp, @message
```

```
| parse @message "tuples: " as @tuples_temp
| parse @tuples_temp "* removed," as @tuples_removed
| parse @tuples_temp "remain, * are dead but not yet removable, " as
 @tuples_not_removable
| filter @message like / automatic vacuum /
| sort @timestamp
| stats  avg(@tuples_removed) as avg_tuples_removed,
avg(@tuples_not_removable) as avg_tuples_not_removable
by bin(5 min)
```



f. Elija Create widget (Crear widget).

El panel debería ser similar al de la siguiente imagen.



Monitorización de planes de ejecución de consultas y máximo de memoria para Aurora PostgreSQL

Puede supervisar los planes de ejecución de consultas en su instancia de base de datos de Aurora PostgreSQL para detectar los planes de ejecución que contribuyen a la carga actual de la base de datos y realizar un seguimiento de las estadísticas de rendimiento de los planes de ejecución a lo largo del tiempo mediante el parámetro `aurora_compute_plan_id`. Cada vez que se ejecuta una consulta, se asigna un identificador al plan de ejecución utilizado por la consulta y se utiliza el mismo identificador en las siguientes ejecuciones del mismo plan.

El `aurora_compute_plan_id` se `OFF` de forma predeterminada en el grupo de parámetros de base de datos de las versiones 14.10, 15.5 y posteriores de Aurora PostgreSQL. Para asignar un identificador de plan, establezca `aurora_compute_plan_id` en `ON` en el grupo de parámetros.

Este identificador de plan se usa en varias utilidades que tienen un propósito diferente.

A partir de las siguientes versiones, puede supervisar el uso máximo de memoria de consultas en su instancia de base de datos para detectar las consultas que contribuyen a un uso elevado de la memoria de la base de datos:

- Versión 16.3 y todas las versiones posteriores
- Versión 15.7 y posteriores
- Versión 14.12 y posteriores

Cada vez que se ejecuta una consulta, se realiza un seguimiento del consumo máximo de memoria utilizado por la consulta. Las consultas suelen ejecutarse varias veces; se pueden ver los valores de uso de memoria promedio, mínimo y máximo de todas las ejecuciones para cada consulta.

Temas

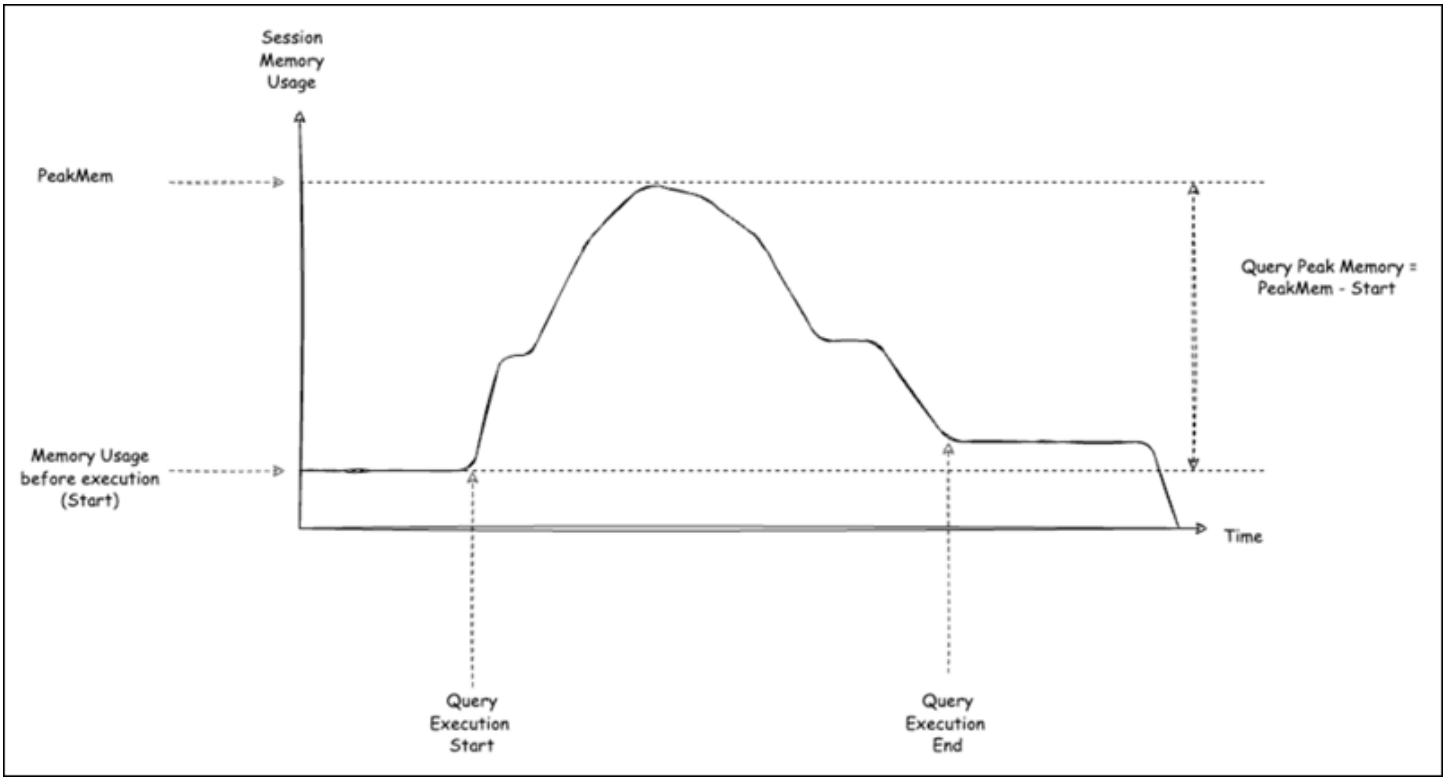
- Acceso a los planes de ejecución de consultas y máximo de memoria mediante las funciones de Aurora
- Referencia de parámetros para los planes de ejecución de consultas de Aurora PostgreSQL

Acceso a los planes de ejecución de consultas y máximo de memoria mediante las funciones de Aurora

Con `aurora_compute_plan_id`, puede acceder a los planes de ejecución mediante las siguientes funciones:

- aurora_stat_activity
- aurora_stat_plans

El máximo de memoria de la consulta no incluye la memoria que se asigna antes de que comience el procesamiento de la consulta. El uso máximo de memoria se registra y se informa por separado para las fases de planificación y ejecución de cada consulta.



Puede acceder a las estadísticas de uso máximo de memoria en consultas mediante estas funciones:

- aurora_stat_statements
- aurora_stat_plans

Para obtener más información sobre estas funciones, consulte Referencia de las funciones de Aurora PostgreSQL.

Referencia de parámetros para los planes de ejecución de consultas de Aurora PostgreSQL

Es posible supervisar planes de ejecución de consultas con los parámetros siguientes de un grupo de parámetros de base de datos.

Parámetros

- aurora_compute_plan_id
- aurora_stat_plans.minutes_until_recapture
- aurora_stat_plans.calls_until_recapture
- aurora_stat_plans.with_costs
- aurora_stat_plans.with_analyze
- aurora_stat_plans.with_timing
- aurora_stat_plans.with_buffers
- aurora_stat_plans.with_wal
- aurora_stat_plans.with_triggers

> **Note**
>
> La configuración de los parámetros `aurora_stat_plans.with_*` solo se surte efecto para los planes recién capturados.

aurora_compute_plan_id

`aurora_compute_plan_id` es un parámetro de configuración que controla si se asigna un identificador de plan durante la ejecución de la consulta.

Predeterminado/a	Valores permitidos	Descripción
off	0(off)	Póngalo en `off` para evitar que se asigne un identificador de plan.
	1(on)	Póngalo en on para asignar un identificador de plan.

aurora_stat_plans.minutes_until_recapture

El número de minutos que deben transcurrir antes de que se recupere un plan. El valor predeterminado es 0, lo que inhabilita la recaptura de un plan. Cuando se supera el umbral de `aurora_stat_plans.calls_until_recapture`, el plan se recaptura.

Predeterminado/a	Valores permitidos	Descripción
0	0-1073741823	Establece el número de minutos que deben transcurrir antes de que se recupere un plan.

aurora_stat_plans.calls_until_recapture

El número de llamadas a un plan antes de volver a capturarlo. El valor predeterminado es 0, lo que inhabilita la recuperación de un plan después de cierto número de llamadas. Cuando se supera el umbral de `aurora_stat_plans.minutes_until_recapture`, el plan se recaptura.

Predeterminado/a	Valores permitidos	Descripción
0	0-1073741823	Establece el número de llamadas antes de recuperar un plan.

aurora_stat_plans.with_costs

Captura un plan EXPLAIN con los costos estimados. Los valores permitidos son on y `off`. El valor predeterminado es on.

Predeterminado/a	Valores permitidos	Descripción
on	0(off)	No muestra el costo estimado ni las filas para cada nodo del plan.
	1(on)	Muestra el costo estimado y las filas de cada nodo del plan.

aurora_stat_plans.with_analyze

Controla el plan EXPLAIN con ANALYZE. Este modo solo se usa la primera vez que se captura un plan. Los valores permitidos son `on` y `off`. El valor predeterminado es `off`.

Predeterminado/a	Valores permitidos	Descripción
off	0(off)	No incluye las estadísticas del tiempo de ejecución real del plan.
	1(on)	Incluye las estadísticas de tiempo de ejecución real del plan.

aurora_stat_plans.with_timing

Los plazos del plan se reflejarán en la explicación cuando se utilice ANALYZE. El valor predeterminado es `on`.

Predeterminado/a	Valores permitidos	Descripción
on	0(off)	No incluye el tiempo real de puesta en marcha ni el tiempo dedicado a cada nodo del plan.
	1(on)	Incluye el tiempo real de puesta en marcha ni el tiempo dedicado a cada nodo del plan.

aurora_stat_plans.with_buffers

Las estadísticas de uso del búfer del plan se capturarán en explain cuando se utilice ANALYZE. El valor predeterminado es `off`.

Predeterminado/a	Valores permitidos	Descripción
off	0(off)	No incluye información sobre el uso del búfer.
	1(on)	Incluye información sobre el uso del búfer.

aurora_stat_plans.with_wal

Las estadísticas de uso del wal del plan se capturarán en explain cuando se utilice ANALYZE. El valor predeterminado es `off`.

Predeterminado/a	Valores permitidos	Descripción
off	0(off)	No incluye información sobre la generación de registros WAL.
	1(on)	Incluye información sobre la generación de registros WAL.

aurora_stat_plans.with_triggers

Las estadísticas de ejecución de los activadores del plan se capturarán en explain cuando `ANALYZE` se utilice. El valor predeterminado es `off`.

Predeterminado/a	Valores permitidos	Descripción
off	0(off)	No incluye las estadísticas de ejecución de los desencadenadores.
	1(on)	Incluye las estadísticas de ejecución de los desencadenadores.

Administración de planes de ejecución de consultas para Aurora PostgreSQL

La administración de planes de consultas de Aurora PostgreSQL es una función opcional que puede utilizar con su clúster de base de datos de Edición compatible con Amazon Aurora PostgreSQL. Esta característica se empaqueta como la extensión `apg_plan_mgmt` que puede instalar en su clúster de base de datos de Aurora PostgreSQL. La administración de planes de consulta le permite administrar los planes de ejecución de consultas generados por el optimizador para sus aplicaciones SQL. La extensión `apg_plan_mgmt` de AWS se basa en la funcionalidad de procesamiento de consultas nativa del motor de base de datos PostgreSQL.

A continuación, encontrará información sobre las funciones de administración del planes de consultas de Aurora PostgreSQL, cómo configurarlas y cómo utilizarlas con su clúster de base de datos de Aurora PostgreSQL. Antes de empezar, le recomendamos que revise las notas de la versión específica de la extensión `apg_plan_mgmt` disponible para su versión de Aurora PostgreSQL. Para obtener más información sobre Aurora PostgreSQL, consulte Aurora PostgreSQL apg_plan_mgmt extension versions (Versiones de la extensión apg_plan_mgmt de Aurora PostgreSQL) en las Notas de la versión de Aurora PostgreSQL.

Temas

- Descripción general de la administración de planes de consultas en Aurora PostgreSQL
- Prácticas recomendadas para la administración de planes de consultas de Aurora PostgreSQL
- Administración de planes de consultas en Aurora PostgreSQL
- Captura de planes de ejecución de Aurora PostgreSQL
- Uso de los planes administrados de Aurora PostgreSQL
- Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans
- Mejora de los planes de consultas en Aurora PostgreSQL
- Eliminación de planes de consultas en Aurora PostgreSQL
- Importación y exportación de planes administrados para Aurora PostgreSQL
- Referencia de parámetros para la administración de planes de consultas de Aurora PostgreSQL
- Referencia de funciones para la administración de planes de consultas de Aurora PostgreSQL
- Referencia de la vista apg_plan_mgmt.dba_plans para la edición compatible con Aurora PostgreSQL
- Funciones avanzadas de la administración de planes de consultas

Descripción general de la administración de planes de consultas en Aurora PostgreSQL

La administración de planes de consultas de Aurora PostgreSQL se ha diseñada para garantizar la estabilidad del plan independientemente de los cambios en la base de datos que puedan provocar una regresión del plan de consultas La regresión del plan de consultas se produce cuando el optimizador elige un plan subóptimo para una sentencia SQL determinada después de cambios en el sistema o la base de datos. Los cambios en estadísticas, restricciones, configuración del entorno, enlaces de parámetros de consultas y actualizaciones del motor de base de datos PostgreSQL pueden provocar una regresión del plan.

Con la gestión del plan de consultas de Aurora PostgreSQL, puede controlar cómo y cuándo cambian los planes de ejecución de las consultas. Entre los beneficios de la administración de planes de consultas en Aurora PostgreSQL se incluyen los siguientes.

- Obligar al optimizador a elegir a partir de un número limitado de planes buenos y conocidos para mejorar la estabilidad de los planes.
- Optimizar los planes de manera centralizada y, a continuación, distribuir los mejores planes globalmente.
- Identificar índices fuera de uso y evaluar el impacto de crear o anular un índice.
- Detectar automáticamente un nuevo plan de costo mínimo que el optimizador haya descubierto.
- Probar características de optimizador nuevas con un menor nivel de riesgo, porque puede optar por aprobar únicamente las modificaciones de planes que mejoren el rendimiento.

Puede utilizar las herramientas que proporciona la administración de planes de consultas de forma proactiva para especificar el mejor plan para determinadas consultas. O bien, puede utilizar la administración de planes de consultas para reaccionar ante las circunstancias cambiantes y evitar las regresiones del plan. Para obtener más información, consulte Prácticas recomendadas para la administración de planes de consultas de Aurora PostgreSQL.

Temas

- Instrucciones SQL compatibles
- Limitaciones de la administración de planes de consultas
- Terminología de administración de planes de consultas
- Versiones de administración de planes de consultas en Aurora PostgreSQL

- Activación de la administración de planes de consultas en Aurora PostgreSQL
- Actualización de la administración de planes de consultas en Aurora PostgreSQL
- Desactivación de la administración de planes de consultas en Aurora PostgreSQL

Instrucciones SQL compatibles

La administración de planes de consultas admite los siguientes tipos de instrucciones SQL.

- Cualquier instrucción SELECT, INSERT, UPDATE o DELETE, independientemente de su complejidad.
- Instrucciones preparadas. Para más información, consulte PREPARE en la documentación de PostgreSQL.
- Instrucciones dinámicas, incluidas las que se ejecutan en modo inmediato. Para obtener más información, consulte Dynamic SQL y EXECUTE IMMEDIATE en la documentación de PostgreSQL.
- Comandos e instrucciones SQL incrustados. Para obtener más información, consulte Embedded SQL Commands en la documentación de PostgreSQL.
- Instrucciones dentro de funciones denominadas. Para obtener más información, consulte CREATE FUNCTION en la documentación de PostgreSQL.
- Instrucciones que contienen tablas temporales.
- Instrucciones dentro de procedimientos y bloques DO.

Puede utilizar la administración de planes de consultas con `EXPLAIN` en modo manual para capturar un plan sin ejecutarlo en realidad. Para obtener más información, consulte Análisis del plan elegido por el optimizador. Para obtener más información sobre los modos de administración del plan de consultas (manual o automático), consulte Captura de planes de ejecución de Aurora PostgreSQL.

La administración de planes de consulta de Aurora PostgreSQL es compatible con todas las características del lenguaje PostgreSQL, incluidas las tablas particionadas, la herencia, la seguridad a nivel de fila y las expresiones comunes de tabla (CTE) recursivas. Para obtener más información sobre estas funciones del lenguaje PostgreSQL, consulte Table Partitioning, Row Security Policies y WITH Queries (Common Table Expressions) y otros temas en la documentación de PostgreSQL.

Para obtener información sobre las diferentes versiones de la característica de administración de planes de consultas de Aurora PostgreSQL, consulte las versiones de la extensión apg_plan_mgmt de Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

Limitaciones de la administración de planes de consultas

La versión actual de la administración de planes de consultas de Aurora PostgreSQL tiene las siguientes limitaciones.

- Los planes no se capturan para las instrucciones que hacen referencia a las relaciones del sistema: las instrucciones que hacen referencia a las relaciones del sistema como, por ejemplo `pg_class`, no se capturan. Esto es por diseño, para evitar que se capture una gran cantidad de planes generados por el sistema que se utilizan internamente. Esto también se aplica a las tablas del sistema dentro de las vistas.
- Es posible que se necesite una clase de instancia de base de datos más grande para el clúster de base de datos de Aurora PostgreSQL: según la carga de trabajo, la administración de planes de consultas podría necesitar una clase de instancia de base de datos que tenga más de 2 vCPU El número de `max_worker_processes` está limitado por el tamaño de la clase de instancia de base de datos. Es posible que el número de `max_worker_processes` que proporciona una clase de instancia de base de datos de 2 vCPU (db.t3.medium, por ejemplo) no sea suficiente para una carga de trabajo determinada. Le recomendamos que elija una clase de instancia de base de datos con más de 2 vCPU para su clúster de base de datos de Aurora PostgreSQL si utiliza la administración de planes de consultas.

  Cuando la clase de instancia de la base de datos no puede soportar la carga de trabajo, la administración de planes de consulta emite un mensaje de error como el siguiente.

  ```
  WARNING: could not register plan insert background process
  HINT: You may need to increase max_worker_processes.
  ```

  En este caso, debe ampliar el clúster de base de datos de Aurora PostgreSQL a un tamaño de clase de instancia de base de datos con más memoria. Para obtener más información, consulte Motores de base de datos compatibles para clases de instancia de base de datos.
- Los planes que ya estén almacenados en las sesiones no se ven afectados: la administración del plan de consultas proporciona una forma de influir en los planes de consultas sin cambiar el código de la aplicación. Sin embargo, si un plan genérico ya está almacenado en una sesión existente y desea cambiar su plan de consultas, primero debe configurar `plan_cache_mode` en `force_custom_plan` en el grupo de parámetros del clúster de base de datos.
- `queryid` en `apg_plan_mgmt.dba_plans` y `pg_stat_statements` pueden diferir cuando:
  - Los objetos se eliminan y se vuelven a crear después de guardarlos en apg_plan_mgmt.dba_plans.

- La tabla `apg_plan_mgmt.plans` se importa de otro clúster.

Para obtener información sobre las diferentes versiones de la característica de administración de planes de consultas de Aurora PostgreSQL, consulte las versiones de la extensión apg_plan_mgmt de Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

Terminología de administración de planes de consultas

Los siguientes términos se utilizan en este tema:

instrucción administrada

Una instrucción SQL capturada por el optimizador mediante la administración del plan de consultas. Una instrucción administrada tiene uno o más planes de ejecución de consultas almacenados en la vista `apg_plan_mgmt.dba_plans`.

línea base del plan

El conjunto de planes aprobados para una instrucción administrada determinada. Es decir, todos los planes de la instrucción administrada que tienen «Aprobado» en su columna `status` de la vista de `dba_plan`.

historial del plan

El conjunto de todos los planes capturados para una instrucción administrada determinada. El historial del plan contiene todos los planes capturados para la instrucción, independientemente de su estado.

regresión del plan de consulta

El caso en el que el optimizador elige un plan menos óptimo que antes de un cambio determinado en el entorno de la base de datos, como una nueva versión de PostgreSQL o cambios en las estadísticas.

Versiones de administración de planes de consultas en Aurora PostgreSQL

La administración de planes de consultas es compatible con todas las versiones de Aurora PostgreSQL disponibles actualmente. Para obtener información detallada sobre la versión, consulte la lista de actualizaciones de Amazon Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

La funcionalidad de administración de planes de consultas se agrega al clúster de base de datos de Aurora PostgreSQL al instalar la extensión `apg_plan_mgmt`. Las distintas versiones de Aurora PostgreSQL admiten distintas versiones de la extensión `apg_plan_mgmt`. Le recomendamos que actualice la extensión de administración de planes de consultas a la versión más reciente de su versión de Aurora PostgreSQL.

> **Note**
>
> Para ver las notas de la versión de cada una de las versiones de la extensión `apg_plan_mgmt`, consulte las versiones de extensión apg_plan_mgmt de Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

Puede identificar la versión que se ejecuta en su clúster conectándose a una instancia mediante `psql` y utilizando el metacomando\ dx para enumerar las extensiones, tal como se muestra a continuación.

```
labdb=> \dx
                        List of installed extensions
     Name      | Version |    Schema     |                            Description
---------------+---------+---------------
+-------------------------------------------------------------------
 apg_plan_mgmt | 1.0     | apg_plan_mgmt | Amazon Aurora with PostgreSQL compatibility
 Query Plan Management
 plpgsql       | 1.0     | pg_catalog    | PL/pgSQL procedural language
(2 rows)
```

El resultado muestra que este clúster utiliza la versión 1.0 de la extensión. Solo hay ciertas versiones de `apg_plan_mgmt` disponibles para una versión determinada de Aurora PostgreSQL. En algunos casos, es posible que necesite actualizar el clúster de base de datos de Aurora PostgreSQL a una nueva versión secundaria o aplicar un parche para poder actualizar a la versión más reciente de la administración del plan de consultas. La versión 1.0 de `apg_plan_mgmt`que se muestra en el resultado proviene de un clúster de base de datos de Aurora PostgreSQL versión 10.17, que no tiene una versión más reciente de `apg_plan_mgmt` disponible. En este caso, el clúster de base de datos de Aurora PostgreSQL debe actualizarse a una versión más reciente de PostgreSQL.

Para obtener más información sobre la actualización de su clúster de bases de datos Aurora PostgreSQL a una nueva versión de PostgreSQL, consulte Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL.

Para obtener información sobre cómo actualizar la extensión `apg_plan_mgmt`, consulte Actualización de la administración de planes de consultas en Aurora PostgreSQL.

Activación de la administración de planes de consultas en Aurora PostgreSQL

La configuración de la administración de planes de consultas para el clúster de base de datos de Aurora PostgreSQL implica instalar una extensión y cambiar varios parámetros del clúster de base de datos. Necesita permisos de `rds_superuser` para instalar la extensión `apg_plan_mgmt` y activar la función del clúster de base de datos de Aurora PostgreSQL.

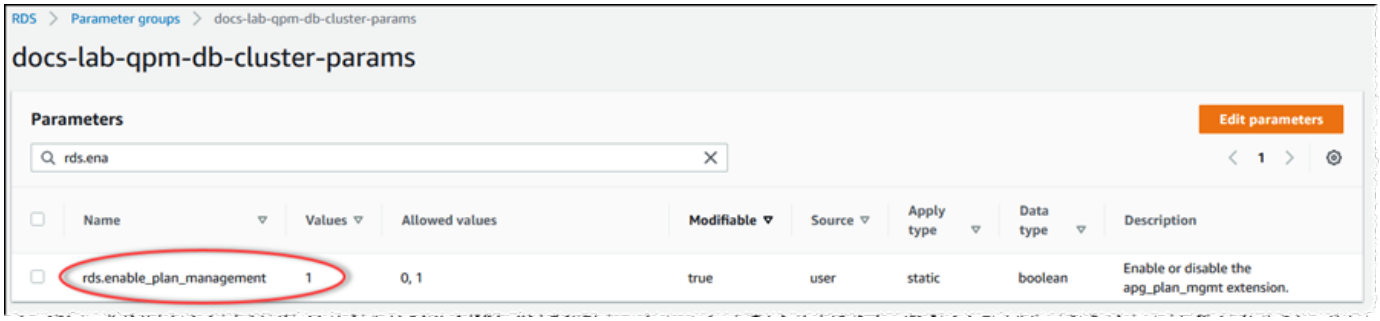
Al instalar la extensión, se crea un nuevo rol, `apg_plan_mgmt`. Esta función permite a los usuarios de bases de datos ver, administrar y mantener planes de consultas. Como administrador con privilegios `rds_superuser`, asegúrese de conceder el rol de `apg_plan_mgmt` a los usuarios de la base de datos según sea necesario.

Sólo los usuarios con el rol `rds_superuser` pueden completar el procedimiento siguiente. El `rds_superuser` es necesario para crear la extensión `apg_plan_mgmt` y su `apg_plan_mgmt` función. Se debe conceder a los usuarios el rol `apg_plan_mgmt` para administrar la extensión `apg_plan_mgmt`.

Para activar la administración de planes de consulta para su clúster de bases de datos Aurora PostgreSQL

Los siguientes pasos activan la administración de planes de consultas para todas las instrucciones SQL que se envían al clúster de base de datos de Aurora PostgreSQL. Esto se conoce como modo automático. Para obtener más información sobre la diferencia entre modos, consulte Captura de planes de ejecución de Aurora PostgreSQL.

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Cree un grupo de parámetros de clúster de base de datos personalizado para su clúster de base de datos de Aurora PostgreSQL. Debe cambiar ciertos parámetros para activar la administración de planes de consultas y establecer su comportamiento. Para obtener más información, consulte Creación de un grupo de parámetros de base de datos en Amazon Aurora.
3. Abra el grupo de parámetros de clúster de base de datos personalizado y establezca el parámetro `rds.enable_plan_management` en `1`, como se muestra en la siguiente imagen.



Para obtener más información, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

4. Cree un grupo de parámetros de base de datos personalizado que pueda usar para establecer los parámetros del plan de consultas a nivel de instancia. Para obtener más información, consulte Creación de un grupo de parámetros de clúster de base de datos en Amazon Aurora.
5. Modifique la instancia del escritor del clúster de base de datos de Aurora PostgreSQL para que utilice el grupo de parámetros de base de datos personalizado Para obtener más información, consulte Modificación de una instancia de base de datos en un clúster de base de datos.
6. Modifique el clúster de base de datos de Aurora PostgreSQL para que utilice el grupo de parámetros del clúster de base de datos personalizado Para obtener más información, consulte Modificación del clúster de base de datos con la consola, CLI y API.
7. Reinicie la instancia de base de datos para habilitar la configuración del grupo de parámetros personalizado.
8. Conecte al punto de conexión de la instancia de base de datos de Aurora PostgreSQL mediante `psql` o `pgAdmin`. En el siguiente ejemplo, se usa la cuenta de `postgres` predeterminada del rol de `rds_superuser`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=my-db
```

9. Cree la extensión `apg_plan_mgmt` para su instancia de base de datos, como se muestra a continuación.

```
labdb=> CREATE EXTENSION apg_plan_mgmt;
CREATE EXTENSION
```

> **Tip**
>
> Instale la extensión `apg_plan_mgmt` en la base de datos de plantillas de la aplicación. La base de datos de plantillas predeterminada se denomina `template1`. Para obtener más información, consulte Template Databases en la documentación de PostgreSQL.

10. Cambie el parámetro `apg_plan_mgmt.capture_plan_baselines` por `automatic`. Esta configuración hace que el optimizador genere planes para cada instrucción SQL que esté planificada o que se ejecute dos o más veces.

    > **Note**
    >
    > La administración de planes de consultas también tiene un modo manual que puede utilizar para instrucciones SQL específicas. Para obtener más información, consulte Captura de planes de ejecución de Aurora PostgreSQL.

11. Cambie el valor del parámetro `apg_plan_mgmt.use_plan_baselines` a «on». Este parámetro hace que el optimizador elija un plan para la instrucción a partir de la línea base del plan. Para obtener más información, consulte Uso de los planes administrados de Aurora PostgreSQL.

    > **Note**
    >
    > Puede modificar el valor de cualquiera de estos parámetros dinámicos de la sesión sin necesidad de reiniciar la instancia.

Cuando la configuración de administración de planes de consultas esté completa, asegúrese de conceder el rol de `apg_plan_mgmt` a todos los usuarios de la base de datos que necesiten ver, administrar o mantener los planes de consultas.

Actualización de la administración de planes de consultas en Aurora PostgreSQL

Le recomendamos que actualice la extensión de administración de planes de consultas a la versión más reciente de su versión de Aurora PostgreSQL.

1. Conecte a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL como un usuario que tiene privilegios de `rds_superuser`. Si mantuvo el nombre predeterminado al

configurar la instancia, conéctese como `postgres`. En este ejemplo se muestra cómo utilizar `psql`, pero también puede usar pgAdmin si lo prefiere.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Ejecute la siguiente consulta para actualizar la extensión.

```
ALTER EXTENSION apg_plan_mgmt UPDATE TO '2.1';
```

3. Use la función apg_plan_mgmt.validate_plans para actualizar los hashes de todos los planes. El optimizador valida todos los planes aprobados, no aprobados y rechazados para garantizar que sigan siendo planes viables para la nueva versión de la extensión.

```
SELECT apg_plan_mgmt.validate_plans('update_plan_hash');
```

   Para obtener más información sobre el uso de esta función, consulte Validación de planes.

4. Utilice la función apg_plan_mgmt.reload para actualizar cualquier plan de la memoria compartida con los planes validados de la vista dba_plans.

```
SELECT apg_plan_mgmt.reload();
```

Para obtener más información sobre todas las funciones disponibles para la administración del plan de consultas, consulte Referencia de funciones para la administración de planes de consultas de Aurora PostgreSQL.

Desactivación de la administración de planes de consultas en Aurora PostgreSQL

Puede deshabilitar la administración de planes de consultas en cualquier momento al desactivar `apg_plan_mgmt.use_plan_baselines` y `apg_plan_mgmt.capture_plan_baselines`.

```
labdb=> SET apg_plan_mgmt.use_plan_baselines = off;

labdb=> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Prácticas recomendadas para la administración de planes de consultas de Aurora PostgreSQL

La administración de planes de consultas le permite controlar cuándo y cómo cambian los planes de ejecución de consultas. Como DBA, sus principales objetivos al utilizar QPM incluyen la prevención de regresiones cuando se producen cambios en la base de datos y controlar si permitir que el optimizador utilice un nuevo plan. A continuación, encontrará algunas prácticas recomendadas para utilizar la administración de planes de consultas. Los enfoques de administración de planes proactivos y reactivos difieren en el cómo y el cuándo se aprueban los nuevos planes para su uso.

Contenido

- Administración de planes proactiva para evitar la regresión del rendimiento
  - Garantizar la estabilidad del plan después de una actualización a una versión principal
- Administración de planes reactiva para detectar y reparar regresiones del rendimiento

Administración de planes proactiva para evitar la regresión del rendimiento

Para evitar que se produzcan regresiones en el rendimiento del plan, desarrolle las bases de referencia del plan mediante la ejecución de un procedimiento que compare el rendimiento de los planes recién descubiertos con el rendimiento de las bases de referencia existentes de los planes aprobados y, a continuación, apruebe automáticamente el conjunto de planes más rápido como la nueva base de referencia. De esta manera, la base de referencia de los planes mejora con el tiempo a medida que se detectan planes más rápidamente.

1. En un entorno de desarrollo, identifique las instrucciones SQL que causen un mayor impacto en el rendimiento del sistema. Después, capture los planes para estas instrucciones según se describen en Captura manual de planes para instrucciones SQL específicas y Captura de planes automática.
2. Exporte los planes capturados desde el entorno de desarrollo e impórtelos al entorno de producción. Para obtener más información, consulte Importación y exportación de planes administrados para Aurora PostgreSQL.
3. En producción, ejecute su aplicación y fuerce el uso de planes administrados aprobados. Para obtener más información, consulte Uso de los planes administrados de Aurora PostgreSQL. Cuando se ejecute la aplicación, añada nuevos planes también a medida que el optimizador los descubra. Para obtener más información, consulte Captura de planes automática.
4. Analice los planes no aprobados y apruebe los que muestren un buen rendimiento. Para obtener más información, consulte Evaluación del rendimiento de los planes.

5. Mientras la aplicación continúa ejecutándose, el optimizador empezará a utilizar los nuevos planes, según resulte conveniente.

Garantizar la estabilidad del plan después de una actualización a una versión principal

Cada versión principal de PostgreSQL incluye mejoras y cambios en el optimizador de consultas que están diseñados para mejorar el rendimiento. Sin embargo, los planes de ejecución de consultas generados por el optimizador en versiones anteriores pueden provocar regresiones en el rendimiento en las versiones actualizadas más recientes. Puede utilizar el administrador de planes de consultas para resolver estos problemas de rendimiento y garantizar la estabilidad del plan tras una actualización de la versión principal.

El optimizador siempre utiliza un plan aprobado con el coste mínimo, incluso si hay más de un plan aprobado para la misma instrucción. Tras una actualización, es posible que el optimizador descubra nuevos planes, pero se guardarán como Planes no aprobados. Estos planes solo se ejecutan si se aprueban mediante el estilo reactivo de gestión de planes con el parámetro unapproved_plan_execution_threshold. Para maximizar la estabilidad del plan, puede utilizar el estilo proactivo de gestión de planes con el parámetro evolve_plan_baselines. De este modo, se compara el rendimiento de los nuevos planes con el de los antiguos y se aprueban o rechazan aquellos que sean al menos un 10 % más rápidos que el siguiente mejor plan.

Después de la actualización, puede utilizar la función `evolve_plan_baselines` para comparar el rendimiento del plan antes y después de la actualización utilizando los enlaces de parámetros de consultas. En los siguientes pasos, se supone que ha estado utilizando planes administrados aprobados en su entorno de producción, como se detalla en Uso de los planes administrados de Aurora PostgreSQL.

1. Antes de actualizar, ejecute la aplicación con el administrador de planes de consultas en ejecución. Mientras se ejecuta la aplicación, añada nuevos planes a medida que el optimizador los detecta. Para obtener más información, consulte Captura de planes automática.
2. Evalúe el rendimiento de cada plan. Para obtener más información, consulte Evaluación del rendimiento de los planes.
3. Después de actualizar, analice de nuevo los planes aprobados utilizando la función `evolve_plan_baselines`. Compare el rendimiento antes y después de utilizar los enlaces de parámetros de consultas. Si el nuevo plan es rápido, puede añadirlo a los planes aprobados. Si es más rápido que otro plan para los mismos enlaces de parámetros, puede marcar el plan más lento como rechazado.

Para obtener más información, consulte Aprobar planes mejores. Para obtener información de referencia acerca de esta función, consulte apg_plan_mgmt.evolve_plan_baselines.

Para obtener más información, consulte Ensuring consistent performance after major version upgrades with Amazon Aurora PostgreSQL-Compatible Edition Query Plan Management.

> **Note**
>
> Cuando actualice una versión principal mediante la replicación lógica o AWS DMS, asegúrese de replicar el esquema de `apg_plan_mgmt` para garantizar que los planes existentes se copien en la instancia actualizada. Para obtener más información sobre la replicación lógica, consulte Uso de la replicación lógica para realizar una actualización de la versión principal para Aurora PostgreSQL.

Administración de planes reactiva para detectar y reparar regresiones del rendimiento

Al monitorizar su aplicación a medida que se ejecuta, puede detectar los planes que causan regresiones del rendimiento. Cuando detecte regresiones, puede rechazar manualmente o reparar los planes defectuosos con estos pasos:

1. Mientras se ejecute su aplicación, fuerce el uso de planes administrados y añada automáticamente los planes recién descubiertos como no aprobados. Para obtener más información, consulte Uso de los planes administrados de Aurora PostgreSQL y Captura de planes automática.
2. Monitorización de su aplicación en ejecución para detectar regresiones de rendimiento.
3. Cuando descubra una regresión del plan, configure el estado del plan en `rejected`. La próxima vez que el optimizador ejecute la instrucción SQL, ignora automáticamente el plan rechazado y emplea un plan aprobado diferente en su lugar. Para obtener más información, consulte Rechazar o desactivar planes más lentos.

   En algunos casos, puede que prefiera reparar un plan defectuoso en lugar de rechazarlo, deshabilitarlo o eliminarlo. Utilice la extensión `pg_hint_plan` para experimentar con la mejora de un plan. Con `pg_hint_plan`, puede utilizar comentarios especiales para decirle al optimizador que anule cómo crea un plan habitualmente. Para obtener más información, consulte Corrección de planes mediante pg_hint_plan.

Administración de planes de consultas en Aurora PostgreSQL

Con la administración de planes de consultas activada para el clúster de base de datos de Aurora PostgreSQL, el optimizador genera y almacena los planes de ejecución de consultas para cualquier sentencia SQL que procese más de una vez. El optimizador siempre establece el estado del primer plan generado de una instrucción administrada en `Approved` y lo almacena en la vista `dba_plans`.

Un conjunto de planes aprobados para una instrucción administrada se denomina base de referencia del plan. A medida que la aplicación se ejecuta, puede que el optimizador encuentre planes adicionales para las instrucciones administradas. El optimizador establece el estado de los planes adicionales capturados en `Unapproved`.

Posteriormente, podrá decidir si los planes `Unapproved` rinden apropiadamente y cambiarlos a `Approved`, `Rejected` o `Preferred`. Para ello, utilice la función `apg_plan_mgmt.evolve_plan_baselines` o la función `apg_plan_mgmt.set_plan_status`.

Cuando el optimizador genera un plan para una sentencia SQL, la administración de planes de consultas guarda el plan en la tabla `apg_plan_mgmt.plans`. Los usuarios de la base de datos a los que se les ha otorgado el rol de `apg_plan_mgmt`pueden ver los detalles del plan consultando la vista `apg_plan_mgmt.dba_plans`. Por ejemplo, la siguiente consulta muestra detalles de los planes que se encuentran actualmente en la vista para un clúster de base de datos de Aurora PostgreSQL que no es de producción.

- `sql_hash`: identificador de la sentencia SQL que es el valor de hash del texto normalizado de la sentencia SQL.
- `plan_hash`: identificador único del plan que es una combinación del `sql_hash` y un hash del plan.
- `status`: estado del plan. El optimizador puede ejecutar un plan aprobado.
- `enabled`: indica si el plan está listo para usarse (verdadero) o no (falso).
- `plan_outline`: representación del plan que se utiliza para recrear el plan de ejecución real. Los operadores de la estructura de árbol se asignan a los operadores de la salida EXPLAIN.

La vista `apg_plan_mgmt.dba_plans` tiene muchas más columnas que contienen todos los detalles del plan, por ejemplo, cuándo se utilizó el plan por última vez. Para ver todos los detalles, consulte Referencia de la vista apg_plan_mgmt.dba_plans para la edición compatible con Aurora PostgreSQL.

Normalización y el hash SQL

En la vista `apg_plan_mgmt.dba_plans`, puede identificar una instrucción administrada por su valor hash SQL. El hash SQL se calcula sobre una representación normalizada de la instrucción SQL que elimina algunas diferencias, como los valores literales.

El proceso de normalización de cada instrucción SQL conserva el espacio y las mayúsculas, de modo que puede seguir leyendo y entendiendo la esencia de la instrucción SQL. La normalización elimina o reemplaza los siguientes elementos.

- Comentarios del bloque principal
- La palabra clave EXPLAIN y las opciones EXPLAIN y EXPLAIN ANALYZE
- Espacios finales
- Todos los literales

Por ejemplo, tomemos la siguiente instrucción.

```
/*Leading comment*/ EXPLAIN SELECT /* Query 1 */ * FROM t WHERE x > 7 AND y = 1;
```

La administración de planes de consultas estandariza esta instrucción del siguiente modo:

```
SELECT /* Query 1 */ * FROM t WHERE x > CONST AND y = CONST;
```

La normalización permite utilizar el mismo hash SQL para instrucciones SQL similares que pueden diferir únicamente en sus valores literales o de parámetro. En otras palabras, pueden existir varios planes para el mismo hash SQL y un plan diferente que resulte óptimo según diferentes condiciones.

> **Note**
>
> Una sola instrucción SQL que se usa con diferentes esquemas tiene diferentes planes porque está vinculada al esquema específico en tiempo de ejecución. El planificador utiliza las estadísticas de enlace de esquemas para elegir el plan óptimo.

Para obtener más información acerca de cómo el optimizador elige un plan, consulte Uso de los planes administrados de Aurora PostgreSQL. En esa sección, puede aprender a usar `EXPLAIN`

y `EXPLAIN ANALYZE` para previsualizar un plan antes de usarlo realmente. Para obtener más información, consulte Análisis del plan elegido por el optimizador. Para obtener una imagen que describa el proceso de elección de un plan, consulte Cómo elige el optimizador qué plan ejecutar..

Captura de planes de ejecución de Aurora PostgreSQL

La administración de planes de consultas de Aurora PostgreSQL ofrece dos modos diferentes para capturar los planes de ejecución de consultas, automático o manual. Para elegir el modo, defina el valor de `apg_plan_mgmt.capture_plans_baselines` en `automatic` o en `manual`. Puede capturar planes de ejecución para instrucciones SQL específicas mediante una captura del plan manual. También puede capturar todos los planes (o los más lentos) que se pongan en marcha dos o más veces mientras se ejecuta su aplicación utilizando la captura de planes automática.

Al capturar planes, el optimizador establece el estado del primer plan capturado de una instrucción administrada en `approved`. El optimizador establece el estado de cualquier plan adicional capturado para una instrucción administrada en `unapproved`. Sin embargo, puede guardarse ocasionalmente más de un plan con el estado `approved`. Esto puede ocurrir cuando se crean varios planes para una instrucción en paralelo, y antes de que se confirme el primer plan para la instrucción.

Para controlar el número máximo de planes que se pueden capturar y almacenar en la vista `dba_plans`, establezca el parámetro `apg_plan_mgmt.max_plans` en su grupo de parámetros en el nivel de la instancia de la base de datos. Un cambio en el parámetro `apg_plan_mgmt.max_plans` requiere un reinicio de la instancia de base de datos para que el nuevo valor tenga efecto. Para obtener más información, consulte el parámetro apg_plan_mgmt.max_plans.

Captura manual de planes para instrucciones SQL específicas

Si tiene un conjunto conocido de instrucciones SQL para administrar, ponga las instrucciones en un archivo de script SQL y capture manualmente los planes. A continuación se muestra un ejemplo psql de cómo capturar planes de consulta manualmente para un conjunto de instrucciones SQL.

```
psql> SET apg_plan_mgmt.capture_plan_baselines = manual;
psql> \i my-statements.sql
psql> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Tras capturar un plan para cada instrucción SQL, el optimizador añade una nueva fila a la vista `apg_plan_mgmt.dba_plans`.

Recomendamos que utilice instrucciones EXPLAIN o EXPLAIN EXECUTE en el archivo de script de SQL. Asegúrese de que incluye suficientes variaciones en los valores de parámetros para capturar todos los planes de interés.

Si conoce un plan mejor que el plan de costo mínimo del optimizador, puede que sea capaz de forzar al optimizador a que use el plan mejor. Para ello, especifique una o más sugerencias del optimizador. Para obtener más información, consulte Corrección de planes mediante pg_hint_plan. Para comparar el rendimiento de los planes `unapproved` y `approved` y aprobarlos, rechazarlos o eliminarlos, consulte Evaluación del rendimiento de los planes.

Captura de planes automática

Utilice la captura de planes automática para situaciones como la siguiente:

- No sabe las instrucciones SQL específicas que quiere administrar.
- Tiene cientos o miles de instrucciones SQL que administrar.
- Su aplicación usa una API de cliente. Por ejemplo, JDBC utiliza instrucciones preparadas sin nombre o instrucciones en modo masivo que no se pueden expresar en psql.

Para capturar planes automáticamente

1. Active la captura de planes automática configurando `apg_plan_mgmt.capture_plan_baselines` como `automatic` en el grupo de parámetros para la instancia de la base de datos. Para obtener más información, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.
2. A medida que se ejecuta la aplicación, el optimizador captura los planes para cada instrucción SQL que se ejecute al menos dos veces.

   A medida que se ejecuta la aplicación con los ajustes de parámetro del administrador de planes de consulta predeterminados, el optimizador captura los planes para cada instrucción SQL que se ejecute al menos dos veces. La captura de todos los planes con los valores predeterminados tiene una sobrecarga de tiempo de ejecución muy pequeña y puede habilitarse en producción.

Para desactivar la captura de planes automática, realice el siguiente parámetro:

- En el parámetro `apg_plan_mgmt.capture_plan_baselines`, establezca el valor `off` desde el grupo de parámetros de base de datos.

Para medir el rendimiento de los planes sin aprobar y aprobar, rechazar o eliminar dichos planes, consulte Evaluación del rendimiento de los planes.

Uso de los planes administrados de Aurora PostgreSQL

Para que el optimizador use los planes capturados para sus instrucciones administradas, establezca el parámetro `apg_plan_mgmt.use_plan_baselines` en `true`. A continuación figura el ejemplo de una instancia local.

```
SET apg_plan_mgmt.use_plan_baselines = true;
```

Mientras se ejecuta la aplicación, esta configuración hace que el optimizador utilice el plan de costo mínimo, preferido o aprobado que sea válido y esté habilitado para cada instrucción administrada.

Análisis del plan elegido por el optimizador

Cuando el parámetro `apg_plan_mgmt.use_plan_baselines` está establecido en `true`, puede utilizar instrucciones SQL EXPLAIN ANALYZE para hacer que el optimizador muestre el plan que usaría si fuera a ejecutar la consulta. A continuación se muestra un ejemplo.

```
EXPLAIN ANALYZE EXECUTE rangeQuery (1,10000);
```

```
                                                        QUERY PLAN
---------------------------------------------------------------------------------
 Aggregate  (cost=393.29..393.30 rows=1 width=8) (actual time=7.251..7.251 rows=1
 loops=1)
   ->  Index Only Scan using t1_pkey on t1 t  (cost=0.29..368.29 rows=10000 width=0)
 (actual time=0.061..4.859 rows=10000 loops=1)
Index Cond: ((id >= 1) AND (id <= 10000))
         Heap Fetches: 10000
 Planning time: 1.408 ms
 Execution time: 7.291 ms
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1984047223, Plan Hash: 512153379
```

El resultado muestra el plan aprobado a partir de la línea de base que se ejecutaría. Sin embargo, el resultado también muestra que ha encontrado un plan con un costo menor. En ese caso, capture este plan de costo mínimo activando la captura de planes automática, como se describe en Captura de planes automática.

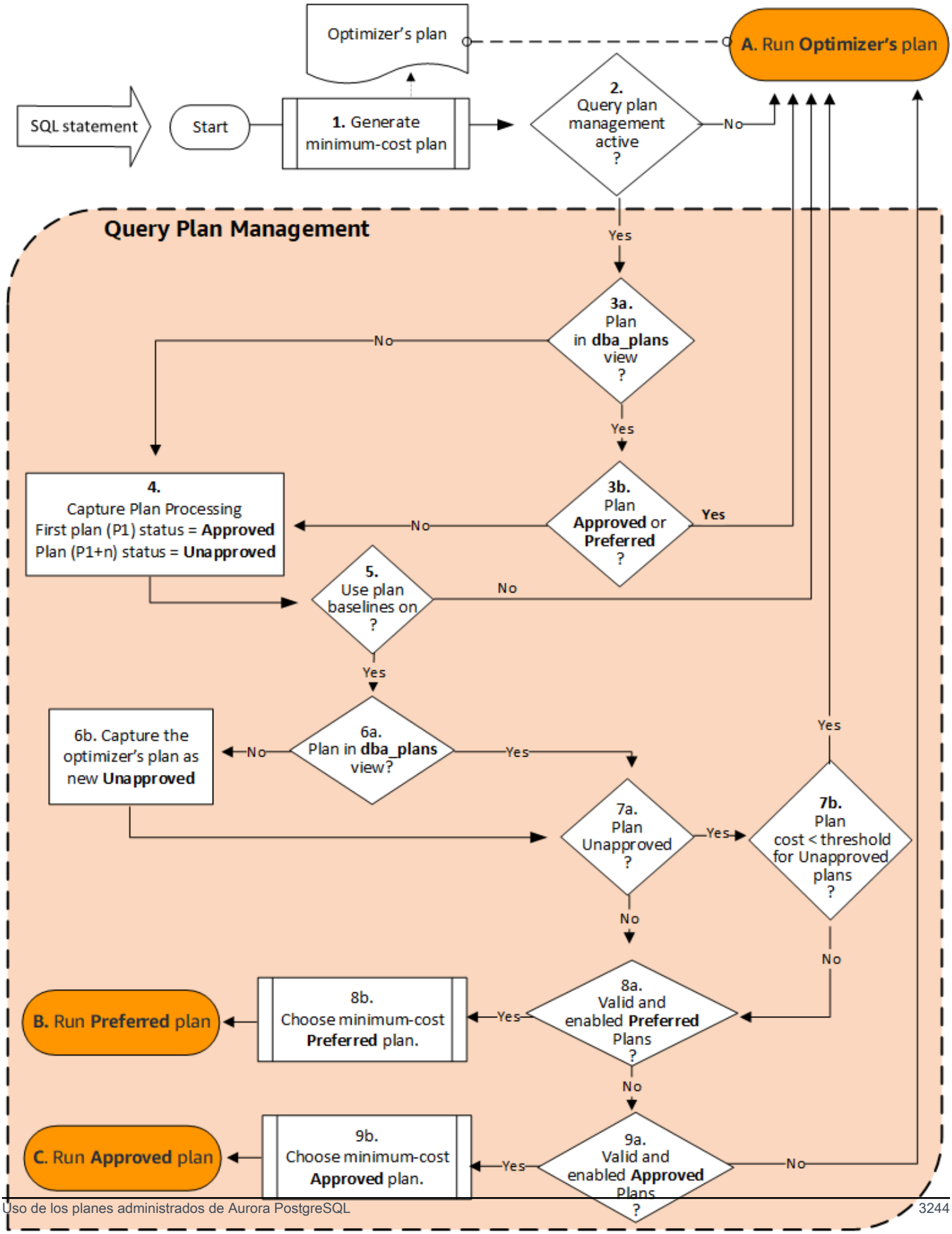
El optimizador siempre captura los nuevos planes como `Unapproved`. Utilice la función `apg_plan_mgmt.evolve_plan_baselines` para comparar planes y cambiarlos a aprobado,

rechazado o deshabilitado. Para obtener más información, consulte Evaluación del rendimiento de los planes.

Cómo elige el optimizador qué plan ejecutar.

El costo de un plan de ejecución es un cálculo que realiza el optimizador para comparar planes distintos. Al calcular el costo de un plan, el optimizador incluye factores tales como las operaciones de CPU y E/S requeridas por ese plan. Para obtener más información acerca de las estimaciones de costos del planificador de consultas de PostgreSQL, consulte el tema sobre planificación de consultas en la documentación de PostgreSQL.

En la siguiente imagen se muestra cómo se elige un plan para una instrucción SQL determinada cuando la administración de planes de consultas está activa y cuándo no.


Optimizer's plan
A. Run Optimizer's plan
SQL statement
Start
1. Generate minimum-cost plan
2. Query plan management active ?
No
Query Plan Management
Yes
3a. Plan in dba_plans view ?
No
Yes
4. Capture Plan Processing
First plan (P1) status = Approved
Plan (P1+n) status = Unapproved
3b. Plan Approved or Preferred ?
No
Yes
5. Use plan baselines on ?
No
Yes
6a. Plan in dba_plans view?
No
Yes
6b. Capture the optimizer's plan as new Unapproved
7a. Plan Unapproved ?
Yes
7b. Plan cost < threshold for Unapproved plans ?
Yes
No
No
8a. Valid and enabled Preferred Plans ?
Yes
8b. Choose minimum-cost Preferred plan.
B. Run Preferred plan
No
9a. Valid and enabled Approved Plans ?
Yes
9b. Choose minimum-cost Approved plan.
C. Run Approved plan
No

El flujo es el siguiente:

1. El optimizador genera un plan de costo mínimo para la instrucción SQL.
2. Si la administración de planes de consultas no está activa, el plan del optimizador se ejecuta inmediatamente (A. Run Optimizer's plan [Ejecutar el plan del optimizador]). La administración de planes de consultas está inactiva cuando los parámetros `apg_plan_mgmt.capture_plan_baselines` y `apg_plan_mgmt.use_plan_baselines` tienen su configuración predeterminada (“off” y “false”, respectivamente).

   De lo contrario, la administración de planes de consultas estará activa. En este caso, la instrucción SQL y el plan del optimizador se evalúan más a fondo antes de elegir un plan.

   > **Tip**
   >
   > Los usuarios de bases de datos con el rol `apg_plan_mgmt` pueden comparar planes de forma proactiva, cambiar el estado de los planes y forzar el uso de planes específicos según sea necesario. Para obtener más información, consulte Mejora de los planes de consultas en Aurora PostgreSQL.

3. Es posible que la instrucción SQL ya tenga planes almacenados previamente por la administración de planes de consultas. Los planes se almacenan en `apg_plan_mgmt.dba_plans`, junto con información sobre las instrucciones SQL que se usaron para crearlas. La información sobre un plan incluye su estado. El estado de un plan puede determinar si se usa o no, de la siguiente manera.
   a. Si el plan no está entre los planes almacenados para la instrucción SQL, significa que es la primera vez que el optimizador genera este plan en particular para la instrucción SQL dada. El plan se envía a Capture Plan Processing (Procesamiento del plan de captura) (4).
   b. Si el plan se encuentra entre los planes almacenados y su estado es Aprobado o Preferido, se ejecuta el plan (A. Ejecutar el plan del optimizador).

      Si el plan se encuentra entre los planes almacenados pero no está Aprobado ni es Preferido, el plan se envía a Procesamiento del plan de captura (4).
4. Cuando se captura un plan por primera vez para una instrucción SQL determinada, el estado del plan siempre se establece en Aprobado (P1). Si el optimizador genera posteriormente el mismo plan para la misma instrucción SQL, el estado de ese plan cambia a No aprobado (P1+n).

   Con el plan capturado y su estado actualizado, la evaluación continúa en el siguiente paso (5).

5. Una base de referencia de un plan consiste en el historial de la instrucción SQL y sus planes en varios estados. La administración de planes de consultas puede tener en cuenta la base de referencia al elegir un plan, en función de si la opción usar base de referencia del plan está activada o no, de la siguiente manera.
   - Usar base de referencia del plan está “off” (desactivado) cuando `apg_plan_mgmt.use_plan_baselines` se establece con su valor predeterminado (`false`). El plan no se compara con la base de referencia antes de ejecutarse (A. Ejecutar el plan del optimizador).
   - Usar la base de referencia del plan está “on” (activado) cuando `apg_plan_mgmt.use_plan_baselines` se establece con su valor predeterminado (`true`). El plan se evalúa más a fondo con la base de referencia (6).
6. El plan se compara con otros planes para la instrucción en la base de referencia.
   a. Si el plan del optimizador se encuentra entre los planes de la base de referencia, se comprueba su estado (7a).
   b. Si el plan del optimizador no está entre los planes de la base de referencia, el plan se agrega a los planes para la instrucción como un plan `Unapproved` nuevo.
7. El estado del plan se verifica para determinar únicamente si no está aprobado.
   a. Si el estado del plan es No aprobado, el costo estimado del plan se compara con el costo estimado especificado para el umbral del plan de ejecución no aprobado.
      - Si el costo estimado del plan está por debajo del umbral, el optimizador lo usa aunque sea un plan no aprobado (A. Ejecutar el plan del optimizador). Por lo general, el optimizador no ejecutará un plan no aprobado. Sin embargo, cuando el parámetro `apg_plan_mgmt.unapproved_plan_execution_threshold` especifica un valor de umbral de costo, el optimizador compara el costo del plan no aprobado con el umbral. Si el costo estimado es de menos del umbral, el optimizador ejecuta el plan. Para obtener más información, consulte apg_plan_mgmt.unapproved_plan_execution_threshold.
      - Si el costo estimado del plan no está por debajo del umbral, se comprueban los demás atributos del plan (8a).
   b. Si el estado del plan no está aprobado, se comprueban sus otros atributos (8a).
8. El optimizador no utilizará un plan desactivado. Es decir, el plan que tenga su atributo `enable` establecido en “f” (false). El optimizador tampoco utilizará un plan que tenga el estado Rechazado.

   El optimizador no puede usar ningún plan que no sea válido. Los planes pueden dejar de ser válidos cuando se eliminan los objetos de los que dependen, como índices o particiones de tabla.

   a. Si la instrucción tiene algún plan preferido habilitado y válido, el optimizador elige el que tenga costo mínimo de entre los planes preferidos almacenados para dicha instrucción SQL. A continuación, el optimizador ejecuta el plan preferido con costo mínimo.
   b. Si el estado de cuenta no tiene ningún plan preferido habilitado y válido, se evalúa en el siguiente paso (9).
9. Si la instrucción tiene algún plan aprobado habilitado y válido, el optimizador elige el que tenga costo mínimo de entre los planes preferidos almacenados para dicha instrucción SQL. A continuación, el optimizador ejecuta el plan aprobado con costo mínimo.

   Si la instrucción no tiene ningún plan aprobado válido y habilitado, el optimizador utiliza el plan de costo mínimo (A. Ejecutar el plan del optimizador).

Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans

Los usuarios y administradores de bases de datos a los que se les ha otorgado el rol `apg_plan_mgmt` pueden ver y administrar los planes almacenados en `apg_plan_mgmt.dba_plans`. El administrador de un clúster de base de datos de Aurora PostgreSQL (alguien con permisos `rds_superuser`) debe conceder explícitamente este rol a los usuarios de la base de datos que tienen que trabajar con la administración de planes de consultas.

La vista `apg_plan_mgmt` contiene el historial del plan de todas las instrucciones SQL administradas para cada base de datos de la instancia de escritor del clúster de base de datos de Aurora PostgreSQL. Esta vista le permite examinar los planes, su estado, cuándo se utilizaron por última vez y todos los demás detalles relevantes.

Como ya hemos abordado en Normalización y el hash SQL, cada plan administrado se identifica mediante un valor hash SQL y un valor hash de plan combinado. Con estos identificadores, puede usar herramientas como Performance Insights de Amazon RDS para seguir el rendimiento individual de los planes. Para obtener más información sobre Información sobre el rendimiento, consulte Using Amazon RDS performance insights.

Descripción de planes administrados.

Para consultar una lista de los planes administrados, utilice una instrucción SELECT en la vista `apg_plan_mgmt.dba_plans`. El siguiente ejemplo muestra algunas columnas en la vista `dba_plans`, como el `status`, que identifica los planes aprobados y no aprobados.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;

 sql_hash   | plan_hash |   status   | enabled | stmt_name
------------+-----------+------------+---------+------------
 1984047223 | 512153379 | Approved   | t       | rangequery
 1984047223 | 512284451 | Unapproved | t       | rangequery
 (2 rows)
```

Para facilitar la lectura, la consulta y el resultado que se muestran incluyen solo algunas de las columnas de la vista `dba_plans`. Para obtener información completa, consulte Referencia de la vista apg_plan_mgmt.dba_plans para la edición compatible con Aurora PostgreSQL.

Mejora de los planes de consultas en Aurora PostgreSQL

Mejore la administración de los planes de consultas evaluando el rendimiento del plan y realizando correcciones. Para obtener más información sobre la mejora de los planes de consultas, vea los siguientes temas.

Temas

- Evaluación del rendimiento de los planes
- Corrección de planes mediante pg_hint_plan

Evaluación del rendimiento de los planes

Después de que el optimizador capture los planes como sin aprobar, utilice la función `apg_plan_mgmt.evolve_plan_baselines` para comparar planes en función de su rendimiento real. Según el resultado de sus experimentos de rendimiento, podrá cambiar el estado de un plan de no aprobado a aprobado o rechazado. También puede decidir utilizar la función `apg_plan_mgmt.evolve_plan_baselines` para deshabilitar temporalmente un plan si no se ajusta a sus requisitos.

Aprobar planes mejores

El siguiente ejemplo muestra cómo cambiar el estado de los planes administrados a aprobados mediante la función `apg_plan_mgmt.evolve_plan_baselines`.

```
SELECT apg_plan_mgmt.evolve_plan_baselines (
```

```
   sql_hash,
   plan_hash,
   min_speedup_factor := 1.0,
   action := 'approve'
)
FROM apg_plan_mgmt.dba_plans WHERE status = 'Unapproved';
```

```
NOTICE:     rangequery (1,10000)
NOTICE:     Baseline   [ Planning time 0.761 ms, Execution time 13.261 ms]
NOTICE:     Baseline+1 [ Planning time 0.204 ms, Execution time 8.956 ms]
NOTICE:     Total time benefit: 4.862 ms, Execution time benefit: 4.305 ms
NOTICE:     Unapproved -> Approved
evolve_plan_baselines
----------------------
0
(1 row)
```

El resultado muestra un informe de rendimiento para la instrucción `rangequery` con vinculaciones de parámetros de 1 y 10 000. El nuevo plan no aprobado (`Baseline+1`) es mejor que el plan aprobado previamente (`Baseline`). Para confirmar que el nuevo plan sea ahora `Approved`, compruebe la vista `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
sql_hash  | plan_hash |  status  | enabled | stmt_name
------------+-----------+----------+---------+------------
1984047223 | 512153379 | Approved | t       | rangequery
1984047223 | 512284451 | Approved | t       | rangequery
(2 rows)
```

El plan administrado incluye ahora dos planes aprobados que componen la base de referencia del plan de la instrucción. También puede llamar a la función `apg_plan_mgmt.set_plan_status` para establecer directamente el campo de estado de un plan en `'Approved'`, `'Rejected'`, `'Unapproved'` o `'Preferred'`.

Rechazar o desactivar planes más lentos

Para rechazar o deshabilitar planes, pase `'reject'` o `'disable'` como parámetro de acción a la función `apg_plan_mgmt.evolve_plan_baselines`. En este ejemplo se deshabilita cualquier

plan capturado `Unapproved` que resulte al menos un 10 por ciento más lento que el mejor plan `Approved` para la instrucción.

```
SELECT apg_plan_mgmt.evolve_plan_baselines(
sql_hash,  -- The managed statement ID
plan_hash, -- The plan ID
1.1,       -- number of times faster the plan must be
'disable'  -- The action to take. This sets the enabled field to false.
)
FROM apg_plan_mgmt.dba_plans
WHERE status = 'Unapproved' AND   -- plan is Unapproved
origin = 'Automatic';       -- plan was auto-captured
```

También puede establecer un plan directamente como rechazado o deshabilitado. Para establecer directamente el campo habilitado de un plan como `true` o `false`, llame a la función `apg_plan_mgmt.set_plan_enabled`. Para establecer directamente el campo de estado de un plan como `'Approved'`, `'Rejected'`, `'Unapproved'` o `'Preferred'`, llame a la función `apg_plan_mgmt.set_plan_status`.

Para eliminar los planes que no son válidos y espera que sigan siendo inválidos, utilice la función `apg_plan_mgmt.validate_plans`. Esta función le permite eliminar o deshabilitar planes no válidos. Para obtener más información, consulte Validación de planes.

Corrección de planes mediante pg_hint_plan

El optimizador de consultas está bien diseñado para encontrar un plan óptimo para todas las instrucciones, y en la mayoría de los casos el optimizador encuentra un plan bueno. Sin embargo, ocasionalmente podría detectar que existe un plan mucho mejor que el generado por el optimizador. Dos formas recomendadas de hacer que el optimizador genere un plan deseado son incluir la extensión `pg_hint_plan` o establecer variables de Grand Unified Configuration (GUC) en PostgreSQL:

- Extensión `pg_hint_plan`: especifique un "consejo" para modificar cómo funciona el planificador mediante la extensión `pg_hint_plan` de PostgreSQL. Para instalar y obtener más información sobre cómo usar la extensión `pg_hint_plan`, consulte la documentación de pg_hint_plan.
- Variables GUC: anule uno o varios parámetros del modelo de costos u otros parámetros del optimizador, como `from_collapse_limit` o `GEQO_threshold`.

Al usar una de estas técnicas para forzar que el optimizador de consultas utilice un plan, también puede utilizar la administración de planes de consulta para capturar y forzar el uso del nuevo plan.

Puede utilizar la extensión `pg_hint_plan` para cambiar el orden de las combinaciones, los métodos de combinación o las rutas de acceso para una instrucción SQL. Puede utilizar un comentario SQL con sintaxis `pg_hint_plan` especial para modificar cómo crea un plan el optimizador. Por ejemplo, supongamos que la instrucción SQL del problema tiene una combinación bidireccional.

```
SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

A continuación, suponga que el optimizador elige el orden de combinación (t1, t2), pero que sabe que el orden de combinación (t2, t1) es más rápido. El siguiente consejo fuerza al optimizador a utilizar el orden de combinación más rápido (t2, t1). Incluya EXPLAIN de modo que el optimizador genere un plan para la instrucción SQL pero sin ejecutar la instrucción. (No se muestra el resultado).

```
/*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

Los siguientes pasos muestran cómo utilizar `pg_hint_plan`.

Para modificar el plan generado por el optimizador y capturar el plan con pg_hint_plan

1. Active el modo de captura manual.

   ```
   SET apg_plan_mgmt.capture_plan_baselines = manual;
   ```

2. Especifique un consejo para la instrucción SQL que le interese.

   ```
   /*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
   FROM t1, t2
   WHERE t1.id = t2.id;
   ```

   Tras la ejecución, el optimizador captura el plan en la vista `apg_plan_mgmt.dba_plans`. El plan capturado no incluye la sintaxis del comentario especial `pg_hint_plan` porque la administración del plan de consultas normaliza la instrucción eliminando los comentarios al principio.

3. Ver los planes administrados utilizando la vista `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, sql_text, plan_outline
FROM apg_plan_mgmt.dba_plans;
```

4. Establezca el estado del plan en `Preferred`. De este modo, se asegurará de que el optimizador decida ejecutarlo en lugar de seleccionarlo del conjunto de planes aprobados cuando el plan de costo mínimo no sea ya `Approved` o `Preferred`.

```
SELECT apg_plan_mgmt.set_plan_status(sql-hash, plan-hash, 'preferred' );
```

5. Desactivar la captura de planes manual y forzar el uso de planes administrados.

```
SET apg_plan_mgmt.capture_plan_baselines = false;
SET apg_plan_mgmt.use_plan_baselines = true;
```

   Ahora, cuando se ejecuta la instrucción SQL original, el optimizador elegirá un plan `Approved` o `Preferred`. Si el plan de costo mínimo no es `Approved` ni `Preferred`, el optimizador elegirá el plan `Preferred`.

Eliminación de planes de consultas en Aurora PostgreSQL

Elimine los planes de ejecución que no esté usando o que no sean válidos. Para obtener más información sobre la eliminación de planes, consulte las siguientes secciones.

Temas

- Eliminación de planes
- Validación de planes

Eliminación de planes

Los planes se eliminan automáticamente si no se usan en más de un mes, específicamente, 32 días. Este es el ajuste predeterminado del parámetro `apg_plan_mgmt.plan_retention_period`. Puede cambiar el período de retención del plan por otro más largo o por un período de tiempo más corto, a partir del valor de 1. Determinar el número de días desde que un plan se usó por última vez se usó restando la fecha de `last_used` de la fecha actual. La fecha de `last_used` es la fecha más

reciente en que el optimizador eligió el plan como plan de costo mínimo o en que se ejecutó el plan. La fecha se almacena para el plan en la vista `apg_plan_mgmt.dba_plans`.

Le recomendamos que elimine planes que no se hayan utilizado durante mucho tiempo o que no resulten útiles. Todos los planes tienen una fecha `last_used` que utiliza el optimizador cada vez que ejecuta un plan o lo elige como plan de costo mínimo para una instrucción. Verifique las últimas fechas de `last_used`para identificar los planes que puede eliminar de forma segura.

La siguiente consulta devuelve una tabla de tres columnas con el recuento del número total de planes, los planes que no se han podido eliminar y los que se han eliminado correctamente. Incluye una consulta anidada que es un ejemplo de cómo usar la función `apg_plan_mgmt.delete_plan` para eliminar todos los planes que no se hayan seleccionado como plan de costo mínimo en los últimos 31 días y cuyo estado no es `Rejected`.

```
SELECT (SELECT COUNT(*) from apg_plan_mgmt.dba_plans) total_plans,
       COUNT(*) FILTER (WHERE result = -1) failed_to_delete,
       COUNT(*) FILTER (WHERE result = 0) successfully_deleted
       FROM (
            SELECT apg_plan_mgmt.delete_plan(sql_hash, plan_hash) as result
            FROM apg_plan_mgmt.dba_plans
            WHERE last_used < (current_date - interval '31 days')
            AND status <> 'Rejected'
            ) as dba_plans ;
```

```
 total_plans | failed_to_delete | successfully_deleted
-------------+------------------+----------------------
           3 |                0 |                    2
```

Para obtener más información, consulte apg_plan_mgmt.delete_plan.

Para eliminar los planes que no son válidos y espera que sigan siendo inválidos, utilice la función `apg_plan_mgmt.validate_plans`. Esta función le permite eliminar o deshabilitar planes no válidos. Para obtener más información, consulte Validación de planes.

> **Important**
>
> Si no elimina los planes extraños, podría quedarse eventualmente sin memoria compartida dedicada a la administración de planes de consulta. Para controlar cuánta memoria tendrá

disponible para los planes administrados, utilice el parámetro `apg_plan_mgmt.max_plans`. Establezca este parámetro en el grupo de parámetros de base de datos personalizado y reinicie la instancia de base de datos para que los cambios surtan efecto. Para obtener más información, consulte el parámetro apg_plan_mgmt.max_plans.

Validación de planes

Use la función `apg_plan_mgmt.validate_plans` para eliminar o deshabilitar planes no válidos.

Los planes pueden volverse no válidos u obsoletos cuando se eliminan los objetos de los que dependen, como un índice o una tabla. Sin embargo, puede que un plan deje de ser válido solo temporalmente si el objeto eliminado se recrea. Si un plan no válido puede pasar a ser válido posteriormente, es posible que prefiera deshabilitar un plan no válido o no hacer nada en lugar de eliminarlo.

Para encontrar y eliminar todos los planes que no sean válidos y no se hayan utilizado en la última semana, utilice la función `apg_plan_mgmt.validate_plans` de la siguiente forma.

```
SELECT apg_plan_mgmt.validate_plans(sql_hash, plan_hash, 'delete')
FROM apg_plan_mgmt.dba_plans
WHERE last_used < (current_date - interval '7 days');
```

Para habilitar o deshabilitar un plan directamente, utilice la función `apg_plan_mgmt.set_plan_enabled`.

Importación y exportación de planes administrados para Aurora PostgreSQL

Puede exportar sus planes administrados e importarlos en otra instancia de base de datos.

Para exportar planes administrados.

Un usuario autorizado puede copiar cualquier subconjunto de la tabla `apg_plan_mgmt.plans` a otra tabla, y después guardarlo mediante el comando pg_dump. A continuación se muestra un ejemplo.

```
CREATE TABLE plans_copy AS SELECT *
```

```
FROM apg_plan_mgmt.plans [ WHERE predicates ] ;
```

```
% pg_dump --table apg_plan_mgmt.plans_copy -Ft mysourcedatabase > plans_copy.tar
```

```
DROP TABLE apg_plan_mgmt.plans_copy;
```

Para importar planes administrados.

1. Copie el archivo .tar de los planes administrados exportados al sistema en el que desee restaurar los planes.
2. Utilice el comando `pg_restore` para copiar el archivo tar en una nueva tabla.

   ```
   % pg_restore --dbname mytargetdatabase -Ft plans_copy.tar
   ```

3. Combine la tabla `plans_copy` con la tabla `apg_plan_mgmt.plans`, como se muestra en el siguiente ejemplo.

   > **Note**
   >
   > En algunos casos, puede volcar de una versión de la extensión `apg_plan_mgmt` y restaurar a una versión diferente. En estos casos, las columnas de la tabla de planes puede ser diferente. De ser así, ponga un nombre explícito a las columnas en lugar de usar SELECT *.

   ```
   INSERT INTO apg_plan_mgmt.plans SELECT * FROM plans_copy
    ON CONFLICT ON CONSTRAINT plans_pkey
    DO UPDATE SET
    status = EXCLUDED.status,
    enabled = EXCLUDED.enabled,
    -- Save the most recent last_used date
    --
    last_used = CASE WHEN EXCLUDED.last_used > plans.last_used
    THEN EXCLUDED.last_used ELSE plans.last_used END,
    -- Save statistics gathered by evolve_plan_baselines, if it ran:
    --
    estimated_startup_cost = EXCLUDED.estimated_startup_cost,
    estimated_total_cost = EXCLUDED.estimated_total_cost,
    planning_time_ms = EXCLUDED.planning_time_ms,
   ```

```
execution_time_ms = EXCLUDED.execution_time_ms,
total_time_benefit_ms = EXCLUDED.total_time_benefit_ms,
execution_time_benefit_ms = EXCLUDED.execution_time_benefit_ms;
```

4. Vuelva a cargar los planes administrados en la memoria compartida y elimine la tabla de planes temporal.

```
SELECT apg_plan_mgmt.reload(); -- refresh shared memory
DROP TABLE plans_copy;
```

Referencia de parámetros para la administración de planes de consultas de Aurora PostgreSQL

Puede configurar sus preferencias para la extensión `apg_plan_mgmt` mediante los parámetros que se indican en esta sección. Están disponibles en el parámetro de clúster de base de datos personalizado y en el grupo de parámetros de base de datos asociado al clúster de base de datos de Aurora PostgreSQL. Estos parámetros controlan el comportamiento de la función de administración de planes de consultas y cómo afecta al optimizador. Para obtener más información sobre la administración de planes de consultas, consulte Activación de la administración de planes de consultas en Aurora PostgreSQL. El cambio de los siguientes parámetros no tiene ningún efecto si la extensión `apg_plan_mgmt` no está configurada tal como se detalla en esa sección. Para obtener información acerca de cómo modificar los parámetros, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora y Grupos de parámetros de base de datos para instancias de Amazon Aurora.

Parámetros

- apg_plan_mgmt.capture_plan_baselines
- apg_plan_mgmt.plan_capture_threshold
- apg_plan_mgmt.explain_hashes
- apg_plan_mgmt.log_plan_enforcement_result
- apg_plan_mgmt.max_databases
- apg_plan_mgmt.max_plans
- apg_plan_mgmt.plan_hash_version
- apg_plan_mgmt.plan_retention_period
- apg_plan_mgmt.unapproved_plan_execution_threshold

- apg_plan_mgmt.use_plan_baselines
- auto_explain.hashes

apg_plan_mgmt.capture_plan_baselines

Captura los planes de ejecución de consultas generados por el optimizador para cada instrucción SQL y los almacena en la vista `dba_plans`. De forma predeterminada, la cantidad máxima de planes que se puede almacenar es de 10 000, según lo especificado en el parámetro `apg_plan_mgmt.max_plans`. Para obtener información de referencia, consulte apg_plan_mgmt.max_plans.

Puede establecer este parámetro en el grupo de parámetros de clúster de base de datos personalizado o en el grupo de parámetros de base de datos personalizado. Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
off	automático	Aplique esta configuración en nivel de sesión o en un grupo de parámetros para capturar los planes que se utilizan dos o más veces.
	manual	Aplique esta configuración en nivel de sesión o en un grupo de parámetros para capturar los planes que se utilizan una o más veces.
	off	Desactiva la captura de planes.

Para obtener más información, consulte Captura de planes de ejecución de Aurora PostgreSQL.

apg_plan_mgmt.plan_capture_threshold

Especifica el límite de modo que si el coste total del plan de ejecución de consultas es inferior a dicho límite, el plan no se incluirá en la vista `apg_plan_mgmt.dba_plans`.

Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
0	0 - 1.79769e+308	Establece el límite del coste total de ejecución del plan de consultas `apg_plan_mgmt` para incluir los planes.

Para obtener más información, consulte Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans.

apg_plan_mgmt.explain_hashes

Especifica si `EXPLAIN [ANALYZE]` muestra `sql_hash` y `plan_hash` al final de su salida. Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
0	0 (off)	EXPLAIN no muestra la opción true de sql_hash y plan_hash sin hashes.
	1 (on)	EXPLAIN muestra la opción true de sql_hash y plan_hash sin hashes.

apg_plan_mgmt.log_plan_enforcement_result

Especifica si los resultados deben registrarse para ver si los planes administrados por QPM se utilizan correctamente. Cuando se utiliza un plan genérico almacenado, no se escribirán registros en los archivos de registro. Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
none	none	No muestra ningún resultado de aplicación del plan en los archivos de registro.

Predeterminado/a	Valores permitidos	Descripción
	on_error	Solo muestra el resultado de la aplicación del plan en los archivos de registro cuando QPM no utiliza los planes administrados.
	all	Muestra todos los resultados de la aplicación del plan en archivos de registro, incluidos los correctos y los incorrectos.

apg_plan_mgmt.max_databases

Especifica el número máximo de bases de datos de la instancia de escritor del clúster de base de datos de Aurora PostgreSQL que pueden usar la administración de planes de consultas. De forma predeterminada, la administración de planes de consultas puede admitir 10 bases de datos. Si tiene más de 10 bases de datos en la instancia, puede cambiar el valor de esta configuración. Para saber cuántas bases de datos hay en una instancia determinada, conéctese a la instancia mediante `psql`. A continuación, utilice el metacomando psql, `\l`, para enumerar las bases de datos.

Para cambiar el valor de este parámetro, es necesario reiniciar la instancia para que la configuración surta efecto.

Predeterminado/a	Valores permitidos	Descripción
10	10-2147483647	Número máximo de bases de datos que pueden usar la administración de planes de consultas en la instancia.

Puede establecer este parámetro en el grupo de parámetros de clúster de base de datos personalizado o en el grupo de parámetros de base de datos personalizado.

apg_plan_mgmt.max_plans

Establece el número máximo de instrucciones SQL que el administrador del plan de consultas puede mantener en la vista `apg_plan_mgmt.dba_plans`. Recomendamos establecer este parámetro en `10000` o superior para todas las versiones de Aurora PostgreSQL.

Puede establecer este parámetro en el grupo de parámetros de clúster de base de datos personalizado o en el grupo de parámetros de base de datos personalizado. Para cambiar el valor de este parámetro, es necesario reiniciar la instancia para que la configuración surta efecto.

Predeterminado/a	Valores permitidos	Descripción
10000	10-2147483647	Número máximo de planes que se pueden almacenar en la vista `apg_plan_mgmt.dba_plans` . El valor predeterminado para la versión 10 de Aurora PostgreSQL y las versiones previas es 1000.

Para obtener más información, consulte Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans.

apg_plan_mgmt.plan_hash_version

Especifica los casos de uso para los que está diseñado el cálculo plan_hash. Una versión superior de `apg_plan_mgmt.plan_hash_version` abarca todas las funciones de la versión inferior. Por ejemplo, la versión 3 abarca los casos de uso admitidos por la versión 2.

El cambio del valor de este parámetro debe ir seguido de una llamada a `apg_plan_mgmt.validate_plans('update_plan_hash')`. Actualiza los valores de plan_hash de cada base de datos con apg_plan_mgmt instalado y las entradas de la tabla de planes. Para obtener más información, consulte Validación de planes

Predeterminado/a	Valores permitidos	Descripción
1	1	Cálculo de plan_hash predeterminado.
	2	Se modificó el cálculo de plan_hash para admitir múltiples esquemas.
	3	Se modificó el cálculo de plan_hash para admitir múltiples esquemas y tablas particionadas.

Predeterminado/a	Valores permitidos	Descripción
	4	Cálculo de plan_hash modificado para operadores paralelos y para admitir nodos materializados.

apg_plan_mgmt.plan_retention_period

Especifica el número de días que se van a conservar planes en la vista `apg_plan_mgmt.dba_plans`, tras el cual se eliminan automáticamente. De forma predeterminada, un plan se elimina cuando han transcurrido 32 días desde la última vez que se usó (la columna `last_used` en la vista `apg_plan_mgmt.dba_plans`). Puede cambiar esta configuración a cualquier número, de 1 o más.

Para cambiar el valor de este parámetro, es necesario reiniciar la instancia para que la configuración surta efecto.

Predeterminado/a	Valores permitidos	Descripción
32	1-2147483647	Número máximo de días desde que un plan se usó por última vez antes de que se elimine.

Para obtener más información, consulte Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans.

apg_plan_mgmt.unapproved_plan_execution_threshold

Especifica un límite de coste por debajo del cual el optimizador puede utilizar un plan no aprobado. De forma predeterminada, el límite es 0, por lo que el optimizador no ejecuta planes sin aprobar. Si se establece este parámetro en un límite relativamente bajo, como 100, se evita la sobrecarga de los planes poco importantes. También puede establecer este parámetro en un valor extremadamente alto, como 10000000, utilizando el estilo reactivo de gestión de planes. Esto permite al optimizador utilizar todos los planes seleccionados sin sobrecargar el plan. Pero si encuentra un plan incorrecto, puede marcarlo manualmente como "rechazado" para que no se utilice la próxima vez.

El valor de este parámetro representa una estimación de costos para ejecutar un plan determinado. Si un plan no aprobado está por debajo de ese costo estimado, el optimizador lo usa para la instrucción SQL. Puedes ver los planes capturados y su estado (Aprobado, No aprobado) en la vista `dba_plans`. Para obtener más información, consulte Examinación de los planes de consultas de Aurora PostgreSQL en la vista dba_plans.

Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
0	0-2147483647	Costo estimado del plan por debajo del cual se usa un plan no aprobado.

Para obtener más información, consulte Uso de los planes administrados de Aurora PostgreSQL.

apg_plan_mgmt.use_plan_baselines

Especifica que el optimizador debe usar uno de los planes aprobados capturados y almacenados en la vista `apg_plan_mgmt.dba_plans`. De forma predeterminada, este parámetro está desactivado (false), lo que provoca que el optimizador utilice el plan de menor costo que genera sin ninguna evaluación adicional. Al activar este parámetro (configurándolo en true) se fuerza al optimizador a elegir un plan de ejecución de consultas para la instrucción a partir de la línea base del plan. Para obtener más información, consulte Uso de los planes administrados de Aurora PostgreSQL. Para ver una imagen que detalle este proceso, consulte Cómo elige el optimizador qué plan ejecutar..

Puede establecer este parámetro en el grupo de parámetros del clúster de base de datos personalizado o en el grupo de parámetros de base de datos personalizado. Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
false	true	Use un plan aprobado, preferido o no aprobado del `apg_plan_mgmt.dba_plans` . Si ninguno de ellos cumple con todos los criterios de evaluación para el optimizador, puede usar su propio

Predeterminado/a	Valores permitidos	Descripción
		plan de costo mínimo generado. Para obtener más información, consulte Cómo elige el optimizador qué plan ejecutar..
	false	Use el plan de costo mínimo generado por el optimizador.

Puede evaluar los tiempos de respuesta de los diferentes planes capturados y cambiar el estado del plan, según sea necesario. Para obtener más información, consulte Mejora de los planes de consultas en Aurora PostgreSQL.

auto_explain.hashes

Especifica si la salida auto_explain muestra sql_hash y plan_hash. Para cambiar el valor de este parámetro no es necesario reiniciar.

Predeterminado/a	Valores permitidos	Descripción
0(off)	0(off)	El resultado `auto_explain` no muestra `sql_hash` y `plan_hash` .
	1(on)	El resultado `auto_explain` muestra `sql_hash` y `plan_hash` .

Referencia de funciones para la administración de planes de consultas de Aurora PostgreSQL

La extensión `apg_plan_mgmt` proporciona las siguientes funciones.

Funciones

- apg_plan_mgmt.copy_outline
- apg_plan_mgmt.delete_plan
- apg_plan_mgmt.evolve_plan_baselines
- apg_plan_mgmt.get_explain_plan

- apg_plan_mgmt.plan_last_used
- apg_plan_mgmt.reload
- apg_plan_mgmt.set_plan_enabled
- apg_plan_mgmt.set_plan_status
- apg_plan_mgmt.update_plans_last_used
- apg_plan_mgmt.validate_plans

apg_plan_mgmt.copy_outline

Copia un hash del plan SQL y un esquema del plan determinados en un hash del plan SQL y un esquema del plan de destino, por lo que sobrescribe el hash y el esquema del plan de destino. Esta función está disponible en las versiones 2.3 y posteriores de `apg_plan_mgmt`.

Sintaxis

```
apg_plan_mgmt.copy_outline(
    source_sql_hash,
    source_plan_hash,
    target_sql_hash,
    target_plan_hash,
    force_update_target_plan_hash
)
```

Valor devuelto

Devuelve 0 si la copia se ha realizado correctamente. Genera excepciones para entradas no válidas.

Parámetros

Parámetro	Descripción
`source_sql_hash`	El identificador del `sql_hash` asociado al `plan_hash` para copiar la consulta de destino.
`source_plan_hash`	El identificador del `plan_hash` que se va a copiar a la consulta de destino.

Parámetro	Descripción
`target_sql_hash`	El identificador del `sql_hash` de la consulta que se va a actualizar con el hash y el esquema del plan de origen.
`target_plan_hash`	El identificador del `plan_hash` de la consulta que se va a actualizar con el hash y el esquema del plan de origen.
`force_update_target_plan_hash`	(Opcional) El ID `target_plan_hash` de la consulta se actualiza incluso si el plan de origen no es reproducible para el`target_sql_hash`. Si se establece en true, la función se puede utilizar para copiar planes de varios esquemas en los que los nombres de las relaciones y las columnas sean coherentes.

Notas de uso

Esta función le permite copiar un hash de plan y un esquema de plan que utiliza sugerencias a otras instrucciones similares, y le evita así tener que utilizar instrucciones de sugerencias en línea en cada ocurrencia en las instrucciones de destino. Si la consulta de destino actualizada da como resultado un plan no válido, esta función genera un error y revierte el intento de actualización.

apg_plan_mgmt.delete_plan

Eliminar un plan administrado.

Sintaxis

```
apg_plan_mgmt.delete_plan(
    sql_hash,
    plan_hash
)
```

Valor devuelto

Devuelve 0 si la eliminación se ha realizado correctamente o -1 si en la eliminación se ha producido un error.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado.

apg_plan_mgmt.evolve_plan_baselines

Comprueba si un plan ya aprobado es más rápido o si un plan identificado por el optimizador de consultas como plan de costo mínimo es más rápido.

Sintaxis

```
apg_plan_mgmt.evolve_plan_baselines(
    sql_hash,
    plan_hash,
    min_speedup_factor,
    action
)
```

Valor devuelto

El número de planes que no han sido más rápidos que el mejor plan aprobado.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado. Utilice NULL para referirse a todos los planes que tengan el mismo valor del ID de `sql_hash`.

Parámetro	Descripción
`min_speedup_factor`	El factor de aceleración mínimo es el número de veces más rápido que debe ser un plan en relación con los planes ya aprobados para aprobarlo . Este factor puede ser también el número de veces más lento que debe ser un plan para rechazarlo o deshabilitarlo. Este valor es un número flotante positivo.
`action`	La acción que va a realizar la función. Entre los valores válidos se incluyen los siguientes. No distingue entre mayúsculas y minúsculas. • `'disable'` : deshabilitar cada plan coincidente que no cumpla el factor de aceleración mínimo. • `'approve'` – habilitar cada plan coincidente que cumpla el factor de aceleración mínimo y establecer su estado en `approved`. • `'reject'` – para cada plan coincidente que no cumpla el factor de aceleración mínimo, establecer su estado en `rejected`. • NULL: la función se limita a devolver el número de planes que no tienen beneficios de rendimiento porque no cumplen el factor de aceleración mínimo.

Notas de uso

Establecer planes específicos como aprobados, rechazados o deshabilitados en función de si el tiempo de planificación más el de ejecución es más rápido que el mejor plan aprobado por un factor que puede configurar. El parámetro de acción puede establecerse en `'approve'` o `'reject'` para aprobar o rechazar automáticamente un plan que cumpla los criterios de rendimiento. Como alternativa, podría establecerse como " (cadena vacía) para realizar el experimento de rendimiento y producir un informe, sin realizar ninguna acción.

Puede evitar una nueva ejecución sin sentido de la función `apg_plan_mgmt.evolve_plan_baselines` para un plan en el que se ha ejecutado recientemente. Para ello, restrinja los planes a los planes sin aprobar creados recientemente. Como alternativa, puede evitar la ejecución de la función `apg_plan_mgmt.evolve_plan_baselines` en cualquier plan aprobado que tenga una marca temporal `last_verified` reciente.

Realizar un experimento de rendimiento para comparar el tiempo de planificación más el de ejecución para cada plan en relación con los demás planes de la base de referencia. En algunos casos, solo hay un plan para una instrucción, y el plan está aprobado. En tal caso, compare el tiempo de planificación más ejecución del plan con el tiempo de planificación más ejecución de no usar ningún plan.

El beneficio (o desventaja) incremental de cada plan queda registrado en la vista `apg_plan_mgmt.dba_plans` de la columna `total_time_benefit_ms`. Si este valor es positivo, existe un beneficio de rendimiento medible al incluir este plan en la base de referencia.

Además de recopilar el tiempo de planificación y ejecución de cada plan candidato, la columna `last_verified` de la vista `apg_plan_mgmt.dba_plans` se actualiza con el `current_timestamp`. La marca temporal `last_verified` se podría utilizar para evitar la ejecución de esta función de nuevo en un plan cuyo rendimiento se haya verificado recientemente.

apg_plan_mgmt.get_explain_plan

Genera el texto de una instrucción `EXPLAIN` para la instrucción SQL especificada.

Sintaxis

```
apg_plan_mgmt.get_explain_plan(
    sql_hash,
    plan_hash,
    [explainOptionList]
)
```

Valor devuelto

Devuelve estadísticas de tiempo de ejecución para las instrucciones SQL especificadas. Utilizar sin `explainOptionList` para devolver un plan `EXPLAIN` simple.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado.

Parámetro	Descripción
`explainOptionList`	Una lista separada por comas de opciones de explicación. Los valores válidos son `'analyze'` , `'verbose'` , `'buffers'` , `'hashes'` y `'format json'`. Si la lista de `explainOptionList` es NULL o una cadena vacía (''), esta función genera una instrucción `EXPLAIN`, sin ninguna estadística.

Notas de uso

Para `explainOptionList`, puede usar cualquiera de las mismas opciones que usaría con una instrucción `EXPLAIN`. El optimizador de Aurora PostgreSQL concatena la lista de opciones que proporciona a la instrucción `EXPLAIN`.

apg_plan_mgmt.plan_last_used

Devuelve la fecha `last_used` del plan especificado de la memoria compartida.

> **Note**
>
> El valor de la memoria compartida siempre es actual en la instancia de base de datos principal del clúster de base de datos. El valor solo se vacía periódicamente en la columna `last_used` de la vista `apg_plan_mgmt.dba_plans`.

Sintaxis

```
apg_plan_mgmt.plan_last_used(
    sql_hash,
    plan_hash
)
```

Valor devuelto

Devuelve la fecha `last_used`.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado.

apg_plan_mgmt.reload

Vuelve a cargar los planes en la memoria compartida desde la vista `apg_plan_mgmt.dba_plans`.

Sintaxis

```
apg_plan_mgmt.reload()
```

Valor devuelto

Ninguno.

Parámetros

Ninguna.

Notas de uso

Llame a `reload` para las siguientes situaciones:

- Utilícelo para actualizar la memoria compartida de una réplica de solo lectura inmediatamente, en lugar de esperar a que los nuevos planes propaguen la réplica.
- Se utiliza tras importar los planes administrados.

apg_plan_mgmt.set_plan_enabled

Habilitar o deshabilitar un plan administrado.

Sintaxis

```
apg_plan_mgmt.set_plan_enabled(
    sql_hash,
    plan_hash,
    [true | false]
)
```

Valor devuelto

Devuelve 0 si el ajuste se ha realizado correctamente o -1 si en el ajuste se ha producido un error.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado.
`enabled`	Valor booleano de verdadero o falso: • Un valor de `true` habilita el plan. • Un valor de `false` deshabilita el plan.

apg_plan_mgmt.set_plan_status

Establezca el estado de un plan administrado en `Approved`, `Unapproved`, `Rejected` o `Preferred`.

Sintaxis

```
apg_plan_mgmt.set_plan_status(
    sql_hash,
    plan_hash,
    status
)
```

Valor devuelto

Devuelve 0 si el ajuste se ha realizado correctamente o -1 si en el ajuste se ha producido un error.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado.
`status`	Cadena con uno de los siguientes valores: • `'Approved'` • `'Unapproved'` • `'Rejected'` • `'Preferred'` El caso que utilice no importa, pero el valor de estado se establece en mayúscula inicial en la vista `apg_plan_mgmt.dba_plans` . Para obtener más informaci ón acerca de estos valores de la tarea, consulte `status` en Referencia de la vista apg_plan_mgmt.dba_plans para la edición compatible con Aurora PostgreSQL.

apg_plan_mgmt.update_plans_last_used

Actualiza inmediatamente la tabla de planes con la fecha de `last_used` almacenada en la memoria compartida.

Sintaxis

```
apg_plan_mgmt.update_plans_last_used()
```

Valor devuelto

Ninguno.

Parámetros

Ninguna.

Notas de uso

Llame a `update_plans_last_used` para asegurarse de que las consultas de la columna `dba_plans.last_used` utilizan la información más actualizada. Si el archivo de la fecha de `last_used` no se actualiza inmediatamente, un proceso en segundo plano actualiza la tabla de planes con la fecha de `last_used` una vez cada hora (de forma predeterminada).

Por ejemplo, si una instrucción con un `sql_hash` determinado comienza a ejecutarse lentamente, puede determinar qué planes para esa instrucción se ejecutaron desde que comenzó la regresión de rendimiento. Para ello, primero vacíe los datos de la memoria compartida al disco para que las fechas de `last_used` estén actualizadas y, a continuación, consulte todos los planes de `sql_hash` de la instrucción con la regresión del rendimiento. En la consulta, asegúrese de que la fecha de `last_used` es superior o igual que la fecha en que comenzó la regresión del rendimiento. La consulta identifica el plan o conjunto de planes que podrían ser responsables de la regresión del rendimiento. Puede usar `apg_plan_mgmt.get_explain_plan` con `explainOptionList` establecidos en `verbose, hashes`. También puede utilizar `apg_plan_mgmt.evolve_plan_baselines` para analizar el plan y cualquier plan alternativo que pueda funcionar mejor.

La función `update_plans_last_used` tiene efecto únicamente en la instancia de base de datos principal del clúster de base de datos.

apg_plan_mgmt.validate_plans

Validar que el optimizador aún puede recrear planes. El optimizador valida los planes `Approved`, `Unapproved` y `Preferred`, aunque el plan esté habilitado o deshabilitado. Los planes `Rejected` no se validan. Opcionalmente, puede usar la función `apg_plan_mgmt.validate_plans` para eliminar o deshabilitar planes no válidos.

Sintaxis

```
apg_plan_mgmt.validate_plans(
    sql_hash,
    plan_hash,
    action)
```

```
apg_plan_mgmt.validate_plans(
    action)
```

Valor devuelto

Número de planes no válidos.

Parámetros

Parámetro	Descripción
`sql_hash`	El ID `sql_hash` de la instrucción SQL administrada por el plan.
`plan_hash`	El ID `plan_hash` del plan administrado. Utilice NULL para referirse a todos los planes para el mismo valor del ID de `sql_hash`.
`action`	La acción que va a realizar la función para los planes no válidos. Entre los valores de cadena válidos se incluyen los siguientes. No distingue entre mayúsculas y minúsculas. • `'disable'` : los planes no válidos se deshabilitan. • `'delete'`: los planes no válidos se eliminan. • `'update_plan_hash'` – Actualiza el `plan_hash` ID de los planes que no se pueden reproducir exactamente. También le permite corregir un plan al reescribir el SQL. A continuación, puede registrar el plan en buen estado como un plan `Approved` para el SQL original. • NULL: la función se limita a devolver el número de planes no válidos. No se realiza ninguna otra acción. • '': una cadena vacía produce un mensaje que indica el número de planes válidos y no válidos. Cualquier otro valor se trata como una cadena vacía.

Notas de uso

Utilice el formulario `validate_plans(action)` para validar todos los planes administrados para las instrucciones administradas en toda la vista `apg_plan_mgmt.dba_plans`.

Utilice el formulario `validate_plans(sql_hash, plan_hash, action)` para validar un plan administrado especificado con `plan_hash`, para una instrucción administrada especificada con `sql_hash`.

Utilice el formulario `validate_plans(sql_hash, NULL, action)` para validar todos los planes administrados para una instrucción administrada especificada con `sql_hash`.

Referencia de la vista apg_plan_mgmt.dba_plans para la edición compatible con Aurora PostgreSQL

Las columnas de la información del plan en la vista `apg_plan_mgmt.dba_plans` incluyen lo siguiente.

Columna dba_plans	Descripción
`cardinality_error`	Medición del error entre la cardinalidad estimada contra la cardinalidad real. La cardinalidad es el número de filas de la tabla que procesará el plan. Si el error de cardinalidad es grande, aumenta la probabilidad de que el plan no resulte óptimo. Esta columna se rellena mediante la función apg_plan_mgmt.evolve_plan_baselines.
`compatibility_level`	Este parámetro muestra cuándo se validó por última vez un plan de consultas. En las versiones 12.19, 13.15, 14.12, 15.7, 16.3 y posteriores de Aurora PostgreSQL, se muestra el número de versión de Aurora. Para versiones anteriores, muestra un número de versión específico de la característica. **Note** Mantenga este valor de parámetro en su valor predeterminado. Aurora PostgreSQL establece y actualiza automáticamente este valor.
`created_by`	El usuario autenticado (`session_user`) que ha creado el plan.

Columna dba_plans	Descripción
`enabled`	Un indicador de si el plan está habilitado o deshabilitado. Todos los planes están habilitados de forma predeterminada. Puede deshabilitar los planes para evitar que el optimizador los use. Para modificar este valor, use la función apg_plan_mgmt.set_plan_enabled.
`environment_variables`	Los parámetros y valores Grand Unified Configuration (GUC) de PostgreSQL que ha anulado el optimizador en el momento de capturar el plan.
`estimated_startup_cost`	El costo de configuración del optimizador estimado antes de que entregue las filas de una tabla.
`estimated_total_cost`	El costo de configuración del optimizador estimado para entregar la última fila de una tabla.
`execution_time_benefit_ms`	El beneficio en tiempo de ejecución, en milisegundos, de habilitar el plan. Esta columna se rellena mediante la función apg_plan_mgmt.evolve_plan_baselines.
`execution_time_ms`	El tiempo estimado, en milisegundos, que se ejecutaría el plan. Esta columna se rellena mediante la función apg_plan_mgmt.evolve_plan_baselines.
`has_side_effects`	Un valor que indica que la instrucción SQL es una instrucción en lenguaje de manipulación de datos (DML) o una instrucción SELECT que contiene una función VOLATILE.

Columna dba_plans	Descripción
`last_used`	Este valor se actualiza a la fecha actual siempre que el plan sea ejecutado o cuando el plan sea el plan de costo mínimo del optimizador de consultas. Este valor se almacena en la memoria compartida y se vacía en el disco periódicamente. Para obtener el valor más actualizado, lea la fecha de la memoria compartid a llamando a la función `apg_plan_mgmt.plan_last_use d(sql_hash, plan_hash)` en lugar de leer el valor `last_used` . Para obtener más información, consulte el parámetro apg_plan_mgmt.plan_retention_period.
`last_validated`	La fecha y hora más recientes en las que se comprobó que el plan se podría recrear mediante la función apg_plan_mgmt.vali date_plans o la función apg_plan_mgmt.evolve_plan_baselines.
`last_verified`	La fecha y hora más recientes en las que se verificó un plan como el que mejor rendimiento ofrece para los parámetro s especificados mediante la función apg_plan_mgmt.evol ve_plan_baselines.
`origin`	Cómo se capturó el plan con el parámetro apg_plan_mgmt.capt ure_plan_baselines. Entre los valores válidos se incluyen los siguientes: `M`: el plan se capturó mediante captura de planes manual. `A`: el plan se capturó mediante captura de planes automática.
`param_list`	Los valores de parámetros que se pasaron a la instrucción si es una instrucción preparada.
`plan_created`	La fecha y hora en que se creó el plan.
`plan_hash`	El identificador del plan. La combinación de `plan_hash` y `sql_hash` identifica un plan específico de forma unívoca.

Columna dba_plans	Descripción
`plan_outline`	Una representación del plan que se utiliza para recrear el plan de ejecución real, y es independiente de la base de datos. Los operadores del árbol se corresponden con los operadores que aparecen en el resultado de EXPLAIN.
`planning_time_ms`	El tiempo real de ejecución del planificador, en milisegundos. Esta columna se rellena mediante la función apg_plan_mgmt.evolve_plan_baselines.
`queryId`	Un hash de instrucción, calculado por la extensión `pg_stat_statements`. No es un identificador estable ni independiente de la base de datos, ya que depende de los identificadores de objetos (OID). El valor será `0` si `compute_query_id` está `off` al capturar el plan de consulta.
`sql_hash`	Un valor hash del texto de la instrucción SQL, normalizado con los literales eliminados.
`sql_text`	El texto completo de la instrucción SQL.

Columna dba_plans	Descripción
`status`	El estado de un plan, que determina cómo utiliza un plan el optimizador. Entre los valores válidos se incluyen los siguientes. • `Approved`: un plan utilizable que el optimizador puede elegir ejecutar. El optimizador ejecuta el plan menos costoso a partir de un conjunto de planes aprobados para una instrucción administrada (base de referencia). Para restaurar un plan como aprobado, utilice la función apg_plan_mgmt.evolve_plan_baselines. • `Unapproved` : un plan capturado que no se ha verificado o para su uso. Para obtener más información, consulte Evaluación del rendimiento de los planes. • `Rejected`: un plan que el optimizador no utilizará. Para obtener más información, consulte Rechazar o desactivar planes más lentos. • `Preferred` : un plan que ha determinado como plan preferido para usar para una instrucción administrada. Si el costo mínimo del optimizador no es un plan aprobado o preferido, puede reducir la sobrecarga de cumplimiento del plan. Para ello, cree un subconjunto de los planes aprobados `Preferred` . Cuando el costo mínimo del optimizador no sea un plan `Approved`, se seleccionará un plan `Preferred` antes que uno `Approved`. Para restaurar un plan como `Preferred` , utilice la función apg_plan_mgmt.set_plan_status.
`stmt_name`	El nombre de la instrucción SQL dentro de una instrucción PREPARE. Este valor es una cadena vacía para una instrucción preparada sin nombre. Este valor es NULL para una instrucción no preparada.

Columna dba_plans	Descripción
`total_time_benefit_ms`	El beneficio en tiempo total, en milisegundos, de habilitar este plan. Este valor tiene en cuenta tanto el tiempo de planificación como el de ejecución. Si este valor es negativo, existe una desventaja al habilitar este plan. Esta columna se rellena mediante la función apg_plan_mgmt.evolve_plan_baselines.

Funciones avanzadas de la administración de planes de consultas

A continuación, encontrará información sobre las funciones avanzadas de la administración de planes de consulta (QPM) de Aurora PostgreSQL:

Temas

- Captura de planes de ejecución de Aurora PostgreSQL en réplicas
- Compatibilidad con partición de tablas

Captura de planes de ejecución de Aurora PostgreSQL en réplicas

QPM (Administración de planes de consultas) le permite capturar los planes de consultas generados por las réplicas de Aurora y los almacena en la instancia de base de datos principal del clúster de base de datos de Aurora. Puede recopilar los planes de consultas de todas las réplicas de Aurora y mantener un conjunto de planes óptimos en una tabla persistente central en la instancia principal. A continuación, puede aplicar estos planes a otras réplicas cuando lo necesite. De este modo, se mantiene la estabilidad de los planes de ejecución y se mejora el rendimiento de las consultas en todos los clústeres de bases de datos y versiones del motor.

Temas

- Requisitos previos
- Administrar la captura de planes para réplicas de Aurora
- Solución de problemas

Requisitos previos

Activar **`capture_plan_baselines parameter`** en una réplica de Aurora: establezca el parámetro `capture_plan_baselines` en automático o manual para capturar los planes en las réplicas de Aurora. Para obtener más información, consulte apg_plan_mgmt.capture_plan_baselines.

Instalar la extensión postgres_fdw: debe instalar la extensión de contenedor de datos externos `postgres_fdw` para capturar los planes en las réplicas de Aurora. Para instalar la extensión, ejecute el siguiente comando en cada base de datos:

```
postgres=> CREATE EXTENSION IF NOT EXISTS postgres_fdw;
```

Administrar la captura de planes para réplicas de Aurora

Activar la captura de planes para réplicas de Aurora

Debe tener privilegios de `rds_superuser` para crear o eliminar la captura de planes en las réplicas de Aurora. Para obtener más información sobre los roles de usuario y los permisos, consulte Descripción de los roles y permisos de PostgreSQL.

Para capturar planes, llame a la función apg_plan_mgmt.create_replica_plan_capture en la instancia de base de datos del escritor, como se muestra a continuación:

```
postgres=> CALL apg_plan_mgmt.create_replica_plan_capture('endpoint', 'password');
```

- punto de conexión: el punto de conexión o cluster_endpoint del escritor de la base de datos global de Aurora proporciona compatibilidad con la conmutación por error para la captura de planes en réplicas de Aurora.

  Para obtener más información sobre el punto de conexión del escritor de la base de datos global de Aurora, consulte Visualización de los puntos de conexión de una base de datos global de Amazon Aurora.

  Para obtener más información sobre los puntos de conexión del clúster, consulte Puntos de conexión de clúster para Amazon Aurora.
- contraseña: le recomendamos que siga las siguientes indicaciones para crear una contraseña y mejorar la seguridad:
  - Debe contener al menos 8 caracteres únicos.
  - Debe contener al menos una letra en mayúsculas, una letra en minúsculas y un número.

- Debe tener al menos un carácter especial (?, !, #, <, >, *, etc.).

> **Note**
>
> Si cambia el punto de conexión, la contraseña o el número de puerto, debe volver a ejecutar `apg_plan_mgmt.create_replica_plan_capture()` con el punto de conexión y la contraseña para iniciar la captura de planes de nuevo. De lo contrario, no se podrán capturar los planes de las réplicas de Aurora.

Desactivar la captura de planes para réplicas de Aurora

Puede desactivar el parámetro `capture_plan_baselines` en la réplica de Aurora estableciendo su valor en `off` en el grupo de parámetros.

Eliminar la captura de planes para réplicas de Aurora

Puede eliminar completamente la captura de planes en las réplicas de Aurora, pero asegúrese antes de hacerlo. Para eliminar la captura de planes, llame a `apg_plan_mgmt.remove_replica_plan_capture` como se muestra a continuación:

```
postgres=> CALL apg_plan_mgmt.remove_replica_plan_capture();
```

Debe volver a llamar a apg_plan_mgmt.create_replica_plan_capture() para activar la captura de planes en las réplicas de Aurora con el punto de conexión y la contraseña.

Solución de problemas

A continuación, puede encontrar ideas para resolver problemas y soluciones si el plan no se captura en las réplicas de Aurora como se esperaba.

- Configuración de parámetros: compruebe si el parámetro `capture_plan_baselines` está establecido en el valor adecuado para activar la captura de planes.
- Instalación de la extensión **`postgres_fdw`**: utilice la siguiente consulta para comprobar si se ha instalado `postgres_fdw`.

  ```
  postgres=> SELECT * FROM pg_extension WHERE extname = 'postgres_fdw'
  ```

- Llamada a create_replica_plan_capture(): utilice el siguiente comando para comprobar si se ha activado el mapeo de usuarios. De lo contrario, llame a `create_replica_plan_capture()` para iniciar la función.

```
postgres=> SELECT * FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- Punto de conexión y número de puerto: compruebe si el punto de conexión y el número de puerto son correctos. Si no lo son, no se mostrará ningún mensaje de error.

  Usa el siguiente comando para comprobar el punto de conexión utilizado en create() y ver en qué base de datos está alojado:

```
postgres=> SELECT srvoptions FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- reload (): debe llamar a apg_plan_mgmt.reload() después de llamar a apg_plan_mgmt.delete_plan() en las réplicas de Aurora para que la función de eliminación sea efectiva. Esto garantiza que el cambio se haya implementado correctamente.
- Contraseña: debe introducir la contraseña en create_replica_plan_capture() según las pautas mencionadas. De lo contrario, se producirá un error. Para obtener más información, consulte Administrar la captura de planes para réplicas de Aurora. Utilice otra contraseña que se ajuste a los requisitos.
- Conexión entre regiones: la captura de planes en réplicas de Aurora también se admite en la base de datos global de Aurora, donde la instancia de escritor y las réplicas de Aurora pueden estar en diferentes regiones. Asegúrese de utilizar el punto de conexión del escritor de la base de datos global de Aurora para mantener la conectividad después de eventos de conmutación por error o transición. Para obtener más información sobre los puntos de conexión de la base de datos global de Aurora, consulte Visualización de los puntos de conexión de una base de datos global de Amazon Aurora. La instancia de escritor y la réplica entre regiones deben poder comunicarse mediante emparejamiento de VPC. Para obtener más información, consulte Interconexión de VPC. Si se produce una conmutación por error entre regiones, debe volver a configurar el punto de conexión para convertirlo en un nuevo punto de conexión del clúster de base de datos principal.

> **Note**
>
> Cuando utilice un punto de conexión de clúster en lugar de un punto de conexión de escritor de base de datos global de Aurora, tendrá que actualizar el punto de conexión de clúster después de realizar una operación de conmutación por error o transición global.

Compatibilidad con partición de tablas

La administración de planes de consulta (QPM) de Aurora PostgreSQL admite particiones de tablas declarativas en las siguientes versiones:

- Versión 15.3 y versiones posteriores a la 15
- Versión 14.8 y versiones posteriores a la 14
- Versión 13.11 y versiones posteriores a la 13

Para obtener más información, consulte Table Partitioning.

Temas

- Configuración de la partición de tablas
- Captura de planos para la partición de tablas
- Aplicación de un plan de partición de tablas
- Convención de nomenclatura

Configuración de la partición de tablas

Para configurar la partición de tablas en QPMde Aurora PostgreSQL, haga lo siguiente:

1. Establezca `apg_plan_mgmt.plan_hash_version` en 3 o más en el grupo de parámetros del clúster de base de datos.
2. Navegue hasta una base de datos que utilice QPM y tenga entradas en la vista `apg_plan_mgmt.dba_plans`.
3. Llame a `apg_plan_mgmt.validate_plans('update_plan_hash')` para actualizar el valor `plan_hash` en la tabla de planes.

4. Repita los pasos 2 y 3 para todas las bases de datos con administración del plan de consultas habilitada que tengan entradas e la vista `apg_plan_mgmt.dba_plans`.

Para obtener más información sobre estos parámetros, consulte Referencia de parámetros para la administración de planes de consultas de Aurora PostgreSQL.

Captura de planos para la partición de tablas

En QPM, los diferentes planes se distinguen por su valor de `plan_hash`. Para entender cómo cambia `plan_hash`, primero hay que entender planes similares.

La combinación de métodos de acceso, nombres de índice sin dígitos y nombres de particiones sin dígitos, acumulados en el nivel del nodo Append, debe ser constante para que los planes se consideren iguales. Las particiones específicas a las que se accede en los planos no son significativas. En el ejemplo siguiente, se crea una tabla `tbl_a` con 4 particiones.

```
postgres=>create table tbl_a(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table tbl_a1 partition of tbl_a for values from (0) to (1000);
CREATE TABLE
postgres=>create table tbl_a2 partition of tbl_a for values from (1001) to (2000);
CREATE TABLE
postgres=>create table tbl_a3 partition of tbl_a for values from (2001) to (3000);
CREATE TABLE
postgres=>create table tbl_a4 partition of tbl_a for values from (3001) to (4000);
CREATE TABLE
postgres=>create index t_i on tbl_a using btree (i);
CREATE INDEX
postgres=>create index t_j on tbl_a using btree (j);
CREATE INDEX
postgres=>create index t_k on tbl_a using btree (k);
CREATE INDEX
```

Los siguientes planes se consideran iguales porque se utiliza un único método de examen para examinar `tbl_a` independientemente del número de particiones que revise la consulta.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 999 and j < 9910 and k > 50;

                        QUERY PLAN
-----------------------------------------------------------------------
```

```
Seq Scan on tbl_a1 tbl_a
    Filter: ((i >= 990) AND (i <= 999) AND (j < 9910) AND (k > 50))
SQL Hash: 1553185667, Plan Hash: -694232056
(3 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                        QUERY PLAN
-----------------------------------------------------------------
Append
    ->  Seq Scan on tbl_a1 tbl_a_1
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    ->  Seq Scan on tbl_a2 tbl_a_2
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    SQL Hash: 1553185667, Plan Hash: -694232056
    (6 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(8 rows)
```

Los siguientes 3 planes también se consideran iguales porque, en el nivel principal, los métodos de acceso, los nombres de índice sin dígitos y los nombres de las particiones sin dígitos son `SeqScan tbl_a`, `IndexScan (i_idx) tbl_a`.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
```

```
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_i_idx on tbl_a2 tbl_a_2
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(7 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
```

```
(11 rows)
```

Independientemente del diferente orden y número de ocurrencias en las particiones secundarias, los métodos de acceso, los nombres de índice sin dígitos y los nombres de las particiones sin dígitos son constantes en el nivel principal para cada uno de los planes anteriores.

Sin embargo, los planes se considerarán diferentes si se cumple alguna de las siguientes condiciones:

- Se utiliza algún método de acceso adicional en el plan.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 1134525070
(11 rows)
```

- Se deja de utilizar alguno de los métodos de acceso del plan.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
```

```
(6 rows)
```

- Se cambia el índice asociado a un método de índice.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                            QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_j_idx on tbl_a2 tbl_a_2
         Index Cond: (j < 9910)
         Filter: ((i >= 990) AND (i <= 1100) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993343726
(7 rows)
```

Aplicación de un plan de partición de tablas

Los planes aprobados para tablas particionadas se aplican mediante correspondencia posicional. Los planes no son específicos de las particiones y se pueden aplicar a particiones distintas de los planes a los que se hace referencia en la consulta original. Los planes también pueden aplicarse a las consultas que accedan a un número de particiones diferente al esquema original aprobado.

Por ejemplo, si el esquema aprobado es para el siguiente plan:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                            QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
```

```
(10 rows)
```

A continuación, este plan también se puede aplicar a las consultas SQL que hagan referencia a 2, 4 o más particiones. Los posibles planes que podrían surgir de estos escenarios para el acceso a 2 y 4 particiones son:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(8 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a4 tbl_a_4
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(12 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(14 rows)
```

Considere otro plan aprobado con diferentes métodos de acceso para cada partición:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 2032136998
(12 rows)
```

En este caso, cualquier plan que lea de dos particiones no podría aplicarse. A menos que se puedan utilizar todas las combinaciones (método de acceso, nombre de índice) del plan aprobado, no se puede aplicar el plan. Por ejemplo, los siguientes planes tienen distintos hashes de plan y el plan aprobado no se puede aplicar en estos casos:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Bitmap Heap Scan on tbl_a1 tbl_a_1
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a1_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
   ->  Bitmap Heap Scan on tbl_a2 tbl_a_2
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a2_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
  Note: This is not an Approved plan.  No usable Approved plan was found.
  SQL Hash: 1553185667, Plan Hash: -568647260
(13 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1900) AND (j < 9910) AND (k > 50))
 Note: This is not an Approved plan.  No usable Approved plan was found.
 SQL Hash: 1553185667, Plan Hash: -496793743
(8 rows)
```

Convención de nomenclatura

Para que QPM aplique un plan con tablas particionadas declarativas, debe seguir reglas de nomenclatura específicas para las tablas principales, las particiones de tablas y los índices:

- Nombres de las tablas principales: estos nombres deben diferir en letras de alfabeto o caracteres especiales y no solo en dígitos. Por ejemplo, tA, tB y tC son nombres aceptables para tablas principales independientes, mientras que t1, t2 y t3 no lo son.
- Nombres de tablas de particiones individuales: las particiones del mismo elemento principal solo deben diferir entre sí en dígitos. Por ejemplo, los nombres de partición aceptables de tA podrían ser tA1, tA2 o t1A, t2A o incluso varios dígitos.

  Cualquier otra diferencia (letras, caracteres especiales) no garantizará la aplicación del plan.
- Nombres de índice: en la jerarquía de tablas de particiones, asegúrese de que todos los índices tengan nombres únicos. Esto significa que las partes no numéricas de los nombres deben ser diferentes. Por ejemplo, si tiene una tabla particionada nombrada `tA` con un nombre de índice `tA_col1_idx1`, no puede tener otro índice nombrado `tA_col1_idx2`. Sin embargo, puede tener un índice llamado `tA_a_col1_idx2` porque la parte no numérica del nombre es única. Esta regla se aplica a los índices creados tanto en la tabla principal como en las tablas de particiones individuales.

El incumplimiento de las convenciones de nomenclatura anteriores puede provocar que los planes aprobados no se apliquen. El siguiente ejemplo ilustra un error de ejecución de este tipo:

```
postgres=>create table t1(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table t1a partition of t1 for values from (0) to (1000);
CREATE TABLE
postgres=>create table t1b partition of t1 for values from (1001) to (2000);
CREATE TABLE
postgres=>SET apg_plan_mgmt.capture_plan_baselines TO 'manual';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 0;

                            QUERY PLAN
------------------------------------------------------------------------------
 Aggregate
   ->  Append
         ->  Seq Scan on t1a t1_1
               Filter: (i > 0)
```

```
         ->  Seq Scan on t1b t1_2
               Filter: (i > 0)
 SQL Hash: -1720232281, Plan Hash: -1010664377
(7 rows)
```

```
postgres=>SET apg_plan_mgmt.use_plan_baselines TO 'on';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 1000;

                            QUERY PLAN
------------------------------------------------------------------------
 Aggregate
   ->  Seq Scan on t1b t1
         Filter: (i > 1000)
 Note: This is not an Approved plan. No usable Approved plan was found.
 SQL Hash: -1720232281, Plan Hash: 335531806
(5 rows)
```

Aunque los dos planes parezcan idénticos, sus valores `Plan Hash` son diferentes debido a los nombres de las tablas secundarias. Los nombres de las tablas varían en caracteres alfabéticos y no solo en dígitos, lo que provoca un error de cumplimiento.

Uso de extensiones y contenedores de datos externos

Para ampliar la funcionalidad al clúster de bases de datos Aurora PostgreSQL-Compatible Edition puede instalar y utilizar varias extensiones de PostgreSQL. Por ejemplo, si su caso de uso requiere entrada intensiva de datos en tablas muy grandes, puede instalar la extensión `pg_partman` para particionar los datos y así difundir la carga de trabajo.

> **Note**
>
> A partir de la versión 14.5 de Aurora PostgreSQL, Aurora PostgreSQL admite extensiones de lenguaje de confianza para PostgreSQL. Esta característica se implementa como la extensión `pg_tle`, que puede añadir a su Aurora PostgreSQL. Con esta extensión, los desarrolladores pueden crear sus propias extensiones de PostgreSQL en un entorno seguro que simplifica los requisitos de instalación y configuración, así como gran parte de las pruebas preliminares de las nuevas extensiones. Para obtener más información, consulte Uso de Extensiones de lenguaje de confianza para PostgreSQL.

En algunos casos, en lugar de instalar una extensión, puede agregar un módulo específico a la lista de `shared_preload_libraries` en el grupo de parámetros del clúster de base de datos personalizado del clúster de base de datos de Aurora PostgreSQL. Por lo general, el grupo de parámetros del clúster de base de datos predeterminado solo carga las `pg_stat_statements`, pero hay varios otros módulos disponibles para agregarlos a la lista. Por ejemplo, puede añadir la capacidad de programación añadiendo el módulo`pg_cron`, tal como se detalla en Programación de mantenimiento con la extensión pg_cron de PostgreSQL. Como otro ejemplo, puede registrar los planes de ejecución de consultas cargando el módulo `auto_explain`. Para obtener más información, consulte Logging execution plans of queries (Registro de los planes de ejecución de las consultas) en el centro de conocimiento de AWS.

Una extensión que proporciona acceso a datos externos se conoce específicamente como contenedor de datos externos(FDW, por sus siglas en inglés). Por ejemplo, la extensión `oracle_fdw` permite al clúster de bases de datos de Aurora PostgreSQL trabajar con bases de datos Oracle.

También puede especificar con precisión qué extensiones se pueden instalar en la instancia de base de datos de Aurora PostgreSQL, enumerándolas en el parámetro `rds.allowed_extensions`. Para obtener más información, consulte Restringir la instalación de extensiones de PostgreSQL.

A continuación, puede encontrar información sobre la configuración y el uso de algunas de las extensiones, módulos y de los FDW disponibles para Aurora PostgreSQL. Para simplificar, todos ellos se denominan “extensiones”. Puede encontrar listas de las extensiones y los FDW que puede usar con las versiones de Aurora PostgreSQL disponibles actualmente, consulte Versiones de extensión para Amazon Aurora PostgreSQL en las Notas de versión de Aurora PostgreSQL.

- Administración de objetos grandes con el módulo lo
- Administración de datos espaciales con la extensión PostGIS
- Administración de las particiones de PostgreSQL con la extensión pg_partman
- Programación de mantenimiento con la extensión pg_cron de PostgreSQL
- Uso de pgAudit para registrar la actividad de la base de datos
- Uso de pglogical para sincronizar datos entre instancias
- Uso de una base de datos de Oracle con la extensión oracle_fdw
- Uso de bases de datos de SQL Server con la extensión mysql_fdw

Uso de la compatibilidad de extensiones delegadas de Amazon Aurora para PostgreSQL

Al utilizar la compatibilidad de extensiones delegadas de Amazon Aurora para PostgreSQL, puede delegar la administración de la extensión a un usuario que no necesita ser un `rds_superuser`. Con esta compatibilidad de extensiones delegadas, se crea un nuevo rol denominado `rds_extension` que debe asignarse a un usuario para que administre otras extensiones. Este rol puede crear, actualizar y eliminar extensiones.

Puede especificar qué extensiones se pueden instalar en la instancia de base de datos de Aurora PostgreSQL, enumerándolas en el parámetro `rds.allowed_extensions`. Para obtener más información, consulte Uso de extensiones PostgreSQL con Amazon RDS para PostgreSQL.

Puede restringir la lista de extensiones disponibles que el usuario puede administrar con el rol `rds_extension` utilizando el parámetro `rds.allowed_delegated_extensions`.

La compatibilidad de extensiones delegadas está disponible en las siguientes versiones:

- Todas las versiones superiores
- Versión 15.5 y versiones posteriores a la 15

- Versión 14.10 y versiones posteriores a la 14
- Versión 13.13 y versiones posteriores a la 13
- Versión 12.17 y versiones posteriores a la 12

Temas

- Activación de la compatibilidad con extensiones delegadas a un usuario
- Configuración utilizada en la compatibilidad de extensiones delegadas de Amazon Aurora para PostgreSQL
- Desactivar la compatibilidad para la extensión delegada
- Ventajas del uso de la compatibilidad de extensiones delegadas de Amazon Aurora
- Limitación de la compatibilidad de extensiones delegadas de Aurora para PostgreSQL
- Permisos necesarios para determinadas extensiones
- Consideraciones de seguridad
- Eliminación de extensión en cascada deshabilitada
- Ejemplos de extensiones que se pueden agregar mediante la compatibilidad de extensiones delegadas

Activación de la compatibilidad con extensiones delegadas a un usuario

Debe realizar lo siguiente para habilitar la compatibilidad con extensiones delegadas en un usuario:

1. Otorgar el rol **`rds_extension`** a un usuario: conéctese a la base de datos como `rds_superuser` y ejecute el siguiente comando:

   ```
   Postgres => grant rds_extension to user_name;
   ```

2. Defina la lista de extensiones disponibles para que las administren los usuarios delegados: `rds.allowed_delegated_extensions` permite especificar un subconjunto de las extensiones disponibles utilizando `rds.allowed_extensions` en el parámetro del clúster de base de datos. Puede realizar esto en uno de los siguientes niveles:

   - En el clúster o en el grupo de parámetros de la instancia, a través de la Consola de administración de AWS o la API. Para obtener más información, consulte Grupos de parámetros para Amazon Aurora.
   - Use el siguiente comando en el nivel de la base de datos:

```
alter database database_name set rds.allowed_delegated_extensions =
 'extension_name_1,
                    extension_name_2,...extension_name_n';
```

- Use el siguiente comando en el nivel de usuario:

```
alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
                    extension_name_2,...extension_name_n';
```

> **Note**
>
> No es necesario reiniciar la base de datos después de cambiar el parámetro dinámico `rds.allowed_delegated_extensions`.

3. Permita el acceso del usuario delegado a los objetos creados durante el proceso de creación de la extensión: algunas extensiones crean objetos que requieren la concesión de permisos adicionales antes de que el usuario con el rol `rds_extension` pueda acceder a ellos. El `rds_superuser` debe conceder al usuario delegado acceso a esos objetos. Una de las opciones es utilizar un desencadenador de eventos para conceder automáticamente el permiso al usuario delegado.

   Ejemplo de desencadenador de eventos

   Si desea permitir que un usuario delegado con `rds_extension` utilice extensiones que requieran configurar permisos en los objetos creados al crear la extensión, puede personalizar el siguiente ejemplo de un desencadenador de eventos y agregar solo las extensiones para las que desee que los usuarios delegados tengan acceso a todas las funciones. Este activador de eventos se puede crear en la plantilla 1 (la plantilla predeterminada), por lo que todas las bases de datos creadas a partir de la plantilla 1 tendrán ese desencadenador de eventos. Cuando un usuario delegado instala la extensión, este desencadenador otorgará automáticamente la propiedad de los objetos creados por la extensión.

```
CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE
```

```
  schemaname TEXT;
  databaseowner TEXT;

  r RECORD;

BEGIN

  IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
    RAISE NOTICE 'SECURITY INVOKER';
    RAISE NOTICE 'user: %', current_user;
    FOR r IN SELECT * FROM pg_catalog.pg_event_trigger_ddl_commands()
    LOOP
        CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
 'extension';

        schemaname = (
            SELECT n.nspname
            FROM pg_catalog.pg_extension AS e
            INNER JOIN pg_catalog.pg_namespace AS n
            ON e.extnamespace = n.oid
            WHERE e.oid = r.objid
        );

        databaseowner = (
            SELECT pg_catalog.pg_get_userbyid(d.datdba)
            FROM pg_catalog.pg_database d
            WHERE d.datname = current_database()
        );
        RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
 schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
 schemaname, databaseowner;
        IF r.object_identity = 'address_standardizer_data_us' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_gaz TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_lex TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_rules TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'dict_int' THEN
            EXECUTE pg_catalog.format('ALTER TEXT SEARCH DICTIONARY %I.intdict
 OWNER TO %I;', schemaname, databaseowner);
        ELSIF r.object_identity = 'pg_partman' THEN
```

```
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE
 ON TABLE %I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname,
 databaseowner);
        ELSIF r.object_identity = 'postgis_topology' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL
 TABLES IN SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE, SELECT ON ALL SEQUENCES IN
 SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA
 topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE ON SCHEMA topology TO %I WITH
 GRANT OPTION;', databaseowner);
        END IF;
    END LOOP;
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE
 create_ext();
```

Configuración utilizada en la compatibilidad de extensiones delegadas de Amazon Aurora para PostgreSQL

Nombre de la configuración	Descripción	Valor predeterminado	Notas	Quién puede modificar o conceder el permiso
`rds.allowed_delegated_extensions`	Este parámetro limita las extensiones que un rol de rds_extension puede administrar en una base de	empty string	• De forma predeterminada, este parámetro es una cadena vacía, lo que significa	rds_superuser

Nombre de la configuración	Descripción	Valor predeterminado	Notas	Quién puede modificar o conceder el permiso
	datos. Debe ser un subconjunto de rds.allowed_extensions.		que no se ha delegado ninguna extensión a los usuarios con `rds_extension` . • Se puede agregar cualquier extensión compatible si el usuario tiene permisos para hacerlo. Para ello, establezca el parámetro `rds.allowed_delegated_extensions` en una cadena de nombres de extensión separados por comas. Al agregar una lista de extensiones a este parámetro, identifica explícitamente las extensiones que puede instalar el usuario con el	

Nombre de la configuración	Descripción	Valor predeterminado	Notas	Quién puede modificar o conceder el permiso
			rol `rds_extension` . • Si se establece en `*`, significa que todas las extensiones que aparecen en `rds_allowed_extensions` se delegan a los usuarios con el rol `rds_extension` . Para obtener más información sobre la configuración de este parámetro, consulte Activación de la compatibilidad con extensiones delegadas a un usuario.	

Nombre de la configuración	Descripción	Valor predeterminado	Notas	Quién puede modificar o conceder el permiso
`rds.allowed_extensions`	Este parámetro permite que un cliente limite las extensiones que se pueden instalar en la instancia de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Restringir la instalación de extensiones de PostgreSQL.	"*"	De forma predeterminada, este parámetro está establecido en “*”, lo que significa que los usuarios con los privilegios necesarios pueden crear todas las extensiones compatibles con RDS para PostgreSQL y Aurora PostgreSQL. Vacío significa que no se pueden instalar extensiones en la instancia de base de datos de Aurora PostgreSQL.	administrator

Nombre de la configuración	Descripción	Valor predeterminado	Notas	Quién puede modificar o conceder el permiso
`rds-delegated_extension_allow_drascac`	Este parámetro controla la capacidad del usuario con `rds_extension` de eliminar la extensión mediante una opción en cascada.	apagar	De forma predeterminada, `rds-delegated_extension_allow_drop_cascade` está establecido en `off`. Esto significa que los usuarios con `rds_extension` no pueden eliminar una extensión mediante la opción en cascada. Para otorgar esa habilidad, el parámetro `rds.delegated_extension_allow_drop_cascade` debe configurarse como on.	rds_superuser

Desactivar la compatibilidad para la extensión delegada

Desactivación parcial

Los usuarios delegados no pueden crear nuevas extensiones, pero sí pueden actualizar las existentes.

- Restablece `rds.allowed_delegated_extensions` al valor predeterminado en el grupo de parámetros del clúster de base de datos.
- Use el siguiente comando en el nivel de la base de datos:

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- Use el siguiente comando en el nivel de usuario:

```
alter user user_name reset rds.allowed_delegated_extensions;
```

Desactivación completa

Al revocar el rol `rds_extension` de un usuario, el usuario recuperará los permisos estándar. El usuario ya no puede crear, actualizar ni eliminar extensiones.

```
postgres => revoke rds_extension from user_name;
```

Ventajas del uso de la compatibilidad de extensiones delegadas de Amazon Aurora

Al utilizar la compatibilidad de extensiones delegadas de Amazon Aurora para PostgreSQL, delega de forma segura la administración de la extensión a los usuarios que no tengan el rol `rds_superuser`. Esta característica proporciona los siguientes beneficios:

- Puede delegar fácilmente la administración de extensiones a los usuarios de su elección.
- Esto no requiere el rol `rds_superuser`.
- Ofrece la posibilidad de admitir diferentes conjuntos de extensiones para diferentes bases de datos en el mismo clúster de base de datos.

Limitación de la compatibilidad de extensiones delegadas de Aurora para PostgreSQL

- Los objetos creados durante el proceso de creación de la extensión pueden requerir privilegios adicionales para que la extensión funcione correctamente.

Permisos necesarios para determinadas extensiones

Para crear, usar o actualizar las siguientes extensiones, el usuario delegado debe tener los privilegios necesarios en las siguientes funciones, tablas y esquemas.

Exter es que nece propi o perm	Función	Tablas	Esquema	Diccionario de búsqueda de texto	Comment
addre tanda er_da s		us_gaz, us_lex, us_lex, l.us_rules			
amch	bt_index_check, bt_index_ parent_check				
dict_i				intdict	
pg_p n		custom_ti me_partitions, part_config, part_config_sub			
pg_st tatem					
Post(	st_tileenvelope	spatial_ref_sys			
postg aster					
postg opolo		topology, layer	topology		el usuario delegado debe ser el propietar io de la base de datos

Exter es que nece propi o perm	Función	Tablas	Esquema	Diccionario de búsqueda de texto	Comment
log_f	create_fo reign_tab le_for_log_file				
rds_t	role_pass word_encr yption_type				
postg iger_ oder		geocode_s ettings_default, geocode_s ettings	tiger		
pg_fr acem	pg_freespace				
pg_vi lity	pg_visibility				

Consideraciones de seguridad

Tenga en cuenta que un usuario con el rol `rds_extension` podrá administrar las extensiones en todas las bases de datos en las que tenga el privilegio de conexión. Si la intención es que un usuario delegado administre la extensión en una única base de datos, una práctica recomendada consiste en revocar todos los privilegios del público en cada base de datos y, a continuación, conceder de forma explícita el privilegio de conexión para esa base de datos específica al usuario delegado.

Existen varias extensiones que permiten a un usuario acceder a la información de varias bases de datos. Asegúrese de que los usuarios a los que conceda `rds_extension` tengan capacidades para múltiples bases de datos antes de agregar estas extensiones a

`rds.allowed_delegated_extensions`. Por ejemplo, `postgres_fdw` y `dblink` proporcionan la funcionalidad de realizar consultas en todas las bases de datos de la misma instancia o de instancias remotas. `log_fdw` lee los archivos de registro del motor postgres, que son de todas las bases de datos de la instancia, y pueden contener consultas lentas o mensajes de error de varias bases de datos. `pg_cron` permite ejecutar trabajos en segundo plano programados en la instancia de base de datos y puede configurar los trabajos para que se ejecuten en una base de datos diferente.

Eliminación de extensión en cascada deshabilitada

La posibilidad de eliminar la extensión con la opción en cascada por parte de un usuario con el rol `rds_extension` la controla el parámetro `rds.delegated_extension_allow_drop_cascade`. De forma predeterminada, `rds-delegated_extension_allow_drop_cascade` está establecido en `off`. Esto significa que los usuarios con el tol `rds_extension` no pueden eliminar una extensión mediante la opción en cascada como se muestra en la siguiente consulta.

```
DROP EXTENSION CASCADE;
```

Esto eliminará automáticamente los objetos que dependan de la extensión y, a su vez, todos los objetos que dependan de esos objetos. El intento de utilizar la opción en cascada generará un error.

Para otorgar esa habilidad, el parámetro `rds.delegated_extension_allow_drop_cascade` debe configurarse como on.

Cambiar el parámetro dinámico `rds.delegated_extension_allow_drop_cascade` no requiere un reinicio de la base de datos. Puede realizar esto en uno de los siguientes niveles:

- En el clúster o en el grupo de parámetros de la instancia, a través de la Consola de administración de AWS o la API.
- Con el siguiente comando en el nivel de la base de datos:

  ```
  alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
  ```

- Con el siguiente comando en el nivel de usuario:

  ```
  alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
  ```

Ejemplos de extensiones que se pueden agregar mediante la compatibilidad de extensiones delegadas

- rds_tools

```
extension_test_db=> create extension rds_tools;
CREATE EXTENSION
extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
 rolname = 'pg_read_server_files';
ERROR: permission denied for function role_password_encryption_type
```

- amcheck

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- pg_freespacemap

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- pg_visibility

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- `postgres_fdw`

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

Administración de objetos grandes con el módulo lo

El módulo lo (extensión) es para usuarios de base de datos y desarrolladores que trabajan con bases de datos PostgreSQL mediante controladores JDBC u ODBC. Tanto JDBC como ODBC esperan que la base de datos se encargue de eliminar los objetos grandes cuando cambian las referencias a ellos. No obstante, PostgreSQL no funciona así. PostgreSQL no asume que un objeto se deba eliminar cuando cambie su referencia. El resultado es que los objetos permanecen en el disco, sin referencia. La extensión lo incluye una función que se utiliza para desencadenarse en los cambios de referencia con el fin de eliminar objetos si es necesario.

> **Tip**
>
> Para determinar si su base de datos se puede beneficiar de la extensión lo, utilice la utilidad `vacuumlo` para comprobar si hay objetos grandes huérfanos. Para obtener el recuento de los objetos grandes huérfanos sin realizar ninguna acción, ejecute la utilidad con la opción `-n` (sin operación). Para saber cómo, consulte vacuumlo utility a continuación.

El módulo lo está disponible para Aurora PostgreSQL 13.7, 12.11, 11.16, 10.21 y versiones secundarias posteriores.

Para instalar el módulo (extensión), necesita privilegios de `rds_superuser`. La instalación de la extensión lo agrega a su base de datos:

- `lo`: es un tipo de datos de objeto grande (lo) que puede utilizar para objetos binarios grandes (BLOB) y otros objetos de gran tamaño. El tipo de datos `lo` es un dominio del tipo de datos `oid`. Es decir, se trata de un identificador de objeto con restricciones opcionales. Para obtener más información, consulte Object identifiers (Identificadores de objetos) en la documentación de PostgreSQL. Dicho de forma sencilla, puede utilizar el tipo de datos `lo` para distinguir las

columnas de base de datos que contienen referencias a objetos grandes de otros identificadores de objetos (OID).

- `lo_manage`: es una función que puede utilizar en los desencadenadores de las columnas de tabla que contienen referencias a objetos grandes. Siempre que elimine o modifique un valor que haga referencia a un objeto grande, el desencadenador desvincula el objeto (`lo_unlink`) de su referencia. Utilice el desencadenador en una columna solo si esta es la única referencia de base de datos al objeto grande.

Para obtener más información sobre el módulo de objetos grandes, consulte lo en la documentación de PostgreSQL.

Instalación de la extensión lo

Antes de instalar la extensión lo, asegúrese de que dispone de privilegios de `rds_superuser`.

Para instalar la extensión lo

1. Use `psql` para conectarse a la instancia de base de datos principal de su clúster de base de datos de Aurora PostgreSQL.

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Escriba la contraseña cuando se le solicite. El cliente `psql` se conecta y muestra la base de datos de conexión administrativa predeterminada, `postgres=>`, como el símbolo del sistema.

2. Instale la extensión de la siguiente manera:

   ```
   postgres=> CREATE EXTENSION lo;
   CREATE EXTENSION
   ```

Ahora puede utilizar el tipo de datos `lo` para definir columnas en las tablas. Por ejemplo, puede crear una tabla (`images`) que contenga datos de imágenes rasterizadas. Puede utilizar el tipo de datos `lo` para una columna `raster`, como se muestra en el siguiente ejemplo, que crea una tabla.

```
postgres=> CREATE TABLE images (image_name text, raster lo);
```

Uso de la función de desencadenador lo_manage para eliminar objetos

Puede utilizar la función `lo_manage` en un desencadenador en `lo` o en otras columnas de objetos grandes para limpiar (y evitar los objetos huérfanos) cuando `lo` se actualiza o se elimina.

Para configurar desencadenadores en columnas que hacen referencia a objetos grandes

- Realice una de las siguientes acciones:
  - Cree un desencadenador BEFORE UPDATE OR DELETE en cada columna para que contenga referencias únicas a objetos grandes, con el nombre de columna como argumento.

```
postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OR DELETE ON images
    FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
```

  - Aplique un desencadenador solo cuando la columna se esté actualizando.

```
postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OF images
    FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
```

La función de desencadenador `lo_manage` solo funciona en el contexto de la inserción o eliminación de datos de columna, en función de cómo se defina el desencadenador. No tiene ningún efecto cuando se realiza una operación `DROP` o `TRUNCATE` en una base de datos. Esto significa que debe eliminar las columnas de objeto de cualquier tabla antes de suprimirla, para evitar que se creen objetos huérfanos.

Por ejemplo, suponga que desea eliminar la base de datos que contiene la tabla `images`. Elimina la columna de la siguiente manera.

```
postgres=> DELETE FROM images COLUMN raster
```

Si se supone que la función `lo_manage` está definida en esa columna para gestionar las eliminaciones, ahora puede eliminar la tabla con seguridad.

Eliminación de objetos grandes huérfanos mediante **`vacuumlo`**

La utilidad `vacuumlo` identifica y puede eliminar los objetos grandes huérfanos de las bases de datos. Esta utilidad está disponible desde PostgreSQL 9.1.24. Si los usuarios de la base de datos

trabajan habitualmente con objetos grandes, le recomendamos que ejecute `vacuumlo` de vez en cuando para limpiar los objetos grandes huérfanos.

Antes de instalar la extensión lo, puede utilizar `vacuumlo` para evaluar si se puede beneficiar su clúster de bases de datos Aurora PostgreSQL. Para ello, ejecute `vacuumlo` con la opción `-n` (sin operación) para mostrar lo que se eliminaría, como se presenta a continuación:

```
$ vacuumlo -v -n -h your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com -
p 5433 -U postgres docs-lab-spatial-db
Password:*****
Connected to database "docs-lab-spatial-db"
Test run: no large objects will be removed!
Would remove 0 large objects from database "docs-lab-spatial-db".
```

Como muestra la salida, los objetos grandes huérfanos no son un problema para esta base de datos concreta.

Para obtener más información sobre esta utilidad, consulte `vacuumlo` en la documentación de PostgreSQL.

Funcionamiento de **`vacuumlo`**

El comando `vacuumlo` elimina los objetos grandes (LO) huérfanos de la base de datos de PostgreSQL sin que ello afecte ni entre en conflicto con las tablas de usuario.

El comando funciona así:

1. `vacuumlo` comienza creando una tabla temporal que contiene todos los ID de objeto (OID) de los objetos grandes de la base de datos.
2. A continuación, `vacuumlo` escanea todas las columnas de la base de datos que utilizan los tipos de datos `oid` o `lo`. Si `vacuumlo` encuentra un OID coincidente en estas columnas, lo elimina de la tabla temporal. `vacuumlo`comprueba solo y específicamente las columnas con los nombres `oid` o `lo`, no los dominios basados en estos tipos.
3. El resto de las entradas de la tabla temporal representan LO huérfanos, que posteriormente elimina `vacuumlo` de forma segura.

Mejora en el rendimiento de **`vacuumlo`**

Puede mejorar el rendimiento de `vacuumlo` aumentando el tamaño del lote mediante la opción `-l`. Esto permite a `vacuumlo` procesar más LO a la vez.

Si el sistema tiene memoria suficiente y puede alojar la tabla temporal completamente en la memoria, aumentar la configuración de `temp_buffers` para la base de datos puede mejorar el rendimiento. Esto permite que la tabla resida completamente en la memoria, lo que puede mejorar el rendimiento general.

La siguiente consulta estima el tamaño de la tabla temporal:

```
SELECT
    pg_size_pretty(SUM(pg_column_size(oid))) estimated_lo_temp_table_size
FROM
    pg_largeobject_metadata;
```

Consideraciones para objetos grandes

A continuación, encontrará algunas consideraciones importantes que debe tener en cuenta al trabajar con objetos grandes:

- `Vacuumlo` es la única solución, ya que actualmente no existe otro método para eliminar los LO huérfanos.
- Las herramientas como pglogical, la replicación lógica nativa y AWS DMS, que utilizan tecnologías de replicación, no admiten la replicación de objetos de gran tamaño.
- Al diseñar el esquema de la base de datos, evite utilizar objetos grandes siempre que sea posible y, en su lugar, considere la posibilidad de utilizar tipos de datos alternativos como `bytea`.
- Ejecute `vacuumlo` con regularidad, al menos una vez por semana, para evitar problemas con los LO huérfanos.
- Utilice un activador con la función `lo_manage` en las tablas que almacenen objetos grandes para evitar que se creen LO huérfanos.

Administración de datos espaciales con la extensión PostGIS

PostGIS es una extensión de PostgreSQL para almacenar y administrar información espacial. Para obtener más información sobre PostGIS, consulte PostGIS.net.

A partir de la versión 10.5, PostgreSQL admite la biblioteca libprotobuf 1.3.0 utilizada por PostGIS para trabajar con datos de teselas vectoriales de Mapbox.

La configuración de la extensión PostGIS requiere privilegios de `rds_superuser`. Le recomendamos que cree un usuario (rol) para administrar instalar la extensión PostGIS y los datos espaciales. La extensión PostGIS y sus componentes relacionados añaden miles de funciones a PostgreSQL. Considere la posibilidad de crear la extensión PostGIS en su propio esquema si eso tiene sentido para su caso de uso. En el ejemplo siguiente, se muestra cómo instalar la extensión en su propia base de datos, pero esto no es obligatorio.

Temas

- Paso 1: cree un usuario (rol) para administrar la extensión PostGIS
- Paso 2: cargue las extensiones PostGIS
- Paso 3: transferir la propiedad de los esquemas de extensión
- Paso 4: transferir la propiedad de las tablas de PostGIS
- Paso 5: pruebe las extensiones
- Paso 6: Actualice la extensión de PostGIS
- Versiones de extensión PostGIS
- Actualización de PostGIS 2 a PostGIS 3

Paso 1: cree un usuario (rol) para administrar la extensión PostGIS

En primer lugar, conéctese a una instancia de base de datos de RDS para PostgreSQL como usuario con privilegios `rds_superuser`. Si mantuvo el nombre predeterminado al configurar la instancia, conéctese como `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

Cree un rol independiente (usuario) para administrar la extensión PostGIS.

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
CREATE ROLE
```

Conceda los privilegios `rds_superuser` de este rol para permitir que el rol instale la extensión.

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

Cree una base de datos para utilizarla para sus artefactos de PostGIS. Este paso es opcional. O puede crear un esquema en la base de datos de usuarios para las extensiones de PostGIS, pero esto tampoco es obligatorio.

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

Conceda todos los privilegios `gis_admin` en la base de datos `lab_gis`.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

Salga de la sesión y vuelva a conectarse a una instancia de base de datos de RDS para PostgreSQL como `gis_admin`.

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

Continúe configurando la extensión tal y como se detalla en los pasos siguientes.

Paso 2: cargue las extensiones PostGIS

La extensión de PostGIS incluye varias extensiones relacionadas que funcionan juntas para proporcionar funcionalidad geoespacial. Dependiendo de su caso de uso, es posible que no necesite todas las extensiones creadas en este paso.

Utilice instrucciones `CREATE EXTENSION` para cargar las extensiones de PostGIS.

```
CREATE EXTENSION postgis;
CREATE EXTENSION
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
CREATE EXTENSION postgis_topology;
```

```
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION
```

Para verificar los resultados, puede ejecutar la consulta SQL que se muestra en el siguiente ejemplo, que enumera las extensiones y sus propietarios.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Paso 3: transferir la propiedad de los esquemas de extensión

Use las declaraciones de ALTER SCHEMA para transferir la propiedad de los esquemas al rol `gis_admin`.

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

Si desea confirmar el cambio de propiedad, ejecute la siguiente consulta SQL. O bien, puede utilizar el metacomando \dn de la línea de comandos psql.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
```

```
  ORDER BY 1;

       List of schemas
     Name     |     Owner
--------------+---------------
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

Paso 4: transferir la propiedad de las tablas de PostGIS

> **Note**
>
> No cambie la propiedad de las funciones de PostGIS. Para que PostGIS funcione correctamente y reciba actualizaciones, estas funciones deben retener la propiedad original. Para obtener más información sobre los permisos de PostGIS, consulte PostgreSQL Security.

Use la siguiente función para transferir la propiedad de las tablas de PostGIS al rol `gis_admin`. Ejecute la siguiente declaración desde el símbolo del sistema psql para crear la función.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

A continuación, ejecute la siguiente consulta para ejecutar la función `exec` que, a su vez, ejecuta las instrucciones y altera los permisos.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Paso 5: pruebe las extensiones

Para evitar tener que especificar el nombre del esquema, añada el esquema `tiger` a la ruta de búsqueda usando el siguiente comando.

```
SET search_path=public,tiger;
SET
```

Pruebe el esquema `tiger` usando la siguiente instrucción SELECT.

```
SELECT address, streetname, streettypeabbrev, zip
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Para obtener más información sobre esta extensión, consulte Tiger Geocoder en la documentación de PostGIS.

Pruebe el acceso al esquema `topology` usando la siguiente instrucción `SELECT`. Esto llama a la función `createtopology` para registrar un nuevo objeto de topología (my_new_topo) con el identificador de referencia espacial especificado (26986) y la tolerancia predeterminada (0,5). Para obtener más información, visite CreateTopology en la documentación de PostgreSQL.

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Paso 6: Actualice la extensión de PostGIS

Cada nueva versión de PostgreSQL admite una o más versiones de la extensión de PostGIS compatibles con esa versión. La actualización del motor de PostgreSQL a una nueva versión no actualiza automáticamente la extensión de PostGIS. Antes de actualizar el motor de PostgreSQL, normalmente se actualiza PostGIS a la versión más reciente disponible para la versión actual de PostgreSQL. Para obtener más información, consulte Versiones de extensión PostGIS.

Después de actualizar el motor de PostgreSQL, vuelva a actualizar la extensión de PostGIS a la versión compatible con la versión del motor de PostgreSQL recién actualizada. Para obtener más información sobre la actualización del motor PostgreSQL, consulte Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal.

Puede comprobar si hay disponibles actualizaciones de la versión de la extensión PostGIS en su clúster de base de datos de Aurora PostgreSQL en cualquier momento. Para ello, ejecute el siguiente comando. Esta función está disponible con PostGIS 2.5.0 y versiones posteriores.

```
SELECT postGIS_extensions_upgrade();
```

Si su aplicación no es compatible con la última versión de PostGIS, puede instalar una versión anterior de PostGIS que esté disponible en su versión principal de la siguiente manera.

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

Si desea actualizar a una versión específica de PostGIS desde una versión anterior, también puede utilizar el siguiente comando.

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

Dependiendo de la versión desde la que se actualice, es posible que tenga que volver a utilizar esta función. El resultado de la primera ejecución de la función determina si se necesita una función de actualización adicional. Por ejemplo, eso es lo que ocurre para la actualización de PostGIS 2 a PostGIS 3. Para obtener más información, consulte Actualización de PostGIS 2 a PostGIS 3.

Si actualizó esta extensión para prepararse para una actualización de la versión principal del motor de PostgreSQL, puede continuar con otras tareas preliminares. Para obtener más información, consulte Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal.

Versiones de extensión PostGIS

Le recomendamos que instale las versiones de todas las extensiones, como PostGIS, como se indica en Versiones de extensión para Amazon Aurora PostgreSQL en las Notas de la versión de Aurora PostgreSQL. Para obtener una lista de las versiones que están disponibles en su versión, utilice el siguiente comando.

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

Puede encontrar información sobre la versión en las siguientes secciones de las Notas de la versión de Amazon RDS para PostgreSQL:

- Versiones de extensión para Aurora PostgreSQL 14
- Versiones de extensión para Aurora PostgreSQL-Compatible Edition 13
- Versiones de extensión para Aurora PostgreSQL-Compatible Edition 12
- Versiones de extensión para Aurora PostgreSQL-Compatible Edition 11
- Versiones de extensión para Aurora PostgreSQL-Compatible Edition 10
- Versiones de extensión para Aurora PostgreSQL-Compatible Edition 9.6

Actualización de PostGIS 2 a PostGIS 3

A partir de la versión 3.0, la funcionalidad de trama de PostGIS es una extensión separada, `postgis_raster`. Esta extensión tiene su propia ruta de instalación y actualización. Esto elimina del núcleo docenas de funciones, tipos de datos y otros artefactos necesarios para el procesamiento de imágenes de trama desde la extensión `postgis` principal. Esto significa que si su caso de uso no requiere procesamiento de tramas, no es necesario que instale la extensión `postgis_raster`.

En el siguiente ejemplo de actualización, el primer comando de actualización extrae la funcionalidad de trama en la extensión `postgis_raster`. Luego, se requiere un segundo comando de actualización para actualizar `postgis_raster` a la nueva versión.

Para actualizar de PostGIS 2 a PostGIS 3

1. Identifique la versión predeterminada de PostGIS que está disponible para la versión de PostgreSQL en su clúster de base de datos de Aurora PostgreSQL Para ello, ejecute la siguiente consulta.

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                          comment
---------+-----------------+-------------------
+-------------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
```

```
(1 row)
```

2. Identifique las versiones de PostGIS instaladas en cada base de datos en la instancia de escritura del clúster de base de datos de Aurora PostgreSQL. En otras palabras, consulte la base de datos de cada usuario de la siguiente manera.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                            Description
---------+---------+--------
+------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

Esta falta de correspondencia entre la versión predeterminada (PostGIS 3.1.4) y la versión instalada (PostGIS 2.3.7) significa que debe actualizar la extensión de PostGIS.

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. Ejecute la siguiente consulta para comprobar que la funcionalidad ráster ahora está en su propio paquete.

```
SELECT
    probin,
    count(*)
FROM
```

```
    pg_proc
WHERE
    probin LIKE '%postgis%'
GROUP BY
    probin;
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
(2 rows)
```

El resultado muestra que aún hay una diferencia entre las versiones. Las funciones de PostGIS son de la versión 3 (postgis-3), mientras que las funciones ráster (rtpostgis) son de la versión 2 (rtpostgis-2.3). Para completar la actualización, vuelva a ejecutar el comando de actualización, como se indica a continuación.

```
postgres=> SELECT postgis_extensions_upgrade();
```

Puede ignorar los mensajes de advertencia sin problemas. Vuelva a ejecutar la siguiente consulta para comprobar que la actualización se ha completado. La actualización se completa cuando en PostGIS y en todas las extensiones relacionadas deja de aparecer una marca que indica que deben actualizarse.

```
SELECT postgis_full_version();
```

4. Utilice la siguiente consulta para ver el proceso de actualización completado y las extensiones empaquetadas por separado, y compruebe que las versiones coinciden.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
```

```
    1;
      Name      | Version | Schema |                                Description
----------------+---------+--------
+-------------------------------------------------------------------------
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

El resultado muestra que la extensión de PostGIS 2 se ha actualizado a PostGIS 3 y que ambas, `postgis`y la extensión `postgis_raster` ya separada, son de la versión 3.1.5.

Una vez completada esta actualización, si no tiene pensado usar la funcionalidad de trama, puede eliminar la extensión de la siguiente manera.

```
DROP EXTENSION postgis_raster;
```

Administración de las particiones de PostgreSQL con la extensión pg_partman

Las particiones de tablas de PostgreSQL proporcionan un marco para el manejo de alto rendimiento de la entrada de datos y la generación de informes. Utilice particiones para bases de datos que requieren una entrada muy rápida de grandes cantidades de datos. Las particiones también proporcionan consultas más rápidas de tablas grandes. Las particiones ayudan a mantener los datos sin afectar la instancia de base de datos porque requiere menos recursos de E/S.

Mediante el uso de particiones, puede dividir los datos en fragmentos de tamaño personalizado para su procesamiento. Por ejemplo, puede dividir datos de series temporales para rangos como por hora, por día, por semana, por mes, por trimestre, por año, personalizados o cualquier combinación de estos. Para un ejemplo de datos de series temporales, si divide la tabla por hora, cada partición contiene una hora de datos. Si divide la tabla de series temporales por día, las particiones contienen datos de un día, y así sucesivamente. La clave de partición controla el tamaño de una partición.

Cuando se utiliza un comando `INSERT` o `UPDATE` de SQL en una tabla particionada, el motor de base de datos enruta los datos a la partición adecuada. Las particiones de tablas de PostgreSQL que almacenan los datos son tablas secundarias de la tabla principal.

Durante las lecturas de consultas de la base de datos, el optimizador de PostgreSQL analiza la cláusula `WHERE` de la consulta y, si es posible, dirige el análisis de la base de datos solo a las particiones relevantes.

A partir de la versión 10, PostgreSQL utiliza particiones declarativas para implementar particiones de tablas. Esto también se conoce como particionado PostgreSQL nativo. Antes de PostgreSQL versión 10, usaba desencadenadores para implementar particiones.

Las particiones de tablas de PostgreSQL proporcionan las siguientes características:

- Creación de nuevas particiones en cualquier momento.
- Rangos de particiones variables.
- Particiones desmontables y reconectables mediante instrucciones de lenguaje de definición de datos (DDL).

  Por ejemplo, las particiones desmontables son útiles para eliminar datos históricos de la partición principal, pero mantienen los datos históricos para su análisis.
- Las nuevas particiones heredan las propiedades de la tabla de base de datos principal, incluidas las siguientes:
  - Índices
  - Claves principales, que deben incluir la columna de la clave de partición
  - Claves externas
  - Restricciones de comprobación
  - Referencias
- creación de índices para la tabla completa o cada partición específica

No se puede modificar el esquema de una partición individual. Sin embargo, se puede modificar la tabla principal (como agregar una nueva columna), que se propaga a las particiones.

Temas

- Información general de la extensión pg_partman de PostgreSQL
- Habilitación de la extensión pg_partman
- Configuración de particiones mediante la función create_parent
- Configuración del mantenimiento de particiones mediante la función run_maintenance_proc

Información general de la extensión pg_partman de PostgreSQL

Puede utilizar la extensión `pg_partman` de PostgreSQL para automatizar la creación y el mantenimiento de las particiones de tablas. Para obtener más información general, consulte PG Partition Manager en la documentación de `pg_partman`.

> **Note**
>
> La extensión `pg_partman` es compatible con las versiones 12.6 y posteriores de Aurora PostgreSQL.

En lugar de tener que crear manualmente cada partición, configure `pg_partman` con las siguientes opciones:

- Tabla que se dividirá
- Tipo de partición
- Clave de partición
- Grado de detalle de la partición
- Opciones de precreación y administración de particiones

Después de crear una tabla con particiones de PostgreSQL, la registra con `pg_partman` al llamar a la función `create_parent`. Al hacerlo, se crean las particiones necesarias en función de los parámetros que pase a la función.

La extensión `pg_partman` también proporciona la función `run_maintenance_proc`, que puede ejecutarse de forma programada para administrar automáticamente las particiones. Para asegurarse de que se creen las particiones apropiadas según sea necesario, programe esta función para que se ejecute periódicamente (por ejemplo, por hora). También puede asegurarse de que las particiones se eliminen automáticamente.

Habilitación de la extensión pg_partman

Si tiene varias bases de datos dentro de la misma instancia de base de dato de PostgreSQL para la que desea administrar particiones, debe habilitar la extensión `pg_partman` por separado para cada base de datos. Para habilitar la extensión `pg_partman` para una base de datos específica, cree el esquema de mantenimiento de particiones y, después, cree la extensión `pg_partman` de la siguiente manera:

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> Para crear la extensión pg_partman, asegúrese de tener privilegios rds_superuser.

Si recibe un error como el siguiente, conceda los privilegios rds_superuser a la cuenta o utilice su cuenta de superusuario.

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

Para conceder privilegios rds_superuser, conéctese con su cuenta de superusuario y ejecute el siguiente comando:

```
GRANT rds_superuser TO user-or-role;
```

Para los ejemplos que muestran el uso de la extensión pg_partman, utilizamos la siguiente tabla de base de datos y partición de muestra. Esta base de datos utiliza una tabla particionada basada en una marca temporal. Un esquema data_mart contiene una tabla denominada events con una columna denominada created_at. En la events tabla se incluyen los siguientes ajustes:

- Claves primarias event_id y created_at, que deben tener la columna utilizada para guiar la partición.
- Una restricción de comprobación ck_valid_operation para aplicar los valores para una columna de la tabla operation.
- Dos claves externas, donde una (fk_orga_membership) apunta a la tabla externa organization y la otra (fk_parent_event_id) es una clave externa con referencia propia.
- Dos índices, donde uno (idx_org_id) es para la clave externa y el otro (idx_event_type) es para el tipo de evento.

Las siguientes instrucciones DDL crean estos objetos, que se incluyen automáticamente en cada partición.

```
CREATE SCHEMA data_mart;
```

```
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

Configuración de particiones mediante la función create_parent

Después de habilitar la extensión `pg_partman`, utilice la función `create_parent` para configurar las particiones dentro del esquema de mantenimiento de particiones. En este ejemplo se utiliza el ejemplo de la tabla `events` creado en Habilitación de la extensión pg_partman. Ejecute la función `create_parent` de la siguiente manera:

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

Los parámetros son los siguientes:

- `p_parent_table` – La tabla principal particionada. Esta tabla ya debe existir y estar totalmente cualificada, incluido el esquema.
- `p_control` – La columna en la que se basará la partición. El tipo de datos debe ser entero o basado en el tiempo.
- `p_type`: el tipo es `'range'` o `'list'`.
- `p_interval` – El intervalo de tiempo o intervalo de enteros para cada partición. Los valores de ejemplo incluyen `1 day`, `1 hour`, etc.
- `p_premake` – La cantidad de particiones que se debe crear de antemano para admitir nuevas inserciones.

Para obtener una descripción completa de la función `create_parent`, consulte Funciones de creación en la documentación de `pg_partman`.

Configuración del mantenimiento de particiones mediante la función run_maintenance_proc

Puede ejecutar operaciones de mantenimiento de particiones para crear automáticamente nuevas particiones, desasociar particiones o eliminar particiones antiguas. El mantenimiento de particiones se basa en la función `run_maintenance_proc` de la extensión `pg_partman` y la extensión `pg_cron`, que inicia un programador interno. El programador `pg_cron` ejecuta automáticamente instrucciones SQL, funciones y procedimientos definidos en las bases de datos.

En el ejemplo siguiente se utiliza el ejemplo de la tabla `events` creado en Habilitación de la extensión pg_partman para establecer que las operaciones de mantenimiento de particiones se ejecuten automáticamente. Como requisito previo, agregue `pg_cron` al parámetro `shared_preload_libraries` en el grupo de parámetros de la instancia de base de datos.

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

A continuación, puede encontrar una explicación paso a paso del ejemplo anterior:

1. Modifique el grupo de parámetros asociado a la instancia de base de datos y agregue `pg_cron` al valor del parámetro `shared_preload_libraries`. Este cambio requiere un reinicio de la instancia de base de datos para que surta efecto. Para obtener más información, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.
2. Ejecute el comando `CREATE EXTENSION pg_cron;` con una cuenta que tenga los permisos `rds_superuser`. Esto habilita la extensión `pg_cron`. Para obtener más información, consulte Programación de mantenimiento con la extensión pg_cron de PostgreSQL.
3. Ejecute el comando `UPDATE partman.part_config` para ajustar la configuración de `pg_partman` para la tabla `data_mart.events`.
4. Ejecute el comando `SET . . .` para configurar la tabla `data_mart.events`, con estas cláusulas:
   a. `infinite_time_partitions = true,` – Configura la tabla para que pueda crear automáticamente nuevas particiones sin ningún límite.
   b. `retention = '3 months',` – Configura la tabla para que tenga una retención máxima de tres meses.
   c. `retention_keep_table=true` – Configura la tabla para que cuando venza el periodo de retención, la tabla no se elimine automáticamente. En su lugar, las particiones que son anteriores al periodo de retención solo se separan de la tabla principal.
5. Ejecute el comando `SELECT cron.schedule . . .` para hacer una llamada a la función `pg_cron`. Esta llamada define la frecuencia con la que el programador ejecuta el procedimiento de mantenimiento de `pg_partman`, `partman.run_maintenance_proc`. Para este ejemplo, el procedimiento se ejecuta cada hora.

Para obtener una descripción completa de la función `run_maintenance_proc`, consulte Funciones de mantenimiento en la documentación de `pg_partman`.

Programación de mantenimiento con la extensión pg_cron de PostgreSQL

Puede utilizar la extensión `pg_cron` de PostgreSQL para programar comandos de mantenimiento dentro de una base de datos de PostgreSQL. Para obtener más información sobre la extensión, consulte ¿Qué es pg_cron? en la documentación de pg_cron.

La extensión `pg_cron` es compatible con las versiones 12.6 y posteriores del motor de Aurora PostgreSQL.

Para obtener más información acerca del uso de `pg_cron`, consulte Programación de mantenimiento con la extensión pg_cron de PostgreSQL para sus bases de datos de RDS para PostgreSQL o las bases de datos de Aurora PostgreSQL-Compatible Edition

> **Note**
>
> La versión de la extensión de `pg_cron` se muestra como una versión de dos dígitos, por ejemplo, 1.6, en la vista pg_available_extensions. Si bien es posible que vea versiones de tres dígitos, por ejemplo, 1.6.4 o 1.6.5, en algunos contextos, debe especificar la versión de dos dígitos al realizar una actualización de extensión.

Temas

- Configuración de la extensión pg_cron
- Concesión de permisos para usuarios de base de datos para usar pg_cron
- Programación de trabajos pg_cron
- Referencia para la extensión pg_cron

Configuración de la extensión pg_cron

Habilite la extensión de `pg_cron` de la siguiente manera:

1. Modifique el grupo de parámetros personalizado asociado a la instancia de base de datos de PostgreSQL agregando `pg_cron` al valor del parámetro `shared_preload_libraries`.

   Reinicie la instancia de base de datos de PostgreSQL para que se apliquen los cambios en el grupo de parámetros. Para obtener más información acerca de cómo trabajar con grupos de parámetros, consulte Parámetros de Amazon Aurora PostgreSQL..
2. Una vez reiniciada la instancia de base de datos de PostgreSQL, ejecute el siguiente comando con una cuenta que tenga permisos `rds_superuser`. Por ejemplo, si utilizó la configuración predeterminada al crear el clúster de bases de datos de Aurora PostgreSQL, conéctese como usuario `postgres` y cree la extensión.

   ```
   CREATE EXTENSION pg_cron;
   ```

El programador `pg_cron` se establece en la base de datos de PostgreSQL predeterminada que se denomina `postgres`. Los objetos `pg_cron` se crean en esta base de datos `postgres` y todas las acciones de programación se ejecutan en esta base de datos.

3. Puede utilizar la configuración predeterminada o programar trabajos que ejecutar en otras bases de datos en la instancia de base de datos de PostgreSQL. Para programar trabajos de otras bases de datos en la instancia de base de datos de PostgreSQL, consulte el ejemplo en Programación de un trabajo cron para una base de datos que no sea la predeterminada.

Concesión de permisos para usuarios de base de datos para usar pg_cron

La instalación de la extensión `pg_cron` requiere privilegios de `rds_superuser`. Sin embargo, los permisos para usar `pg_cron` se pueden conceder (los concede un miembro del grupo/rol `rds_superuser`) a otros usuarios de la base de datos para que puedan programar sus propios trabajos. Es recomendable que conceda permisos al esquema `cron` solo según sea necesario si mejora las operaciones en su entorno de producción.

Para conceder permiso a un usuario de base de datos en el esquema `cron`, ejecute el siguiente comando:

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

Esto da permiso *`db-user`* para acceder al esquema de `cron` para programar trabajos cron para los objetos a los que tienen permiso de acceso. Si el usuario de la base de datos no tiene permisos, se produce un error en el trabajo tras publicar el mensaje de error en el `postgresql.log`, como se muestra a continuación:

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

En otras palabras, asegúrese de que los usuarios de bases de datos a los que se les conceden permisos en el esquema de `cron` también tengan permisos sobre los objetos (tablas, esquemas, etc.) que tienen pensado programar.

Los detalles del trabajo cron y su éxito o fracaso también se capturan en la tabla `cron.job_run_details`. Para obtener más información, consulte Tablas para programar trabajos y capturar estado .

Programación de trabajos pg_cron

En las secciones que siguen se muestra cómo programar varias tareas de administración con trabajos `pg_cron`.

> **Note**
>
> Al crear trabajos `pg_cron`, compruebe que el valor `max_worker_processes` sea mayor que el número de `cron.max_running_jobs`. Se producirá un error en el trabajo `pg_cron` si se queda sin procesos de trabajo en segundo plano. El número predeterminado de trabajos `pg_cron` es 5. Para obtener más información, consulte Parámetros para administrar la extensión pg_cron.

Temas

- Limpieza de tablas
- Depuración de la tabla del historial pg_cron
- Registrar errores únicamente en el archivo postgresql.log
- Programación de un trabajo cron para una base de datos que no sea la predeterminada

Limpieza de tablas

En la mayoría de los casos, autovacuum maneja el mantenimiento de limpieza. Sin embargo, se recomienda programar una limpieza de una tabla específica en el momento que lo desee.

A continuación, se muestra un ejemplo del uso de la función `cron.schedule` para configurar un trabajo para usar `VACUUM FREEZE` en una tabla específica todos los días a las 22:00 (GMT).

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

Una vez ejecutado el ejemplo anterior, puede comprobar del siguiente modo el historial de la `cron.job_run_details` tabla.

```
postgres=> SELECT * FROM cron.job_run_details;
```

```
jobid  | runid | job_pid | database | username | command                          |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+-------------------------------
+-----------+---------------+------------------------------
+-------------------------------
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
(1 row)
```

A continuación, se presenta una consulta de la tabla `cron.job_run_details` para ver los trabajos fallidos.

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
jobid | runid | job_pid | database | username | command                        | status
 | return_message                                | start_time                    |
 end_time
------+-------+---------+----------+----------+--------------------------------+--------
+-------------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

Para obtener más información, consulte Tablas para programar trabajos y capturar estado .

Depuración de la tabla del historial pg_cron

La tabla `cron.job_run_details` contiene un historial de los trabajos cron que con el tiempo pueden volverse muy grandes. Se recomienda programar un trabajo que depure esta tabla. Por ejemplo, mantener entradas de una semana podría ser suficiente para solucionar problemas.

En el siguiente ejemplo se utiliza la función cron.schedule para programar un trabajo que se ejecuta todos los días a la medianoche para depurar la tabla `cron.job_run_details`. El trabajo mantiene solo los últimos siete días. Utilice su cuenta de `rds_superuser` para programar el trabajo de la siguiente manera:

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

Para obtener más información, consulte Tablas para programar trabajos y capturar estado .

Registrar errores únicamente en el archivo postgresql.log

Para evitar escribir en la tabla `cron.job_run_details`, modifique el grupo de parámetros asociado a la instancia de base de datos de PostgreSQL y establezca el parámetro `cron.log_run` en Off (Desactivado). La extensión `pg_cron` ya no escribe en la tabla y captura errores solo en el archivo `postgresql.log`. Para obtener más información, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Utilice el siguiente comando para comprobar el valor del parámetro `cron.log_run`.

```
postgres=> SHOW cron.log_run;
```

Para obtener más información, consulte Parámetros para administrar la extensión pg_cron.

Programación de un trabajo cron para una base de datos que no sea la predeterminada

Todos los metadatos de `pg_cron` se mantienen en la base de datos predeterminada de PostgreSQL que se denomina `postgres`. Dado que los trabajadores en segundo plano se utilizan para ejecutar los trabajos cron de mantenimiento, puede programar un trabajo en cualquiera de sus bases de datos dentro de la instancia de base de datos de PostgreSQL:

> **Note**
>
> Solo los usuarios con el rol `rds_superuser` o los privilegios `rds_superuser` pueden mostrar todos los trabajos cron en la base de datos. Los demás usuarios solo pueden ver sus propios trabajos en la tabla `cron.job`.

1. En la base de datos cron, programe el trabajo como lo hace normalmente mediante el uso de cron.schedule.

   ```
   postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
    freeze test_table');
   ```

2. Como usuario con el rol `rds_superuser`, actualice la columna de base de datos para el trabajo que acaba de crear a fin de que se ejecute en otra base de datos dentro de la instancia de base de datos de PostgreSQL.

```
postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
```

3. Verifique consultando la tabla `cron.job`.

```
postgres=> SELECT * FROM cron.job;
jobid | schedule    | command                          | nodename  | nodeport |
 database | username  | active | jobname
------+-------------+----------------------------------+-----------+----------
+----------+-----------+--------+-------------------------
106   | 29 03 * * * | vacuum freeze test_table         | localhost | 8192     |
 database1| adminuser | t      | database1 manual vacuum
  1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
 postgres | adminuser | t      | manual vacuum
(2 rows)
```

> **Note**
>
> En algunas situaciones, puede agregar un trabajo cron que desea ejecutar en otra base de datos. En tales casos, el trabajo podría intentar ejecutarse en la base de datos predeterminada (`postgres`) antes de actualizar la columna de la base de datos correcta. Si el nombre de usuario tiene permisos, el trabajo se ejecuta correctamente en la base de datos predeterminada.

Referencia para la extensión pg_cron

Con la extensión `pg_cron`, puede utilizar los siguientes parámetros, funciones y tablas. Para obtener más información, consulte ¿Qué es pg_cron? en la documentación de pg_cron.

Temas

- Parámetros para administrar la extensión pg_cron
- Referencia de función: cron.schedule
- Referencia de función: cron.unschedule
- Tablas para programar trabajos y capturar estado

Parámetros para administrar la extensión pg_cron

A continuación, aparece la lista de parámetros para controlar el comportamiento de la extensión `pg_cron`.

Parámetro	Descripción
cron.database_name	La base de datos en la que se conservan los metadatos de `pg_cron`.
`cron.host`	El nombre de host que se va a conectar a PostgreSQL. No se puede modificar este valor.
`cron.log_run`	Registre todos los trabajos que se ejecutan en la tabla `job_run_details` . Los valores son on o `off`. Para obtener más informaci ón, consulte Tablas para programar trabajos y capturar estado .
`cron.log_statement`	Registre todas las instrucciones cron antes de ejecutarlas. Los valores son on o `off`.
`cron.max_running_jobs`	La cantidad máxima de trabajos que se pueden ejecutar simultáneamente.
`cron.use_background_workers`	Utilice procesos de trabajo secundarios en lugar de sesiones de cliente. No se puede modificar este valor.

Utilice el siguiente comando SQL para mostrar estos parámetros y sus valores:

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

Referencia de función: cron.schedule

Esta función programa un trabajo cron. El trabajo se programa inicialmente en la base de datos predeterminada `postgres`. La función devuelve un valor `bigint` que representa el identificador del

trabajo. Para programar trabajos para que se ejecuten en otras bases de datos dentro de la instancia de base de datos de PostgreSQL, consulte el ejemplo en Programación de un trabajo cron para una base de datos que no sea la predeterminada.

La función presenta dos formatos de sintaxis.

Sintaxis

```
cron.schedule (job_name,
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

Parámetros

Parámetro	Descripción
`job_name`	El nombre del trabajo cron.
`schedule`	Texto que indica la programación del trabajo cron. El formato es el formato cron estándar.
`command`	Texto del comando que se va a ejecutar.

Ejemplos

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)

postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
```

```
(1 row)
```

Referencia de función: cron.unschedule

Esta función elimina un trabajo cron. Puede especificar `job_name` o `job_id`. Una política se asegura de que usted es el propietario para quitar la programación del trabajo. La función devuelve un valor booleano que indica éxito o error.

La función tiene los siguientes formatos de sintaxis.

Sintaxis

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

Parámetros

Parámetro	Descripción
`job_id`	El identificador de trabajo que se devolvió desde la función `cron.schedule` cuando se programó el trabajo cron.
`job_name`	El nombre de un trabajo cron que se programó con la función `cron.schedule` .

Ejemplos

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)

postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
```

```
(1 row)
```

Tablas para programar trabajos y capturar estado

Las siguientes tablas se crean y utilizan para programar los trabajos cron y registrar la forma en la que se completaron.

Tabla	Descripción
`cron.job`	Contiene los metadatos de cada trabajo programado. La mayoría de las interacciones con esta tabla se deben hacer mediante el uso de las funciones `cron.schedule` y `cron.unschedule` . **Important** No recomendamos conceder privilegios de actualización o inserción directamente a esta tabla. Al hacerlo, el usuario podría actualizar la columna `username` para que se ejecute como `rds-superuser` .
`cron.job_run_details`	Contiene información histórica sobre ejecuciones de trabajos programados anteriores. Esto resulta útil para investigar el estado, los mensajes devueltos y la hora de inicio y finalización de la ejecución del trabajo. **Note** Para evitar que esta tabla crezca indefinidamente, púrguela regularmente. Para ver un ejemplo, consulte Depuración de la tabla del historial pg_cron.

Uso de pgAudit para registrar la actividad de la base de datos

Las instituciones financieras, las agencias gubernamentales y muchas industrias necesitan mantener registros de auditoría para cumplir con los requisitos reglamentarios. Al utilizar la extensión de

auditoría de PostgreSQL (PGAudit) con su Clúster de base de datos de Aurora PostgreSQL, puede capturar los registros detallados que suelen necesitar los auditores o para cumplir con los requisitos reglamentarios. Por ejemplo, puede configurar la extensión pgAudit para realizar un seguimiento de los cambios realizados en bases de datos y tablas específicas, para registrar el usuario que realizó el cambio y muchos otros detalles.

La extensión pgAudit se basa en la funcionalidad de la infraestructura de registro nativa de PostgreSQL ampliando los mensajes de registro con más detalle. En otras palabras, utiliza el mismo método para ver el registro de auditoría que para ver cualquier mensaje de registro. Para obtener más información sobre los registros de PostgreSQL, consulte Archivos de registro de bases de datos de Aurora PostgreSQL.

La extensión PGAudit elimina los datos confidenciales, como las contraseñas de texto no cifrado, de los registros. Si su clúster de base de datos de Aurora PostgreSQL está configurado para registrar las instrucciones del lenguaje de manipulación de datos (DML) tal como se detalla en Activación de registro de consultas para su clúster de base de datos de Aurora PostgreSQL, puede evitar el problema de la contraseña de texto sin cifrar mediante la extensión de auditoría de PostgreSQL.

Puede configurar la auditoría en las instancias de la base de datos con un alto grado de especificidad. Puede auditar todas las bases de datos y todos los usuarios. O bien, puede optar por auditar solo determinadas bases de datos, usuarios y otros objetos. También puede excluir explícitamente a determinados usuarios y bases de datos de la auditoría. Para obtener más información, consulte Exclusión de usuarios o bases de datos del registro de auditoría.

Dada la cantidad de detalles que se pueden capturar, le recomendamos que, si usa pgAudit, controle su consumo de almacenamiento.

La extensión pgAudit es compatible con todas las versiones de Aurora PostgreSQL disponibles. Para obtener una lista de las versiones de pgAudit compatibles con la versión de Aurora PostgreSQL, consulte Versiones de extensión para Amazon Aurora PostgreSQL en las Notas de la versión de Aurora PostgreSQL.

Temas

- Configuración de la extensión pgAudit
- Auditoría de objetos de base de datos
- Exclusión de usuarios o bases de datos del registro de auditoría
- Referencia para la extensión pgAudit

Configuración de la extensión pgAudit

Para configurar la extensión pgAudit en el clúster de base de datos de Aurora PostgreSQL, primero hay que añadir pgAudit a las bibliotecas compartidas en el grupo de parámetros de clústeres de bases de datos personalizados para su clúster de bases de datos de Aurora PostgreSQL. Para obtener información acerca de cómo crear el grupo de parámetros del clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora. A continuación, instale la extensión pgAudit. Por último, especifique las bases de datos y objetos que desea auditar. Los procedimientos de esta sección le muestran cómo hacerlo. Puede utilizar la Consola de administración de AWS o la AWS CLI.

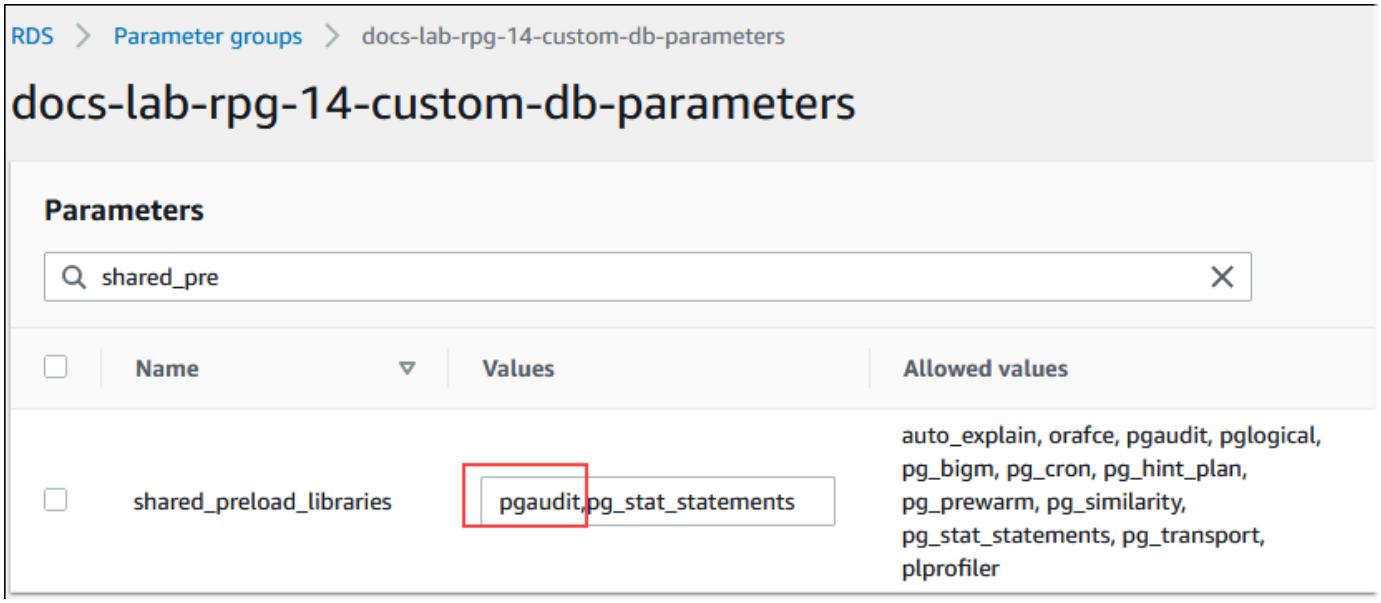
Debe tener permisos como el rol `rds_superuser` para realizar todas estas tareas.

En los pasos siguientes se supone que el clúster de ase de datos de Aurora PostgreSQL está asociado a un grupo de parámetros de clúster de base de datos.

Consola

Para configurar la extensión pgAudit

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija la instancia de escritor del clúster de base de datos de Aurora PostgreSQL .
3. Abra la pestaña Configuration (Configuración) para su instancia de escritor del clúster de base de datos de Aurora PostgreSQL. Entre los detalles de la instancia, busque el enlace del grupo de parámetros.
4. Elija el enlace para abrir los parámetros personalizados asociados al clúster de base de datos de Aurora PostgreSQL.
5. En el campo de búsqueda Parametes (Parámetros), escriba `shared_pre` para buscar el parámetro `shared_preload_libraries`.
6. Seleccione Edit parameters (Editar parámetros) para acceder a los valores de las propiedades.
7. Añada `pgaudit` a la lista en el campo Values (Valores). Utilice una coma para separar los elementos de la lista de valores.



8. Reinicie la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que surta efecto el cambio en el parámetro `shared_preload_libraries`.
9. Cuando la instancia esté disponible, compruebe que pgAudit se haya inicializado. Use `psql` para conectarse a la instancia de escritor de su clúster de bases de datos Aurora PostgreSQL, y, a continuación, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. Con pgAudit inicializado, ahora puede crear la extensión. Debe crear la extensión después de inicializar la biblioteca, ya que la extensión `pgaudit` instala activadores de eventos para auditar las sentencias del lenguaje de definición de datos (DDL).

```
CREATE EXTENSION pgaudit;
```

11. Cierre la sesión de `psql`.

```
labdb=> \q
```

12. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

13. Busque el parámetro `pgaudit.log` en la lista y configúrelo con el valor adecuado para su caso de uso. Por ejemplo, al establecer el parámetro `pgaudit.log` en `write` como se muestra en la siguiente imagen, se capturan las inserciones, las actualizaciones, las eliminaciones y algunos otros tipos de cambios en el registro.

También puede elegir uno de los siguientes valores para el parámetro `pgaudit.log`.

- none: es el valor predeterminado. No se registran cambios en la base de datos.
- all: registra todo (read, write, function, role, ddl, misc).
- ddl: registra todas las instrucciones del lenguaje de definición de datos (DDL) que no están incluidas en la clase `ROLE`.
- function: registra llamadas a funciones y bloques `DO`.
- misc: registra comandos variados como, por ejemplo, `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` y `SET`.
- read: registra `SELECT` y `COPY` cuando el origen es una relación (como una tabla) o una consulta.
- role: registra instrucciones relacionadas con roles y privilegios, como `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` y `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` y `COPY` cuando el destino es una relación (tabla).

14. Elija Guardar cambios.
15. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

16. Elija su instancia de escritor de clúster de base de datos de Aurora PostgreSQL desde la lista de bases de datos.

AWS CLI

Para configurar pgAudit

Para configurar pgAudit mediante AWS CLI, llame a la operación modify-db-parameter-group para modificar los parámetros del registro de auditoría de su grupo de parámetros personalizado, como se muestra en el siguiente procedimiento.

1. Utilice el siguiente comando AWS CLI para añadir `pgaudit` al parámetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilice el siguiente comando AWS CLI para reiniciar la instancia de escritor de la instancia de base de datos de Aurora PostgreSQL para que se inicialice la biblioteca pgaudit.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Cuando la instancia esté disponible, puede verificar si `pgaudit` se ha inicializado. Use `psql` para conectarse a la instancia de escritor de su clúster de bases de datos Aurora PostgreSQL, y, a continuación, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

   Con pgAudit inicializado, ahora puede crear la extensión.

```
CREATE EXTENSION pgaudit;
```

4. Cierre la sesión de `psql` para poder utilizar AWS CLI.

```
labdb=> \q
```

5. Utilice el siguiente comando AWS CLI para especificar las clases de instrucciones que desea registrar con el registro de auditoría de sesión. El ejemplo establece el parámetro `pgaudit.log` en`write`, que captura las inserciones, las actualizaciones y las eliminaciones del registro.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
   --region aws-region
```

   También puede elegir uno de los siguientes valores para el parámetro `pgaudit.log`.

   - none: es el valor predeterminado. No se registran cambios en la base de datos.
   - all: registra todo (read, write, function, role, ddl, misc).
   - ddl: registra todas las instrucciones del lenguaje de definición de datos (DDL) que no están incluidas en la clase `ROLE`.
   - function: registra llamadas a funciones y bloques `DO`.
   - misc: registra comandos variados como, por ejemplo, `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` y `SET`.
   - read: registra `SELECT` y `COPY` cuando el origen es una relación (como una tabla) o una consulta.
   - role: registra instrucciones relacionadas con roles y privilegios, como `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` y `DROP ROLE`.
   - write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` y `COPY` cuando el destino es una relación (tabla).

   Reinicie la instancia de escritor de su clúster de bases de datos Aurora PostgreSQL mediante el siguiente comando AWS CLI.

```
aws rds reboot-db-instance \
```

```
    --db-instance-identifier writer-instance \
    --region aws-region
```

Auditoría de objetos de base de datos

Con PGAudit configurado en su clúster de base de datos de Aurora PostgreSQL y configurado según sus requisitos, se captura información más detallada en el registro de PostgreSQL. Por ejemplo, si bien la configuración de registro predeterminada de PostgreSQL identifica la fecha y la hora en que se realizó un cambio en una tabla de base de datos, con la extensión pgAudit la entrada de registro puede incluir el esquema, el usuario que realizó el cambio y otros detalles, según cómo estén configurados los parámetros de la extensión. Puede configurar la auditoría para realizar un seguimiento de los cambios de las siguientes maneras.

- Para cada sesión, por usuario. Para el nivel de sesión, puede capturar el texto completo del comando.
- Para cada objeto, por usuario y por base de datos.

La capacidad de auditoría de objetos se activa cuando se crea el rol `rds_pgaudit` en el sistema y, a continuación, se agrega este rol al parámetro `pgaudit.role` del grupo de parámetros personalizados. De forma predeterminada, el parámetro `pgaudit.role` no está configurado y el único valor permitido es `rds_pgaudit`. En los siguientes pasos se asume que `pgaudit` se ha inicializado y que ha se ha creado la extensión `pgaudit` siguiendo el procedimiento descrito en Configuración de la extensión pgAudit.

```
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: statement: SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: AUDIT: SESSION,2,1,READ,SELECT,TABLE,public.support,"SELECT
feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;",<none>
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:LOG: QUERY STATISTICS
2022-10-07 23:36:51 UTC:52.95.4.10(14410):postgres@labdb:[1374]:DETAIL: ! system usage stats:
! 0.009494 s user, 0.007442 s system, 0.141985 s elapsed
! [0.022327 s user, 0.007442 s system total]
```

Como se muestra en este ejemplo, la línea «LOG: AUDIT: SESSION» proporciona información sobre la tabla y su esquema, entre otros detalles.

Para configurar la auditoría de objetos

1. Use `psql` para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

```
psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
username=postgrespostgres --password --dbname=labdb
```

2. Cree un rol de base de datos llamado `rds_pgaudit` mediante el siguiente comando.

```
labdb=> CREATE ROLE rds_pgaudit;
CREATE ROLE
labdb=>
```

3. Cierre la sesión de `psql`.

```
labdb=> \q
```

En los siguientes pasos, use elAWS CLI para modificar los parámetros del registro de auditoría en el grupo de parámetros personalizado.

4. Utilice el siguiente comando AWS CLI para establecer el parámetro `pgaudit.role` en `rds_pgaudit`. De forma predeterminada, este parámetro está vacío y `rds_pgaudit` es el único valor permitido.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
 \
   --region aws-region
```

5. Reinicie AWS CLIla instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que sus cambios en los parámetros surtan efecto.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

6. Ejecute el siguiente comando para confirmar que `pgaudit.role` se establece en `rds_pgaudit`.

```
SHOW pgaudit.role;
pgaudit.role
------------------
rds_pgaudit
```

Para probar el registro de pgAudit, puede ejecutar varios comandos de ejemplo que desee auditar. Por ejemplo, podría ejecutar los siguientes comandos.

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

Los registros de base de datos contendrán una entrada similar a la siguiente.

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

Para obtener información acerca de la visualización de los registros, consulte Supervisión de archivos de registro de Amazon Aurora.

Para obtener más información sobre la extensión pgAudit, consulte pgAudit en GitHub.

Exclusión de usuarios o bases de datos del registro de auditoría

Como se explica en Archivos de registro de bases de datos de Aurora PostgreSQL, los registros de PostgreSQL consumen espacio de almacenamiento. El uso de la extensión pgAudit aumenta el volumen de datos recopilados en los registros en diversos grados, según los cambios de los que realice un seguimiento. Es posible que no necesite auditar todos los usuarios o bases de datos de su clúster de base de datos de Aurora PostgreSQL.

Para minimizar los impactos en el almacenamiento y evitar la captura innecesaria de registros de auditoría, puede excluir a los usuarios y las bases de datos de la auditoría. También puede cambiar el registro dentro de una sesión determinada. Los siguientes ejemplos muestran cola forma de hacerlo.

> **Note**
>
> La configuración de los parámetros a nivel de sesión tiene prioridad sobre la configuración del grupo de parámetros del grupo de parámetros del clúster DB personalizado para la instancia de escritor del clúster DB de Aurora PostgreSQL. Si no desea que los usuarios de la base de datos omitan los ajustes de configuración del registro de auditoría, asegúrese de cambiar sus permisos.

Supongamos que su clúster de base de datos de Aurora PostgreSQL está configurado para auditar el mismo nivel de actividad para todos los usuarios y bases de datos. A continuación, decide que no desea auditar al usuario `myuser`. Puede desactivar la auditoría de `myuser` con el siguiente comando de SQL.

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

A continuación, puede utilizar la siguiente consulta para comprobar la columna `user_specific_settings` de `pgaudit.log` para confirmar que el parámetro está establecido en `NONE`.

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

Debería ver una salida como la siguiente.

```
 user_name | user_specific_settings
-----------+------------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

Puede desactivar el registro de un usuario determinado en medio de su sesión con la base de datos con el siguiente comando.

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

Utilice la siguiente consulta para comprobar la columna de configuración de pgaudit.log para una combinación específica de usuario y base de datos.

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

Se muestra una salida similar a la siguiente.

```
  user_name | database_name |     settings
-----------+---------------+------------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

Tras desactivar la auditoría de `myuser`, decide que no desea realizar un seguimiento de los cambios en `mydatabase`. Puede desactivar la auditoría de esa base de datos específica mediante el siguiente comando.

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

A continuación, utilice la siguiente consulta para comprobar la columna database_specific_settings y confirmar que pgaudit.log tiene el valor NONE.

```
SELECT
a.datname AS database_name,
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
```

```
a.datname = 'mydatabase';
```

Debería ver una salida como la siguiente.

```
 database_name | database_specific_settings
---------------+---------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

Para restablecer la configuración predeterminada de myuser, use el siguiente comando:

```
ALTER USER myuser RESET pgaudit.log;
```

Para restablecer la configuración predeterminada de una base de datos, use el siguiente comando:

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

Para restablecer el usuario y la base de datos a la configuración por defecto, utilice el siguiente comando.

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

También puede capturar eventos específicos en el registro configurando `pgaudit.log` en uno de los otros valores permitidos para el parámetro `pgaudit.log`. Para obtener más información, consulte Lista de ajustes permitidos para el parámetro `pgaudit.log`.

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

Referencia para la extensión pgAudit

Puede especificar el nivel de detalle que desea para el registro de auditoría cambiando uno o más de los parámetros que se enumeran en esta sección.

Control del comportamiento de pgAudit

Puede controlar el registro de auditoría cambiando uno o más de los parámetros que aparecen en la tabla siguiente.

Parámetro	Descripción
`pgaudit.log`	Especifica las clases de instrucciones que se registrarán mediante el registro de auditoría de sesión. Los valores permitidos incluyen ddl, function, misc, read, role, write, none, all. Para obtener más información, consulte Lista de ajustes permitidos para el parámetro `pgaudit.log` .
`pgaudit.log_catalog`	Cuando se activa (se establece en 1), agrega instrucciones al registro de auditoría si todas las relaciones de una instrucción se encuentran en pg_catalog.
`pgaudit.log_level`	Especifica el nivel de registro que se usará para las entradas de registro. Valores permitidos, debug5, debug4, debug3, debug2, debug1, info, notice, warning, log.
`pgaudit.log_parameter`	Cuando se activa (se establece en 1), los parámetros transmitidos con la instrucción se capturan en el registro de auditoría.
`pgaudit.log_relation`	Cuando se activa (se establece en 1), el registro de auditoría de la sesión crea una entrada de registro separada para cada relación (TABLE, VIEW, etc.) referenciada en una instrucción SELECT o DML.
`pgaudit.log_statement_once`	Especifica si el registro incluirá el texto de la instrucción y los parámetros con la primera entrada de registro para una combinación de instrucción o subinstrucción o con cada entrada.
`pgaudit.role`	Especifica el rol maestro que se usará para el registro de auditoría de objetos. La única entrada permitida es `rds_pgaudit` .

Lista de ajustes permitidos para el parámetro **`pgaudit.log`**

Valor	Descripción
Ninguno	Esta es la opción predeterminada. No se registran cambios en la base de datos.
Todos	Registra todo (read, write, function, role, ddl, misc).
ddl	Registra todas las instrucciones del lenguaje de definición de datos (DDL) que no están incluidas en la clase `ROLE`.
función	Registra llamadas a funciones y bloques `DO`.
misc	Registra comandos variados como, por ejemplo, `DISCARD`, `FETCH`, `CHECKPOINT` , `VACUUM` y `SET`.
leer	Registra `SELECT` y `COPY` cuando el origen es una relación (como una tabla) o una consulta.
role	Registra instrucciones relacionadas con roles y privilegios, como, por ejemplo, `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` y `DROP ROLE`.
write	Registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` y `COPY` cuando el destino es una relación (tabla).

Para registrar varios tipos de eventos con auditorías de sesiones, utilice una lista separada por comas. Para registrar todos los tipos de eventos, establezca `pgaudit.log` en `ALL`. Reinicie la instancia de base de datos para aplicar los cambios.

Con la auditoría de objetos, puede mejorar los registros de auditoría para que funcionen con algunas relaciones específicas. Por ejemplo, puede especificar que desea crear registros de auditoría para las operaciones `READ` en una o más tablas.

Uso de pglogical para sincronizar datos entre instancias

Todas las versiones de Aurora PostgreSQL disponibles actualmente admiten la extensión `pglogical`. La extensión pglogical es anterior a la función de replicación lógica funcionalmente similar que se introdujo en la versión 10 de PostgreSQL. Para obtener más información, consulte Información general sobre la replicación lógica de PostgreSQL con Aurora.

La extensión `pglogical` admite la replicación lógica entre dos o más clústeres de base de datos de Aurora PostgreSQL. También admite la replicación entre diferentes versiones de PostgreSQL y entre bases de datos que se ejecutan en instancias de base de datos de RDS para PostgreSQL y clústeres de bases de datos de Aurora PostgreSQL. La extensión `pglogical` utiliza un modelo de publicación y suscripción para replicar los cambios en las tablas y otros objetos, como secuencias, de un publicador a un suscriptor. Se basa en una ranura de replicación para garantizar que los cambios se sincronicen de un nodo publicador a un nodo suscriptor, que se define de la siguiente manera.

- El nodo publicador es el clúster de base de datos de Aurora PostgreSQL, que es la fuente de datos que se van a replicar en otros nodos. El nodo publicador define las tablas que se van a replicar en un conjunto de publicaciones.
- El nodo suscriptor es el clúster de base de datos de Aurora PostgreSQL que recibe las actualizaciones WAL del publicador. El suscriptor crea una suscripción para conectarse al publicador y obtener los datos WAL decodificados. Cuando el suscriptor crea la suscripción, se crea la ranura de replicación en el nodo del publicador.

A continuación, encontrará información sobre cómo configurar la extensión `pglogical`.

Temas

- Requisitos y limitaciones de la extensión pglogical
- Configuración de la extensión pglogical
- Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL
- Restablecimiento de la replicación lógica después de una actualización principal
- Administración de ranuras de replicación lógica para Aurora PostgreSQL
- Referencia de parámetros para la extensión pglogical

Requisitos y limitaciones de la extensión pglogical

Todas las versiones disponibles actualmente de Aurora PostgreSQL admiten la extensión `pglogical`.

Tanto el nodo publicador como el nodo suscriptor deben estar configurados para la replicación lógica.

Las tablas que desee replicar desde un publicador a un suscriptor deben tener los mismos nombres y el mismo esquema. Estas tablas también deben contener las mismas columnas y las columnas deben utilizar los mismos tipos de datos. Tanto las tablas de los publicadores como las de suscriptores deben tener las mismas claves principales. Se recomienda utilizar únicamente la PRIMARY KEY como restricción única.

Las tablas del nodo suscriptor pueden tener restricciones más permisivas que las del nodo publicador para las restricciones CHECK y NOT NULL.

La extensión `pglogical` proporciona funciones como la replicación bidireccional que no son compatibles con la función de replicación lógica integrada en PostgreSQL (versión 10 y posteriores). Para obtener más información, consulte PostgreSQL bi-directional replication using pglogical (Replicación bidireccional de PostgreSQL mediante pglogical).

Configuración de la extensión pglogical

Para configurar la extensión `pglogical` en el clúster de base de datos de Aurora PostgreSQL, añada `pglogical` a las bibliotecas compartidas en el grupo de parámetros de clúster de bases de datos personalizado para su clúster de bases de datos de Aurora PostgreSQL. También debe establecer el valor del parámetro `rds.logical_replication` en `1` para activar la descodificación lógica. Por último, cree la extensión en la base de datos. Puede utilizar la Consola de administración de AWS o la AWS CLI para estas tareas.

Debe tener permisos como el rol `rds_superuser` para realizar estas tareas.

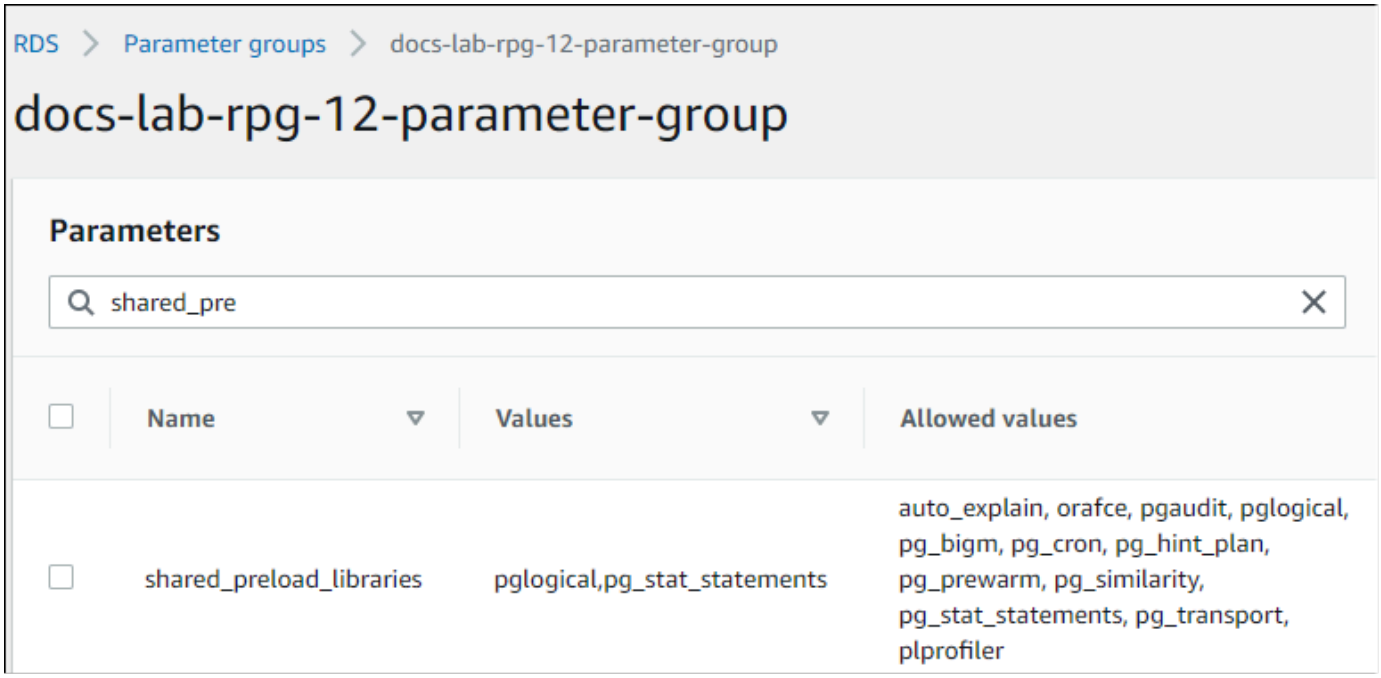
En los pasos siguientes se supone que el clúster de ase de datos de Aurora PostgreSQL está asociado a un grupo de parámetros de clúster de base de datos. Para obtener información acerca de cómo crear el grupo de parámetros del clúster de base de datos, consulte Grupos de parámetros para Amazon Aurora.

Consola

Para configurar la extensión pglogical

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija la instancia de escritor del clúster de base de datos de Aurora PostgreSQL .

3. Abra la pestaña Configuration (Configuración) para su instancia de escritor del clúster de base de datos de Aurora PostgreSQL. Entre los detalles de la instancia, busque el enlace del grupo de parámetros.
4. Elija el enlace para abrir los parámetros personalizados asociados al clúster de base de datos de Aurora PostgreSQL.
5. En el campo de búsqueda Parametes (Parámetros), escriba `shared_pre` para buscar el parámetro `shared_preload_libraries`.
6. Seleccione Edit parameters (Editar parámetros) para acceder a los valores de las propiedades.
7. Añada `pglogical` a la lista en el campo Values (Valores). Utilice una coma para separar los elementos de la lista de valores.



8. Busque el parámetro `rds.logical_replication` y configúrelo en 1 para activar la replicación lógica.
9. Reinicie la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que surtan efecto los cambios.
10. Cuando la instancia esté disponible, puede usar `psql` (o pgAdmin) para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. Para comprobar que pglogical esté inicializado, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pglogical
(1 row)
```

12. Compruebe la configuración que permite la descodificación lógica, de la siguiente manera.

```
SHOW wal_level;
wal_level
-----------
 logical
(1 row)
```

13. Cree la extensión de la siguiente manera.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. Elija Guardar cambios.
15. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
16. Elija la instancia de escritura del clúster de RDS for PostgreSQL en la lista de bases de datos para seleccionarla y, a continuación, elija Reboot (Reiniciar) en el menú Actions (Acciones).

AWS CLI

Para configurar la extensión pglogical

Para configurar pglogical mediante la AWS CLI, llame a la operación modify-db-parameter-group para modificar determinados parámetros de su grupo de parámetros personalizado, tal como se muestra en el siguiente procedimiento.

1. Utilice el siguiente comando AWS CLI para añadir `pglogical` al parámetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
```

```
   --region aws-region
```

2. Utilice el siguiente comando AWS CLI para configurar `rds.logical_replication` en 1 y activar la función de descodificación lógica para la instancia de escritor del clúster de base de datos de Aurora PostgreSQL.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
   --region aws-region
```

3. Utilice el siguiente comando AWS CLI para reiniciar la instancia de escritor de la instancia de base de datos de Aurora PostgreSQL para que se inicialice la biblioteca de pglogical.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

4. Cuando la instancia esté disponible, use `psql` para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. Cree la extensión de la siguiente manera.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. Reinicie la instancia de escritor de su clúster de bases de datos Aurora PostgreSQL mediante el siguiente comando AWS CLI.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Configuración de la replicación lógica para el clúster de base de datos de Aurora PostgreSQL

En el siguiente procedimiento, se muestra cómo iniciar la replicación lógica entre dos clústeres de base de datos de Aurora PostgreSQL. En los pasos, se asume que tanto el origen (publicador) como el destino (suscriptor) tienen la extensión `pglogical` configurada como se detalla en Configuración de la extensión pglogical.

> **Note**
>
> El `node_name` de un nodo de suscriptor no puede empezar por `rds`.

Para crear el nodo publicador y definir las tablas que se van a replicar

En estos pasos se asume que el clúster de base de datos de Aurora PostgreSQL tiene una instancia de escritor con una base de datos que tiene una o más tablas que desea replicar en otro nodo. Debe volver a crear la estructura de tablas del publicador en el suscriptor, así que primero debe obtener la estructura de la tabla si es necesario. Para ello, utilice el metacomando de `psql \d` *`tablename`* y, a continuación, cree la misma tabla en la instancia del suscriptor. El siguiente procedimiento crea una tabla de ejemplo en el publicador (origen) con fines de demostración.

1. Utilice `psql` para conectarse a la instancia que tiene la tabla que desea usar como origen para los suscriptores.

   ```
   psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=labdb
   ```

   Si no dispone de una tabla existente que desee replicar, puede crear una tabla de ejemplo de la siguiente manera.

   a. Cree una tabla de ejemplo con la siguiente instrucción SQL.

      ```
      CREATE TABLE docs_lab_table (a int PRIMARY KEY);
      ```

   b. Rellene la tabla con los datos generados mediante la siguiente instrucción SQL.

      ```
      INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
      INSERT 0 5000
      ```

c. Compruebe que los datos existen en la tabla mediante la siguiente instrucción SQL.

```
SELECT count(*) FROM docs_lab_table;
```

2. Identifique este clúster de base de datos de Aurora PostgreSQL como nodo publicador de la siguiente manera.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_provider',
    dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 dbname=labdb');
 create_node
-------------
   3410995529
(1 row)
```

3. Añada la tabla que desea replicar al conjunto de replicación predeterminado. Para obtener más información sobre los conjuntos de replicación, consulte Replication sets (Conjuntos de replicación) en la documentación de pglogical.

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  --------------------------
  t
  (1 row)
```

Se ha completado la configuración del nodo publicador. Ahora puede configurar el nodo suscriptor para recibir las actualizaciones del publicador.

Para configurar el nodo suscriptor y crear una suscripción para recibir actualizaciones

En estos pasos se asume que el clúster de base de datos de Aurora PostgreSQL se ha configurado con la extensión `pglogical`. Para obtener más información, consulte Configuración de la extensión pglogical.

1. Utilice `psql` para conectarse a la instancia en la que desea recibir actualizaciones del publicador.

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. En el clúster de base de datos de Aurora PostgreSQL, del suscriptor, cree la misma tabla que existe en el publicador. En este ejemplo, la tabla es `docs_lab_table`. Puede crear la tabla tal y como se indica a continuación.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. Compruebe que esta tabla esté vacía.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. Identifique este clúster de base de datos de Aurora PostgreSQL como nodo suscriptor de la siguiente manera.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. Cree la suscripción.

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
```

```
(1 row)
```

Al completar este paso, los datos de la tabla del publicador se crean en la tabla del suscriptor. Para comprobar que ha ocurrido esto, utilice la siguiente consulta SQL.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
(1 row)
```

A partir de este momento, los cambios realizados en la tabla del publicador se replicarán en la tabla del suscriptor.

Restablecimiento de la replicación lógica después de una actualización principal

Para poder realizar una actualización de una versión principal de un clúster de base de datos de Aurora PostgreSQL que se haya configurado como nodo publicador para la replicación lógica, debe eliminar todas las ranuras de replicación, incluso las que no estén activas. Se recomienda desviar temporalmente las transacciones de la base de datos del nodo publicador, eliminar las ranuras de replicación, actualizar el clúster de base de datos de Aurora PostgreSQL, y, a continuación, restablecer y reiniciar la replicación.

Las ranuras de replicación se alojan únicamente en el nodo publicador. El nodo suscriptor de Aurora PostgreSQL en un escenario de replicación lógica no tiene ranuras que eliminar. El proceso de actualización de la versión principal de Aurora PostgreSQL permite actualizar el suscriptor a una nueva versión principal de PostgreSQL independiente del nodo publicador. Sin embargo, el proceso de actualización interrumpe el proceso de replicación e interfiere con la sincronización de los datos WAL entre el nodo publicador y el nodo suscriptor. Debe restablecer la replicación lógica entre el publicador y el suscriptor después de actualizar el publicador, el suscriptor o ambos. En el procedimiento siguiente se muestra cómo determinar que se ha interrumpido la replicación y cómo resolver el problema.

Determinación de que la replicación lógica se ha interrumpido

Puede determinar que el proceso de replicación se ha interrumpido consultando el nodo publicador o el nodo suscriptor de la siguiente manera.

Para comprobar el nodo publicador

- Utilice `psql` para conectarse al nodo publicador y, a continuación, consulte la función `pg_replication_slots`. Anote el valor de la columna activa. Normalmente, esto devolverá `t` (true) y mostrará que la replicación está activa. Si la consulta devuelve `f` (false), indica que la replicación en el suscriptor se ha detenido.

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

Para comprobar el nodo suscriptor

En el nodo suscriptor, puede comprobar el estado de la replicación de tres maneras diferentes.

- Revise los registros de PostgreSQL en el nodo suscriptor para encontrar los mensajes de error. El registro identifica el error con mensajes que incluyen el código de salida 1, como se muestra a continuación.

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- Consulte la función `pg_replication_origin`. Conéctese a la base de datos en el nodo suscriptor mediante `psql` y consulte la función `pg_replication_origin` de la siguiente manera.

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

  Un conjunto de resultados vacío significa que la replicación se ha interrumpido. Debería ver una salida como la siguiente.

```
   roident |                      roname
```

```
---------+-----------------------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

- Consulte la función `pglogical.show_subscription_status` tal y como se muestra en el siguiente ejemplo.

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
-------+----------------+--------+-----------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

  Este resultado muestra que la replicación se ha interrumpido. Su estado es `down`. Normalmente, la salida muestra el estado como `replicating`.

Si el proceso de replicación lógica se ha interrumpido, puede restablecerla siguiendo estos pasos.

Para restablecer la replicación lógica entre los nodos publicador y suscriptor

Para restablecer la replicación, primero debe desconectar el suscriptor del nodo publicador y, a continuación, restablecer la suscripción, tal como se describe en estos pasos.

1. Conéctese al nodo suscriptor con `psql` de la siguiente manera.

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Desactive la suscripción mediante la función `pglogical.alter_subscription_disable`.

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. Obtenga el identificador del nodo publicador consultando el `pg_replication_origin` de la siguiente manera.

```
SELECT * FROM pg_replication_origin;
 roident |               roname
```

```
---------+-------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. Utilice la respuesta del paso anterior con el comando `pg_replication_origin_create` para asignar el identificador que podrá utilizar la suscripción cuando se restablezca.

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. Para activar la suscripción, introduzca su nombre con un estado de `true`, tal como se muestra en el ejemplo siguiente.

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
 t
(1 row)
```

Compruebe el estado del nodo. Su estado debería ser `replicating`, tal y como se muestra en este ejemplo.

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+-------------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

Compruebe el estado de la ranura de replicación del suscriptor en el nodo publicador. La columna `active` de la ranura debe devolver `t` (true), lo que indica que se ha restablecido la replicación.

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                   slot_name              |      plugin      | slot_type | active
------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
```

```
(1 row)
```

Administración de ranuras de replicación lógica para Aurora PostgreSQL

Para poder realizar una actualización de una versión principal en una instancia de escritor de base de datos de Aurora PostgreSQL que se utilice como nodo publicador en un escenario de replicación lógica, debe eliminar todas las ranuras de replicación de la instancia. El proceso de comprobación previa de la actualización de la versión principal le indica que la actualización no puede continuar hasta que se eliminen las ranuras disponibles.

Para identificar las ranuras de replicación que se crearon con la extensión `pglogical`, inicie sesión en cada base de datos y obtenga el nombre de los nodos. Al consultar el nodo suscriptor, aparecen los nodos publicador y suscriptor en el resultado, tal como se muestra en este ejemplo.

```
SELECT * FROM pglogical.node;
node_id   |     node_name
------------+------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
(2 rows)
```

Puede obtener los detalles de la suscripción con la siguiente consulta.

```
SELECT sub_name,sub_slot_name,sub_target
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

Ahora puede eliminar la suscripción de la siguiente manera.

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

Después de eliminar la suscripción, puede eliminar el nodo.

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
```

```
 drop_node
-----------
 t
(1 row)
```

Puede comprobar que el nodo ya no existe de la siguiente manera.

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

Referencia de parámetros para la extensión pglogical

En la tabla verá los parámetros asociados a la extensión `pglogical`. Parámetros como `pglogical.conflict_log_level` y `pglogical.conflict_resolution` se utilizan para gestionar los conflictos de actualización. Pueden surgir conflictos cuando los cambios se realizan localmente en las mismas tablas que están suscritas a los cambios del publicador. Los conflictos también pueden producirse en varios escenarios, como la replicación bidireccional o cuando varios suscriptores replican desde el mismo publicador. Para obtener más información, consulte PostgreSQL bi-directional replication using pglogical (Replicación bidireccional de PostgreSQL mediante pglogical).

Parámetro	Descripción
pglogical.batch_inserts	Inserciones por lotes si es posible. No establecido de manera predeterminada. Se cambia a 1 para activarlo y a 0 para desactivarlo.
pglogical.conflict_log_level	Establece el nivel de registro utilizado para registrar los conflicto s resueltos. Los valores permitidos son debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal y panic.
pglogical.conflict_resolution	Establece el método que se utilizará para resolver conflictos si es posible resolverlos. Los valores de cadena admitidos son error, apply_remote, keep_local, last_update_wins y first_upd ate_wins.

Parámetro	Descripción
pglogical.extra_connection_options	Opciones de conexión a añadir a todas las conexiones de los nodos pares.
pglogical.synchronous_commit	valor de confirmación sincrónica específico de pglogical
pglogical.use_spi	Utilice la SPI (interfaz de programación de servidores) en lugar de la API de nivel inferior para aplicar los cambios. Se establece en 1 para activarlo y en 0 para desactivarlo. Para obtener más información acerca de la SPI, consulte Server Programming Interface (Interfaz de programación de servidores) en la documentación de PostgreSQL.

Uso de los contenedores de datos externos compatibles para Amazon Aurora PostgreSQL

Un FDW es un tipo específico de extensión que proporciona acceso a datos externos. Por ejemplo, la extensión `oracle_fdw` permite a su Instancia de base de datos de Aurora PostgreSQL trabajar con bases de datos Oracle.

A continuación, puede encontrar información sobre varios contenedores de datos externos de PostgreSQL compatibles.

Temas

- Uso de la extensión log_fdw para acceder al registro de base de datos mediante SQL
- Uso de la extensión postgres_fdw para acceder a datos externos
- Uso de bases de datos MySQL con la extensión mysql_fdw
- Uso de una base de datos de Oracle con la extensión oracle_fdw
- Uso de bases de datos de SQL Server con la extensión mysql_fdw

Uso de la extensión log_fdw para acceder al registro de base de datos mediante SQL

El clúster de base de datos de Aurora PostgreSQL admite la extensión `log_fdw`, que se puede utilizar para el acceso al registro del motor de base de datos a través de una interfaz SQL. La

extensión `log_fdw` proporciona dos funciones que facilitan la creación de tablas externas para los registros de la base de datos:

- `list_postgres_log_files`: muestra los archivos del directorio de registro de la base de datos y el tamaño del archivo en bytes.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)`: crea una tabla externa para el archivo especificado en la base de datos actual.

Todas las funciones creadas por `log_fdw` pertenecen a `rds_superuser`. Los miembros del rol `rds_superuser` pueden conceder acceso a estas funciones a otros usuarios de la base de datos.

De forma predeterminada, Amazon Aurora genera los archivos de registro en formato `stderr` (error estándar), como se especifica en el parámetro `log_destination`. Solo hay dos opciones para este parámetro: `stderr` y `csvlog` (valores separados por comas, CSV). Si se añade la opción `csvlog` al parámetro, Amazon Aurora generará tanto el registro `stderr` como el registro `csvlog`. Esto puede afectar a la capacidad de almacenamiento del clúster de base de datos, por lo que debe tener en cuenta los demás parámetros que afectan a la gestión de jlos registros. Para obtener más información, consulte Configuración del destino del registro (`stderr`, `csvlog`).

Uno de los beneficios de generar registros `csvlog` es que la extensión `log_fdw` permite crear tablas externas con los datos perfectamente divididos en varias columnas. Para ello, la instancia debe asociarse a un grupo de parámetros de base de datos personalizado para que usted pueda cambiar la configuración de `log_destination`. Para obtener información acerca de cómo hacerlo, consulte Grupos de parámetros para Amazon Aurora.

En el ejemplo siguiente se presupone que el parámetro `log_destination` incluye `cvslog`.

Para utilizar la extensión log_fdw

1. Instale la extensión de `log_fdw`.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Cree el servidor de registros como contenedor de datos externo.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Seleccione todos los elementos de una lista de archivos de registro.

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

A continuación, se muestra una respuesta de ejemplo.

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
(4 rows)
```

4. Crear una tabla con una sola columna log_entry para el archivo seleccionado.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

La respuesta no proporciona más detalles que el hecho de que la tabla ya existe.

```
-----------------------------------
(1 row)
```

5. Seleccione una muestra del archivo de registro. El siguiente código recupera la hora del registro y la descripción del mensaje de error.

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

A continuación, se muestra una respuesta de ejemplo.

```
             log_time             |                                 message
----------------------------------
+------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
```

```
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

Uso de la extensión postgres_fdw para acceder a datos externos

Puede acceder a los datos en una tabla en un servidor de base de datos remoto con la extensión postgres_fdw. Si establece una conexión remota desde su instancia de base de datos de PostgreSQL, el acceso también está disponible para su réplica de lectura.

Para utilizar postgres_fdw para acceder a un servidor de base de datos remoto

1. Instale la extensión postgres_fdw.

```
CREATE EXTENSION postgres_fdw;
```

2. Cree el servidor de datos externo utilizando CREATE SERVER.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. Cree un mapeo de usuario para identificar la función que utilizar en el servidor remoto.

> **Important**
>
> Para redactar la contraseña para que no aparezca en los registros, configure `log_statement=none` en el nivel de sesión. Si se establece en el nivel de parámetro, no se redacta la contraseña.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. Cree una tabla que se mapee a la tabla del servidor remoto.

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
```

```
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

Uso de bases de datos MySQL con la extensión mysql_fdw

Para tener acceso desde un clúster de base de datos de Aurora PostgreSQL a una base de datos compatible con MySQL, puede instalar y utilizar la extensión `mysql_fdw`. Este contenedor de datos externo le permite trabajar con RDS for MySQL, Aurora MySQL, MariaDB y otras bases de datos compatibles con MySQL. La conexión desde el clúster de base de datos de Aurora PostgreSQL a la base de datos MySQL se cifra tanto como sea posible, depeniendo de la configuración del cliente y del servidor. No obstante, puede aplicar cifrado si lo desea. Para obtener más información, consulte Uso de cifrado en tránsito con la extensión.

La extensión `mysql_fdw` es complatible con las versiones de Amazon Aurora PostgreSQL 15.4, 14,9, 13.12, 12.16 y posteriores. Es compatible con selecciones, inserciones, actualizaciones y eliminaciones de una base de datos de RDS for PostgreSQL en tablas de una instancia de base de datos compatible con MySQL.

Temas

- Configuración de una base de datos de Aurora PostgreSQL para utilizar la extensión mysql_fdw
- Ejemplo: Acceso a una base de datos de Aurora MySQL desde Aurora PostgreSQL
- Uso de cifrado en tránsito con la extensión

Configuración de una base de datos de Aurora PostgreSQL para utilizar la extensión mysql_fdw

Para configurar la extensión `mysql_fdw` en el clúster de base de datos de Aurora PostgreSQL es necesario cargar la extensión en el clúster y, a continuación, crear el punto de conexión a la instancia de base de datos MySQL. Para esa tarea debe disponer de los siguientes detalles sobre la instancia de base de datos MySQL:

- Nombre de host o del punto de conexión. Con un clúster de base de datos de Aurora MySQL el punto de conexión puede encontrarse a través de la consola. Elija la pestaña Conectividad y seguridad y busque en la sección “Punto de enlace y puerto”.
- Número de puerto. El número de puerto predeterminado para MySQL es 3306.
- Nombre de la base de datos. El identificador de la base de datos.

También tiene que proporcionar acceso en el grupo de seguridad o en la lista de control de acceso (ACL) para el puerto MySQL, 3306. Tanto el clúster de base de datos de Aurora PostgreSQL como el de Aurora MySQL necesitan acceso al puerto 3306. Si el acceso no está configurado correctamente, al intentar conectarse a una tabla compatible con MySQL aparecerá un mensaje de error similar al siguiente:

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-
region.rds.amazonaws.com:3306' (110)
```

En el procedimiento que sigue, usted (como cuenta de `rds_superuser`) crea el servidor externo. A continuación, concede acceso al servidor externo a usuarios específicos. A continuación, estos usuarios crean sus propias asignaciones a las cuentas de usuario de MySQL adecuadas para trabajar con la instancia de base de datos MySQL.

Para utilizar mysql_fdw para acceder a un servidor de base de datos MySQL

1. Conéctese a la instancia de base de datos PostgreSQL a través de una cuenta que tenga el rol de `rds_superuser`. Si al crear el clúster de base de datos de Aurora PostgreSQL aceptó los valores predeterminados, el nombre de usuario será `postgres` y se podrá conectar mediante la herramienta de línea de comandos `psql` de este modo:

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Instale la extensión `mysql_fdw` de la siguiente manera:

   ```
   postgres=> CREATE EXTENSION mysql_fdw;
   CREATE EXTENSION
   ```

Una vez instalada la extensión en el clúster de base de datos de Aurora PostgreSQL, configure el servidor externo que proporciona la conexión a una base de datos MySQL.

Para crear el servidor externo

Realice estas tareas en el clúster de bases de datos de Aurora PostgreSQL. Para seguir estos pasos se entiende que está conectado como usuario con privilegios `rds_superuser`, como `postgres`.

1. Cree un servidor externo en el clúster de bases de datos Aurora PostgreSQL:

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-
name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Conceda a los usuarios que corresponsa acceso al servidor externo. Deben ser usuarios que no sean administradores, es decir, usuarios que no tengan el rol `rds_superuser`.

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

Los usuarios de PostgreSQL crean y administran sus propias conexiones a la base de datos MySQL a través del servidor externo.

Ejemplo: Acceso a una base de datos de Aurora MySQL desde Aurora PostgreSQL

Supongamos que tiene una tabla simple en una instancia de base de datos de Aurora PostgreSQL. Los usuarios de Aurora PostgreSQL desean consultar (`SELECT`), insertar (`INSERT`), actualizar (`UPDATE`) y eliminar (`DELETE`) elementos de la tabla. Supongamos que la exstensión `mysql_fdw` se creó en la instancia de base de datos de RDS for PostgreSQL, como se detalla en el procedimiento anterior. Después de conectarse a la instancia de base de datos de RDS for PostgreSQL como usuario con privilegios `rds_superuser`, podrá continuar con los pasos que se describen a continuación.

1. Cree un servidor externo en la instancia de base de datos de Aurora PostgreSQL:

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-
DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Conceda permiso de uso a un usuario que no tiene permisos `rds_superuser`, por ejemplo `user1`.

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. Conéctese como *user1* y, a continuación, cree una asignación para el usuario de MySQL:

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
```

```
CREATE USER MAPPING
```

4. Cree una tabla externa vinculada a la tabla MySQL:

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. Ejecute una consulta simple en la tabla externa:

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. Puede añadir, modificar y quitar datos de la tabla MySQL. Por ejemplo:

```
test=> INSERT INTO mytab values (2, 'mango');
INSERT 0 1
```

   Ejecute la consulta `SELECT` de nuevo para ver los resultados:

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
(2 rows)
```

Uso de cifrado en tránsito con la extensión

De forma predeterminada, la conexión a MySQL desde Aurora PostgreSQL utiliza cifrado en tránsito (TLS/SSL). No obstante, la conexión vuelve a ser no cifrada cuando la configuración del cliente y del servidor difieren. Puede aplicar el cifrado para todas las conexiones salientes especificando la opción `REQUIRE SSL` en las cuentas de usuario de RDS for MySQL. Este mismo método también funciona para las cuentas de usuario de MariaDB y Aurora MySQL.

Para cuentas de usuario MySQL configuradas en `REQUIRE SSL`, el intento de conexión falla si no se puede establecer una conexión segura.

Para aplicar el cifrado de cuentas de usuario de bases de datos MySQL existentes, puede utilizar el comando `ALTER USER`. La sintaxis varía en función de la versión de MySQL, como se muestra en la siguiente tabla. Para obtener más información, consulte ALTER USER en el manual de referencia de MySQL.

MySQL 5.7, MySQL 8.0	MySQL 5.6
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

Para obtener más información acerca de la extensión `mysql_fdw`, consulte la documentación sobre mysql_fdw.

Uso de una base de datos de Oracle con la extensión oracle_fdw

Para acceder a una base de datos de Oracle desde su clúster de bases de datos de Aurora PostgreSQL puede instalar y utilizar la extensión `oracle_fdw`. Esta extensión es un contenedor de datos externos para bases de datos Oracle. Para obtener más información sobre la extensión, consulte la documentación de oracle_fdw.

La extensión `oracle_fdw` es compatible con Aurora PostgreSQL 12.7, (Amazon Aurora PostgreSQL versión 4.2) y versiones posteriores.

Temas

- Activación de la extensión oracle_fdw
- Ejemplo: Usar un servidor externo vinculado a una Amazon RDS for Oracle Database
- Trabajo con cifrado en tránsito
- Comprensión y permisos de la vista pg_user_mappings

Activación de la extensión oracle_fdw

Para utilizar la extensión oracle_fdw, lleve a cabo el siguiente procedimiento.

Para habilitar la extensión oracle_fdw

- Ejecute el siguiente comando con una cuenta que tenga los permisos `rds_superuser`.

```
CREATE EXTENSION oracle_fdw;
```

Ejemplo: Usar un servidor externo vinculado a una Amazon RDS for Oracle Database

El siguiente ejemplo muestra el uso de un servidor externo vinculado a una base de datos de Amazon RDS for Oracle.

Crear un servidor externo vinculado a una base de datos de RDS for Oracle

1. Tenga en cuenta lo siguiente en la instancia de base de datos de RDS for Oracle:
   - punto de enlace
   - Puerto
   - Nombre de base de datos
2. Cree un servidor externo.

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. Otorgue uso a un usuario que no tenga permisos `rds_superuser`, por ejemplo `user1`.

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. Conéctese como `user1` y cree una asignación a un usuario de Oracle.

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Cree una tabla externa vinculada a una tabla de Oracle.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
CREATE FOREIGN TABLE
```

6. Consulte la tabla externa.

```
test=>  SELECT * FROM mytab;
```

```
a
---
1
(1 row)
```

Si la consulta informa el siguiente error, verifique el grupo de seguridad y la lista de control de acceso (ACL) para asegurarse de que ambas instancias puedan comunicarse.

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

Trabajo con cifrado en tránsito

El cifrado de PostgreSQL a Oracle en tránsito se basa en una combinación de parámetros de configuración de cliente y servidor. Para obtener un ejemplo que utiliza Oracle 21c, consulte About the Values for Negotiating Encryption and Integrity en la documentación de Oracle. El cliente utilizado para oracle_fdw en Amazon RDS está configurado con `ACCEPTED`, lo que significa que el cifrado depende de la configuración del servidor de base de datos de Oracle y utiliza la biblioteca de seguridad de Oracle (libnnz) para cifrado.

Si su base de datos está en RDS for Oracle, consulte Oracle Native Network Encryption para configurar el cifrado.

Comprensión y permisos de la vista pg_user_mappings

El catálogo de PostgreSQL `pg_user_mapping` almacena la asignación desde un usuario Aurora PostgreSQL en el usuario de un servidor de datos externo (remoto). El acceso al catálogo está restringido, pero usted utiliza la vista `pg_user_mappings` para ver las asignaciones. A continuación, se muestra un ejemplo sobre cómo se aplican los permisos en una base de datos de Oracle de ejemplo, aunque esta información es válida también en general para cualquier contenedor de datos externo.

En el siguiente resultado, puede encontrar roles y permisos asignados a tres usuarios de ejemplo diferentes. Usuarios de `rdssu1` y `rdssu2` son miembros del rol `rds_superuser`, y el usuario `user1` no lo es. En el ejemplo se usa el metacomando `\du` de `psql` para enumerar los roles existentes.

```
test=>  \du
                                                    List of roles
```

```
    Role name    |                         Attributes                         |
                  Member of
-----------------+------------------------------------------------------------
+-----------------------------------------------------------
 rdssu1          |                                                            |
 {rds_superuser}
 rdssu2          |                                                            |
 {rds_superuser}
 user1           |                                                            | {}
```

Todos los usuarios, incluidos los usuarios con privilegios `rds_superuser`, pueden ver sus propias asignaciones de usuarios (`umoptions`) en la tabla `pg_user_mappings`. Como se muestra en el siguiente ejemplo, cuando `rdssu1` intenta obtener todas las asignaciones de usuario, se genera un a pesar de los privilegios `rds_superuser` de `rdssu1`:

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

A continuación, se muestran algunos ejemplos:

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)

test=> SET SESSION AUTHORIZATION user1;
SET
```

```
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

Debido a las diferencias en la implementación de `information_schema._pg_user_mappings` y `pg_catalog.pg_user_mappings`, un `rds_superuser` que se crea manualmente requiere permisos adicionales para ver las contraseñas en `pg_catalog.pg_user_mappings`.

No requieren otros permisos para un `rds_superuser` para ver las contraseñas en `information_schema._pg_user_mappings`.

Los usuarios que no tienen el rol `rds_superuser` pueden ver contraseñas en `pg_user_mappings` solo en las condiciones que se describen a continuación:

- El usuario actual es el usuario que se está asignando y es el propietario del servidor o tiene el privilegio de `USAGE` en él.
- El usuario actual es el propietario del servidor, y la asignación es para `PUBLIC`.

Uso de bases de datos de SQL Server con la extensión mysql_fdw

Puede utilizar la extensión de PostgreSQL `tds_fdw` para acceder a bases de datos compatibles con el protocolo de flujo de datos tabular (TDS), como bases de datos Sybase y Microsoft SQL Server. Este contenedor de datos externo le permite conectarse desde su clúster de bases de datos Aurora PostgreSQL a bases de datos que utilizan el protocolo TDS, incluido Amazon RDS for Microsoft SQL Server. Para obtener más información, consulte la documentación sobre tds-fdw/tds_fdw en GitHub.

La extensión `tds_fdw` es compatible con las versiones 13.6 y posteriores de Aurora PostgreSQL.

Configuración de la base de datos de Aurora PostgreSQL para utilizar la extensión mysql_fdw

En los procedimientos que siguen encontrará un ejemplo de configuración y uso de `tds_fdw` con un clúster de bases de datos Aurora PostgreSQL. Antes de poder conectarse a una base de datos SQL Server mediante `tds_fdw`, tiene que obtener los siguientes detalles de la instancia:

- Nombre de host o del punto de conexión. Para instancias de RDS for MySQL encontrará los puntos de conexión con la consola. Elija la pestaña Conectividad y seguridad y busque en la sección “Punto de enlace y puerto”.
- Número de puerto. El puerto 1433 es el predeterminado para Microsoft SQL Server.
- Nombre de la base de datos. El identificador de la base de datos.

También deberá proporcionar acceso en el grupo de seguridad o en la lista de control de acceso (ACL) al puerto MySQL, 1433. Tanto como la instancia de base de datos RDS for MySQL Server necesitan poder acceder al puerto 1433. Si el acceso no está configurado correctamente, cuando intente consultar Microsoft SQL Server aparecerá el siguiente mensaje de error:

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

Para usar tds_fdw para conectarse a una base de datos de SQL Server

1. Conéctese a su la instancia principal del clúster de bases de datos de Aurora PostgreSQL con una cuenta con rol `rds_superuser`:

   ```
   psql --host=your-cluster-name-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=test --password
   ```

2. Instale la extensión `tds_fdw`.

   ```
   test=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

Después de instalar la extensión en su clúster de bases de datos Aurora PostgreSQL , configure el servidor externo.

Para crear el servidor externo

Realice estas tareas en el clúster de bases de datos Aurora PostgreSQL con una cuenta que con privilegios `rds_superuser`.

1. Cree un servidor externo en el clúster de bases de datos Aurora PostgreSQL:

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
 (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
 'tds_fdw_testing');
CREATE SERVER
```

Para acceder a datos no que sean ASCII en el lado de SQLServer, cree un enlace de servidor con la opción character_set en el clúster de base de datos de Aurora PostgreSQL:

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
 character_set 'UTF-8');
CREATE SERVER
```

2. Conceda permisos a un usuario que no tenga los privilegios del rol `rds_superuser`, por ejemplo `user1`:

```
test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
```

3. Conéctese como user1 y, a continuación, cree una asignación para el usuario de SQL Server:

```
test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
 'sqlserveruser', password 'password');
CREATE USER MAPPING
```

4. Cree una tabla externa vinculada a una tabla de SQL Server.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. Consulte la tabla externa:

```
test=> SELECT * FROM mytab;
 a
---
 1
(1 row)
```

Uso de cifrado en tránsito para la conexión

La conexión de Aurora PostgreSQLa SQL Server utiliza cifrado en tránsito (TLS/SSL) según la configuración de la base de datos de SQL Server. Si SQL Server no está configurado para el cifrado, el RDS para el cliente PostgreSQL que realiza la solicitud a la base de datos de SQL Server vuelve a no ir cifrado.

Puede aplicar el cifrado para la conexión a RDS para instancias de base de datos de SQL Server configurando el parámetro `rds.force_ssl`. Para saber cómo, consulte Requerir que las conexiones a la instancia de base de datos usen SSL. Para obtener más información sobre la configuración de SSL/TLS para RDS for SQL Server, consulte Uso de SSL con una instancia de base de datos de Microsoft SQL Server.

Uso de Extensiones de lenguaje de confianza para PostgreSQL

Extensiones de lenguaje de confianza para PostgreSQL es un kit de desarrollo de código abierto para crear extensiones de PostgreSQL. Le permite crear extensiones de PostgreSQL de alto rendimiento y ejecutarlas de forma segura en su clúster de base de datos de Aurora PostgreSQL. Al utilizar Extensiones de lenguaje de confianza (TLE) para PostgreSQL, puede crear extensiones de PostgreSQL que sigan el enfoque documentado para ampliar la funcionalidad de PostgreSQL. Para obtener más información, consulte el punto Packaging Related Objects into an Extension (Empaquetar objetos relacionados en una extensión) en la documentación de PostgreSQL.

Una ventaja clave de TLE es que se puede utilizar en entornos que no proporcionan acceso al sistema de archivos subyacente a la instancia de PostgreSQL. Anteriormente, la instalación de una nueva extensión requería acceso al sistema de archivos. TLE elimina esta restricción. Pues proporciona un entorno de desarrollo para crear nuevas extensiones para cualquier base de datos de PostgreSQL, incluidas las que se ejecutan en los clústeres de base de datos de Aurora PostgreSQL.

TLE está diseñado para evitar el acceso a recursos no seguros para las extensiones que se crean con TLE. Su entorno de ejecución limita el impacto de cualquier defecto de extensión a una única conexión de base de datos. TLE también proporciona a los administradores de bases de datos un control preciso sobre quién puede instalar las extensiones y proporciona un modelo de permisos para ejecutarlas.

TLE es compatible con la versión 14.5 de Aurora PostgreSQL y versiones posteriores.

El entorno de desarrollo y el entorno de ejecución de Extensiones de lenguaje de confianza se empaquetan como la extensión `pg_tle` de PostgreSQL, versión 1.0.1. Admite la creación de extensiones en JavaScript, Perl, Tcl, PL/pgSQL y SQL. La extensión `pg_tle` se instala en el clúster de base de datos de Aurora PostgreSQL del mismo modo que se instalan otras extensiones de PostgreSQL. Una vez configurada `pg_tle`, los desarrolladores pueden usarla para crear nuevas extensiones de PostgreSQL, conocidas como extensiones TLE.

En los temas siguientes, encontrará información sobre cómo configurar Extensiones de lenguaje de confianza y cómo comenzar a crear sus propias extensiones TLE.

Temas

- Terminología
- Requisitos para usar Extensiones de lenguaje de confianza para PostgreSQL

- Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL
- Información general de Extensiones de lenguaje de confianza para PostgreSQL
- Creación de extensiones TLE para Aurora PostgreSQL
- Eliminar las extensiones TLE de una base de datos
- Desinstalación de Extensiones de lenguaje de confianza para PostgreSQL
- Uso de enlaces de PostgreSQL con sus extensiones TLE
- Referencia de funciones para Extensiones de lenguaje de confianza para PostgreSQL
- Referencia de enlaces para Extensiones de lenguaje de confianza para PostgreSQL

Terminología

Para entender mejor Extensiones de lenguaje de confianza, consulta el siguiente glosario para ver los términos utilizados en este tema.

Extensiones de lenguaje de confianza para PostgreSQL

: Extensiones de lenguaje de confianza para PostgreSQL es el nombre oficial del kit de desarrollo de código abierto que se incluye como extensión `pg_tle`. Está disponible para su uso en cualquier sistema PostgreSQL. Para obtener más información, consulte aws/pg_tle en GitHub.

Extensiones de lenguaje de confianza

: Extensiones de lenguaje de confianza es la versión abreviada de Extensiones de lenguaje de confianza para PostgreSQL. En esta documentación se utilizan el nombre abreviado y sus siglas (TLE).

lenguaje de confianza

: Un lenguaje de confianza es un lenguaje de programación o de scripting que tiene atributos de seguridad específicos. Por ejemplo, los lenguajes de confianza suelen restringir el acceso al sistema de archivos y limitan el uso de las propiedades de red especificadas. El kit de desarrollo TLE está diseñado para ser compatible con lenguajes de confianza. PostgreSQL admite varios lenguajes diferentes que se utilizan para crear extensiones fiables o no fiables. Para ver un ejemplo, consulte el punto Trusted and Untrusted PL/Perl (PL/Perl fiable y no fiable) en la documentación de PostgreSQL. Al crear una extensión con Extensiones de lenguaje de confianza, la extensión utiliza mecanismos de lenguaje de confianza de forma inherente.

Extensión TLE

Una extensión TLE es una extensión de PostgreSQL que se ha creado mediante el kit de desarrollo de Extensiones de lenguaje de confianza (TLE).

Requisitos para usar Extensiones de lenguaje de confianza para PostgreSQL

Estos son los requisitos para configurar y usar el kit de desarrollo TLE.

- Versiones de Aurora PostgreSQL : las extensiones de lenguaje de confianza se admiten en Aurora PostgreSQL versión 14.5 y versiones posteriores únicamente.
  - Si necesita actualizar su clúster de base de datos deAurora PostgreSQL, consulte Actualización de clústeres de base de datos PostgreSQL de Amazon Aurora.
  - Si aún no tiene una instancia de base de datos de Amazon RDS que ejecute PostgreSQL, puede crear una. Para obtener más información, consulte Creación de un clúster de base de datos de Aurora PostgreSQL y conexión a él.
- Requiere privilegios de **`rds_superuser`**: para instalar y configurar la extensión `pg_tle`, el rol de usuario de la base de datos debe tener permisos del rol `rds_superuser`. De forma predeterminada, este rol se otorga al usuario `postgres` que crea el clúster de base de datos de Aurora PostgreSQL.
- Requiere un grupo de parámetros de base de datos personalizado: su clúster de base de datos de Aurora PostgreSQL debe configurarse con un grupo de parámetros de base de datos personalizado. Use el grupo de parámetros de base de datos personalizado para la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.
  - Si su clúster de base de datos de Aurora PostgreSQL no está configurado con un grupo de parámetros de base de datos personalizado, debe crear uno y asociarlo a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL. Para obtener un breve resumen de los pasos, consulte Creación y aplicación de un grupo de parámetros de base de datos personalizado.
  - Si su clúster de base de datos de Aurora PostgreSQL ya se ha configurado con un grupo de parámetros de base de datos personalizado, puede configurar Extensiones de lenguaje de confianza. Para obtener más información, consulte Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL.

Creación y aplicación de un grupo de parámetros de base de datos personalizado

Siga los siguientes pasos para crear un grupo de parámetros de base de datos personalizado y configure su clúster de base de datos de Aurora PostgreSQL para utilizarlo.

Consola

Para crear un grupo de parámetros de base de datos personalizado y utilizarlo con su clúster de base de datos de Aurora PostgreSQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Parameter groups (Grupos de parámetros) en el menú de Amazon RDS.
3. Elija Create parameter group.
4. En la página Parameter group details (Detalles del grupo de parámetros), escriba la siguiente información.
   - En Parameter group family (Familia de grupo de parámetros), elija aurora-postgresql14.
   - En Type (Tipo), elija DB Parameter Group (Grupo de parámetros de bases de datos).
   - En Group name (Nombre de grupo), asigne al grupo de parámetros un nombre significativo en el contexto de sus operaciones.
   - En Description (Descripción), introduzca una descripción útil para que los demás miembros de su equipo puedan encontrarla fácilmente.
5. Seleccione Crear. El grupo de parámetros de base de datos personalizado se crea en su Región de AWS. Ahora puede modificar su clúster de base de datos de Aurora PostgreSQL para usarlo. Para ello, siga los siguientes pasos.
6. Seleccione Databases (Bases de datos) en el menú de Amazon RDS.
7. Elija el clúster de base de datos de Aurora PostgreSQL que desea usar con TLE de entre las enumeradas y, a continuación, elija Modify (Modificar).
8. En la página Modify DB cluster settings (Modificar la configuración del clúster de base de datos), busque Database options (Opciones de la base de datos) y utilice el selector para elegir su grupo de parámetros de base de datos personalizado. En la
9. Elija Continue (Continuar) para guardar el cambio.
10. Elija Apply immediately (Aplicar inmediatamente) para poder seguir configurando el clúster de base de datos de Aurora PostgreSQL para utilizar TLE.

Para continuar con la configuración del sistema para Extensiones de lenguaje de confianza, consulte Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL.

Para obtener más información sobre cómo trabajar con grupos de parámetros de base de datos y de clúster de bases de datos, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

AWS CLI

Puede evitar especificar el argumento `--region` al utilizar los comandos de la CLI al configurar su AWS CLI con su Región de AWS predeterminada. Para obtener más información, consulte Fundamentos de configuración en la Guía del usuario de AWS Command Line Interface.

Para crear un grupo de parámetros de base de datos personalizado y utilizarlo con su clúster de base de datos de Aurora PostgreSQL

1. Utilice el comando create-db-parameter-group de la AWS CLI para crear un grupo de parámetros de base de datos personalizado basado en aurora-postgresql14 para su Región de AWS. Tenga en cuenta que en este paso se crea un grupo de parámetros de base de datos para aplicarlo a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

   Para Linux, macOS o:Unix

   ```
   aws rds create-db-parameter-group \
     --region aws-region \
     --db-parameter-group-name custom-params-for-pg-tle \
     --db-parameter-group-family aurora-postgresql14 \
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   En:Windows

   ```
   aws rds create-db-parameter-group ^
     --region aws-region ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --db-parameter-group-family aurora-postgresql14 ^
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

Su grupo de parámetros de base de datos personalizado está disponible en su Región de AWS, por lo que puede modificar la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para utilizarla.

2. Utilice el comando modify-db-instance de la AWS CLI para aplicar su grupo de parámetros de base de datos personalizado a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL. Este comando reinicia inmediatamente la instancia activa.

   Para Linux, macOS o:Unix

   ```
   aws rds modify-db-instance \
     --region aws-region \
     --db-instance-identifier your-writer-instance-name \
     --db-parameter-group-name custom-params-for-pg-tle \
     --apply-immediately
   ```

   En:Windows

   ```
   aws rds modify-db-instance ^
     --region aws-region ^
     --db-instance-identifier your-writer-instance-name ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --apply-immediately
   ```

Para continuar con la configuración del sistema para Extensiones de lenguaje de confianza, consulte Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL.

Para obtener más información, consulte Grupos de parámetros de base de datos para instancias de Amazon Aurora .

Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL

En los pasos siguientes se supone que su clúster de base de datos de Aurora PostgreSQL está asociado a un grupo de parámetros de clúster de base de datos personalizado. Puede utilizar la Consola de administración de AWS o la AWS CLI para estos pasos.

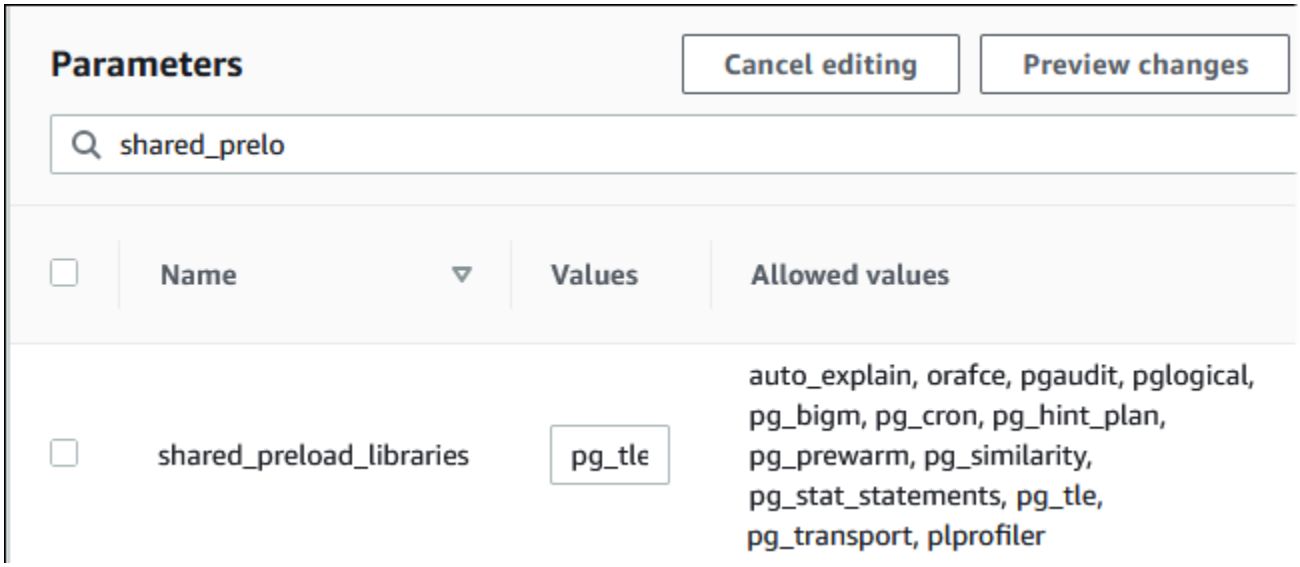
Al configurar Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL, las instala en una base de datos específica para que las usen los usuarios de la base de datos que tienen permisos en esa base de datos.

Consola

Para configurar Extensiones de lenguaje de confianza

Realice los siguientes pasos con una cuenta que sea miembro del grupo (rol) `rds_superuser`.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija la instancia de escritor del clúster de base de datos de Aurora PostgreSQL .
3. Abra la pestaña Configuration (Configuración) para su instancia de escritor del clúster de base de datos de Aurora PostgreSQL. Entre los detalles de la instancia, busque el enlace del grupo de parámetros.
4. Elija el enlace para abrir los parámetros personalizados asociados al clúster de base de datos de Aurora PostgreSQL.
5. En el campo de búsqueda Parametes (Parámetros), escriba `shared_pre` para buscar el parámetro `shared_preload_libraries`.
6. Seleccione Edit parameters (Editar parámetros) para acceder a los valores de las propiedades.
7. Añada `pg_tle` a la lista en el campo Values (Valores). Utilice una coma para separar los elementos de la lista de valores.



8. Reinicie la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL para que surta efecto el cambio en el parámetro `shared_preload_libraries`.
9. Cuando la instancia esté disponible, verifique si se ha inicializado `pg_tle`. Use `psql` para conectarse a la instancia de escritor de su clúster de bases de datos Aurora PostgreSQL, y, a continuación, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. Con la extensión `pg_tle` inicializada, ahora ya puede crear la extensión.

```
CREATE EXTENSION pg_tle;
```

Para comprobar que la extensión esté instalada, use el metacomando `psql`.

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+-------------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. Asigne el rol `pgtle_admin` al nombre de usuario principal que creó para el clúster de base de datos de Aurora PostgreSQL al configurarla. Si ha aceptado el valor predeterminado, es `postgres`.

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

Puede comprobar si se ha realizado la concesión con el metacomando `psql`, tal como se muestra en el siguiente ejemplo. Solo los roles `pgtle_admin` y `postgres` se muestran en el resultado. Para obtener más información, consulte Descripción de los roles y permisos de PostgreSQL.

```
labdb=> \du
                            List of roles
```

```
    Role name    |           Attributes            |              Member of
-----------------+---------------------------------
+---------------------------------
pgtle_admin     | Cannot login                     | {}
postgres        | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                | Password valid until infinity    |...
```

12. Cierre la sesión de `psql` con el metacomando `\q`.

```
\q
```

Para empezar a crear extensiones TLE, consulte Ejemplo: creación de una extensión de lenguaje de confianza mediante SQL.

AWS CLI

Puede evitar especificar el argumento `--region` al utilizar los comandos de la CLI al configurar su AWS CLI con su Región de AWS predeterminada. Para obtener más información, consulte Fundamentos de configuración en la Guía del usuario de AWS Command Line Interface.

Para configurar Extensiones de lenguaje de confianza

1. Use el comando modify-db-cluster-parameter-group de AWS CLI para añadir `pg_tle` al parámetro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Use el comando reboot-db-instance de AWS CLI para reiniciar la instancia de escritor del clúster de base de datos de Aurora PostgreSQL e inicialice la biblioteca de `pg_tle`.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Cuando la instancia esté disponible, puede verificar si `pg_tle` se ha inicializado. Use `psql` para conectarse a la instancia de escritor de su clúster de bases de datos Aurora PostgreSQL, y, a continuación, ejecute el siguiente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

   Con `pg_tle` inicializado, ahora ya puede crear la extensión.

```
CREATE EXTENSION pg_tle;
```

4. Asigne el rol `pgtle_admin` al nombre de usuario principal que creó para el clúster de base de datos de Aurora PostgreSQL al configurarla. Si ha aceptado el valor predeterminado, es `postgres`.

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

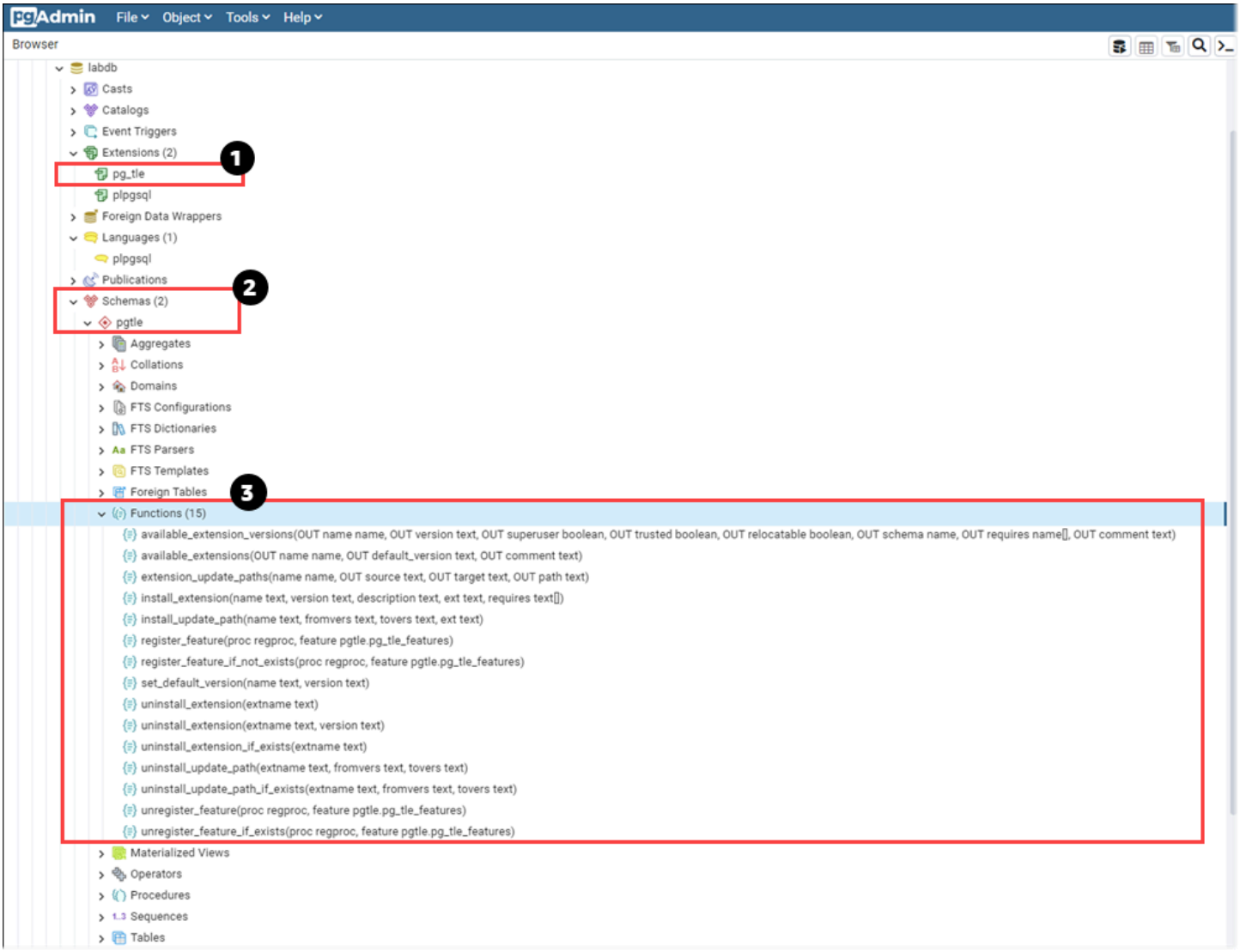
5. Cierre la sesión de `psql` de la siguiente manera.

```
labdb=> \q
```

Para empezar a crear extensiones TLE, consulte Ejemplo: creación de una extensión de lenguaje de confianza mediante SQL.

Información general de Extensiones de lenguaje de confianza para PostgreSQL

Extensiones de lenguaje de confianza para PostgreSQL es una extensión de PostgreSQL que se instala en el clúster de base de datos de Aurora PostgreSQL de la misma manera que se configuran otras extensiones de PostgreSQL. En la siguiente imagen de un ejemplo de base de datos de la herramienta de cliente pgAdmin, puede ver algunos de los componentes que componen la extensión `pg_tle`.



Puede ver los siguientes detalles.

1. El kit de desarrollo de Extensiones de lenguaje de confianza (TLE) está empaquetado como la extensión `pg_tle`. De este modo, `pg_tle` se añade a las extensiones disponibles para la base de datos en la que se instala.
2. TLE tiene su propio esquema: `pgtle`. Este esquema contiene funciones auxiliares (3) para instalar y administrar las extensiones que cree.
3. TLE proporciona más de una docena de funciones auxiliares para instalar, registrar y administrar las extensiones. Para obtener más información sobre estas funciones, consulte Referencia de funciones para Extensiones de lenguaje de confianza para PostgreSQL.

Otros componentes de la extensión `pg_tle` incluyen lo siguiente:

- El rol **`pgtle_admin`**: el rol `pgtle_admin` se crea al instalar la extensión `pg_tle`. Este rol es privilegiado y debe tratarse como tal. Le recomendamos encarecidamente que siga el principio de privilegio mínimo al conceder el rol `pgtle_admin` a los usuarios de la base de datos. En otras palabras, conceda el rol `pgtle_admin` solo a los usuarios de bases de datos que estén autorizados a crear, instalar y administrar nuevas extensiones TLE, como `postgres`.
- La tabla **`pgtle.feature_info`**: la tabla `pgtle.feature_info` es una tabla protegida que contiene información sobre los TLE, los enlaces, los procedimientos y las funciones personalizados almacenados que utilizan. Si tiene privilegios `pgtle_admin`, utilice las siguientes funciones de Extensiones de lenguaje de confianza para añadir y actualizar la información de la tabla.
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

Creación de extensiones TLE para Aurora PostgreSQL

Puede instalar cualquier extensión que cree con TLE en cualquier clúster de base de datos de Aurora PostgreSQL que tenga la extensión `pg_tle` instalada. La extensión `pg_tle` se limita a la base de datos PostgreSQL en la que está instalada. Las extensiones que cree con TLE están incluidas en la misma base de datos.

Utilice las distintas funciones de `pgtle` para instalar el código que conforma la extensión TLE. Todas las siguientes funciones de Extensiones de lenguaje de confianza requieren el rol `pgtle_admin`.

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(nombre, versión)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists

- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

Ejemplo: creación de una extensión de lenguaje de confianza mediante SQL

El siguiente ejemplo muestra cómo crear una extensión TLE denominada `pg_distance` que contenga algunas funciones SQL para calcular distancias mediante diferentes fórmulas. En la lista, puede encontrar la función para calcular la distancia Manhattan y la función para calcular la distancia euclidiana. Para obtener más información sobre la diferencia entre estas fórmulas, consulte Geometría del taxista y Geometría euclidiana en la Wikipedia.

Puede utilizar este ejemplo en su clúster de base de datos de Aurora PostgreSQL si tiene la extensión `pg_tle` configurada como se detalla en Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL.

> **Note**
>
> Debe tener los privilegios del rol `pgtle_admin` para seguir este procedimiento.

Para crear la extensión TLE de ejemplo

En los pasos siguientes se utiliza un ejemplo de base de datos denominado `labdb`. Esta base de datos es propiedad del usuario `postgres` principal. El rol `postgres` también tiene los permisos del rol `pgtle_admin`.

1. Use `psql` para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Cree una extensión TLE denominada `pg_distance` copiando el siguiente código y pegándolo en la consola de sesión de `psql`.

   ```
   SELECT pgtle.install_extension
   (
    'pg_distance',
    '0.1',
   ```

```
  'Distance functions for two points',
$_pg_tle_$
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;

    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

Debería ver un resultado como el siguiente.

```
install_extension
---------------
 t
(1 row)
```

Los artefactos que componen la extensión `pg_distance` ahora ya están instalados en su base de datos. Estos artefactos incluyen el archivo de control y el código de la extensión, que son elementos que deben estar presentes para poder crear la extensión mediante el comando `CREATE EXTENSION`. En otras palabras, aún debe crear la extensión para que sus funciones estén disponibles para los usuarios de la base de datos.

3. Para crear la extensión, utilice el comando `CREATE EXTENSION` como lo haría con cualquier otra extensión. Al igual que con otras extensiones, el usuario de la base de datos debe tener los permisos `CREATE` en la base de datos.

```
CREATE EXTENSION pg_distance;
```

4. Para probar la extensión TLE `pg_distance`, puede utilizarla para calcular la distancia Manhattan entre cuatro puntos.

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

Para calcular la distancia euclidiana entre el mismo conjunto de puntos, puede utilizar lo siguiente.

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
5.656854249492381
```

La extensión `pg_distance` carga las funciones de la base de datos y las pone a disposición de cualquier usuario con permisos en la base de datos.

Modificación de su extensión TLE

Para mejorar el rendimiento de las consultas para las funciones incluidas en esta extensión TLE, añada los dos atributos de PostgreSQL siguientes a sus especificaciones.

- `IMMUTABLE`: el atributo `IMMUTABLE` garantiza que el optimizador de consultas pueda utilizar optimizaciones para mejorar los tiempos de respuesta de las consultas. Para obtener más información, consulte Function Volatility Categories (Categorías de volatilidad de función) en la documentación de PostgreSQL.
- `PARALLEL SAFE`: el atributo `PARALLEL SAFE` es otro atributo que permite a PostgreSQL ejecutar la función en modo paralelo. Para obtener más información, consulte CREATE FUNCTION en la documentación de PostgreSQL.

En el siguiente ejemplo, puede ver cómo se usa la función `pgtle.install_update_path` para agregar estos atributos a cada función a fin de crear una versión `0.2` de la extensión TLE `pg_distance`. Para obtener más información acerca de esta función, consulte pgtle.install_update_path. Debe tener el rol `pgtle_admin` para realizar esta tarea.

Para actualizar una extensión TLE existente y especificar la versión predeterminada

1. Conecte con la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL con `psql` u otra herramienta de cliente como pgAdmin.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. Modifique una extensión TLE existente copiando el siguiente código y pegándolo en la consola de sesión de `psql`.

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

Verá una respuesta similar a la siguiente.

```
install_update_path
---------------------
 t
(1 row)
```

Puede hacer que esta versión de la extensión sea la versión predeterminada para que los usuarios de la base de datos no tengan que especificar una versión al crear o actualizar la extensión en su base de datos.

3. Para especificar que la versión modificada (versión 0.2) de la extensión TLE es la versión predeterminada, utilice la función `pgtle.set_default_version` tal como se muestra en el siguiente ejemplo.

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

Para obtener más información acerca de esta función, consulte pgtle.set_default_version.

4. Con el código en su lugar, puede actualizar la extensión TLE instalada de la forma habitual, mediante el comando `ALTER EXTENSION ... UPDATE`, tal como se muestra aquí:

```
ALTER EXTENSION pg_distance UPDATE;
```

Eliminar las extensiones TLE de una base de datos

Puede eliminar sus extensiones TLE mediante el comando `DROP EXTENSION` de la misma manera que lo hace con otras extensiones de PostgreSQL. Al eliminar la extensión, no se eliminan los archivos de instalación que la componen, lo que permite a los usuarios volver a crearla. Para eliminar la extensión y sus archivos de instalación, realice el siguiente proceso de dos pasos.

Para eliminar la extensión TLE y eliminar sus archivos de instalación

1. Use `psql` u otra herramienta de cliente para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=dbname
```

2. Elimine la extensión tal como haría con cualquier extensión de PostgreSQL.

```
DROP EXTENSION your-TLE-extension
```

Por ejemplo, si crea la extensión `pg_distance` tal como se indica en Ejemplo: creación de una extensión de lenguaje de confianza mediante SQL, puede eliminarla de la siguiente manera.

```
DROP EXTENSION pg_distance;
```

Verá un resultado que confirma que se ha eliminado la extensión, de la siguiente manera.

```
DROP EXTENSION
```

En este punto, la extensión ya no está activa en la base de datos. Sin embargo, sus archivos de instalación y su archivo de control siguen disponibles en la base de datos, por lo que los usuarios de la base de datos pueden volver a crear la extensión si lo desean.

- Si desea dejar los archivos de extensión intactos para que los usuarios de la base de datos puedan crear su extensión TLE, puede detenerse aquí.
- Si desea eliminar todos los archivos que conforman la extensión, proceda con el siguiente paso.

3. Para eliminar todos los archivos de instalación de la extensión, utilice la función `pgtle.uninstall_extension`. Esta función elimina todos los archivos de código y control de la extensión.

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

Por ejemplo, para eliminar todos los archivos de instalación `pg_distance`, utilice el siguiente comando.

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

Desinstalación de Extensiones de lenguaje de confianza para PostgreSQL

Si ya no quiere crear sus propias extensiones TLE con TLE, puede eliminar la extensión `pg_tle` y eliminar todos los artefactos. Esta acción incluye eliminar cualquier extensión TLE de la base de datos y el esquema `pgtle`.

Para eliminar la extensión **pg_tle** y su esquema de una base de datos

1. Use `psql` u otra herramienta de cliente para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password --dbname=dbname
   ```

2. Elimine la extensión `pg_tle` de la base de datos. Si la base de datos está ejecutando sus propias extensiones TLE, también debe eliminar esas extensiones. Para ello, puede utilizar la palabra clave `CASCADE`, tal como se muestra a continuación.

   ```
   DROP EXTENSION pg_tle CASCADE;
   ```

   Si la extensión `pg_tle` no sigue activa en la base de datos, no es necesario que utilice la palabra clave `CASCADE`.

3. Elimine el esquema `pgtle`. Esta acción elimina todas las funciones de administración de la base de datos.

   ```
   DROP SCHEMA pgtle CASCADE;
   ```

   El comando devuelve lo siguiente cuando se completa el proceso.

   ```
   DROP SCHEMA
   ```

   Se eliminan la extensión `pg_tle`, su esquema y sus funciones, así como todos los artefactos. Para crear nuevas extensiones con TLE, vuelva a realizar el proceso de configuración. Para obtener más información, consulte Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL.

Uso de enlaces de PostgreSQL con sus extensiones TLE

Un enlace es un mecanismo de devolución de llamada disponible en PostgreSQL que permite a los desarrolladores llamar a funciones personalizadas u otras rutinas durante las operaciones normales de la base de datos. El kit de desarrollo de TLE admite enlaces de PostgreSQL para que pueda integrar funciones personalizadas con el comportamiento de PostgreSQL en el tiempo de ejecución. Por ejemplo, puede utilizar un enlace para asociar el proceso de autenticación a su propio código

personalizado o para modificar el proceso de planificación y ejecución de consultas según sus necesidades específicas.

Sus extensiones TLE pueden utilizar enlaces. Si un enlace tiene un alcance global, se aplica a todas las bases de datos. Por lo tanto, si su extensión TLE usa un enlace global, debe crear su extensión TLE en todas las bases de datos a las que puedan acceder sus usuarios.

Cuando usa la extensión `pg_tle` para crear sus propias Extensiones de lenguaje de confianza, puede usar los enlaces disponibles de una API de SQL para crear las funciones de su extensión. Debe registrar cualquier enlace con `pg_tle`. Para algunos enlaces, es posible que también tenga que establecer varios parámetros de configuración. Por ejemplo, el enlace de retención `passcode` se puede configurar como activado, desactivado u obligatorio. Para obtener más información sobre los requisitos específicos de los enlaces `pg_tle` disponibles, consulte Referencia de enlaces para Extensiones de lenguaje de confianza para PostgreSQL.

Ejemplo: Crear una extensión que utilice un enlace de PostgreSQL

El ejemplo descrito en esta sección utiliza un enlace de PostgreSQL para comprobar la contraseña proporcionada durante operaciones SQL específicas e impide que los usuarios de la base de datos establezcan sus contraseñas iguales a las que figuran en la tabla `password_check.bad_passwords`. La tabla contiene las diez opciones de contraseñas más utilizadas, pero fáciles de descifrar.

Para configurar este ejemplo en su clúster de base de datos de Aurora PostgreSQL, ya tiene que tener Extensiones de lenguaje de confianza instalado. Para obtener más información, consulte Configuración de Extensiones de lenguaje de confianza en su clúster de base de datos de Aurora PostgreSQL.

Para configurar el ejemplo del enlace de verificación de contraseñas

1. Use `psql` para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Copie el código de la Lista de códigos del enlace password_check y péguelo en su base de datos.

   ```
   SELECT pgtle.install_extension (
   ```

```
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
```

```
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Cuando la extensión se haya cargado en la base de datos, verá un resultado como el siguiente.

```
 install_extension
-------------------
 t
(1 row)
```

3. Mientras siga conectado a la base de datos, ya podrá crear la extensión.

```
CREATE EXTENSION my_password_check_rules;
```

4. Puede confirmar que la extensión se ha creado en la base de datos mediante el siguiente metacomando `psql`.

```
\dx
                        List of installed extensions
          Name           | Version |   Schema   |
 Description
-------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. Abra otra sesión de terminal para trabajar con la AWS CLI. Debe modificar su grupo de parámetros de base de datos personalizado para activar el enlace de verificación de contraseñas. Para ello, utilice el comando modify-db-parameter-group de la CLI tal como se muestra en el siguiente ejemplo.

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

   Puede que el cambio en la configuración del grupo de parámetros tarde unos minutos en aplicarse. Sin embargo, este parámetro es dinámico, por lo que no es necesario reiniciar la instancia de escritor del clúster de base de datos de Aurora para PostgreSQL para que la configuración surta efecto.

6. Abra la sesión `psql` y consulte la base de datos para comprobar que el enlace password_check esté activado.

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
----------------------------
on
(1 row)
```

El enlace password-check ahora está activo. Puede probarlo creando un rol nuevo y utilizando una de las contraseñas incorrectas, tal como se muestra en el siguiente ejemplo.

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

El resultado se ha modificado para que se pueda leer.

El siguiente ejemplo muestra que el comportamiento `\password` del metacomando interactivo `pgsql` también se ve afectado por el enlace password_check.

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

Puede eliminar esta extensión TLE y desinstalar sus archivos de código fuente si lo desea. Para obtener más información, consulte Eliminar las extensiones TLE de una base de datos.

Lista de códigos del enlace password_check

El código de ejemplo que se muestra aquí define la especificación de la extensión TLE `my_password_check_rules`. Al copiar este código y pegarlo en la base de datos, el código de la extensión `my_password_check_rules` se carga en la base de datos y el enlace `password_check` queda registrado para que lo utilice la extensión.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
```

```
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Referencia de funciones para Extensiones de lenguaje de confianza para PostgreSQL

Consulte la siguiente documentación de referencia sobre las funciones disponibles en Extensiones de lengua de confianza para PostgreSQL. Utilice estas funciones para instalar, registrar, actualizar y administrar sus extensiones TLE, es decir, las extensiones de PostgreSQL que desarrolla con el kit de desarrollo de Extensiones de lenguaje de confianza.

Funciones

- pgtle.available_extensions
- pgtle.available_extension_versions
- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(nombre, versión)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

La función `pgtle.available_extensions` es una función de devolución de conjuntos. Devuelve todas las extensiones TLE disponibles en la base de datos. Cada fila devuelta contiene información sobre una sola extensión TLE.

Prototipo de función

```
pgtle.available_extensions()
```

Rol

Ninguna.

Argumentos

Ninguna.

Salida

- `name`: nombre de la extensión TLE.
- `default_version`: versión de la extensión TLE que se utilizará cuando se llame a `CREATE EXTENSION` sin especificar una versión.
- `description`: descripción más detallada acerca de la extensión TLE.

Ejemplo de uso

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

La función `available_extension_versions` es una función de devolución de conjuntos. Esta función devuelve una lista de todas las extensiones de TLE disponibles y sus versiones. Cada fila contiene información sobre una versión específica de la extensión TLE dada, incluso si requiere un rol específico.

Prototipo de función

```
pgtle.available_extension_versions()
```

Rol

Ninguna.

Argumentos

Ninguna.

Salida

- `name`: nombre de la extensión TLE.

- `version`: versión de la extensión TLE.
- `superuser`: este valor es siempre `false` para sus extensiones TLE. Los permisos necesarios para crear la extensión TLE o actualizarla son los mismos que para crear otros objetos en la base de datos dada.
- `trusted`: este valor es siempre `false` para una extensión TLE.
- `relocatable`: este valor es siempre `false` para una extensión TLE.
- `schema`: especifica el nombre del esquema en el que está instalada la extensión TLE.
- `requires`: matriz que contiene los nombres de otras extensiones que necesita esta extensión TLE.
- `description`: descripción detallada de la extensión TLE.

Para obtener más información acerca de los valores de salida, vea Packaging Related Objects into an Extension > Extension Files (Empaquetar objetos relacionados en una extensión > Archivos de extensión) en la documentación de PostgreSQL.

Ejemplo de uso

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

La función `extension_update_paths` es una función de devolución de conjuntos. Devuelve una lista de todas las rutas de actualización posibles para una extensión TLE. Cada fila incluye las actualizaciones a un nivel superior o inferior disponibles para esa extensión TLE.

Prototipo de función

```
pgtle.extension_update_paths(name)
```

Rol

Ninguna.

Argumentos

`name`: nombre de la extensión TLE desde la que se obtienen las rutas de actualización.

Salida

- `source`: versión de origen de una actualización.
- `target`: versión de destino de una actualización.
- `path`: ruta de actualización utilizada para actualizar una extensión TLE de una versión `source` a otra `target`, por ejemplo, `0.1--0.2`.

Ejemplo de uso

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

La función `install_extension` le permite instalar los artefactos que componen la extensión TLE en la base de datos, después de lo cual se puede crear mediante el comando `CREATE EXTENSION`.

Prototipo de función

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

Rol

Ninguna.

Argumentos

- `name`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.
- `version`: versión de la extensión TLE.
- `description`: descripción detallada acerca de la extensión TLE. Esta descripción se muestra en el campo `comment` de `pgtle.available_extensions()`.
- `ext`: contenido de la extensión TLE. Este valor contiene objetos como funciones.
- `requires`: parámetro opcional que especifica las dependencias de esta extensión TLE. La extensión `pg_tle` se añade automáticamente como una dependencia.

Muchos de estos argumentos son los mismos que se incluyen en un archivo de control de extensiones para instalar una extensión de PostgreSQL en el sistema de archivos de una instancia de PostgreSQL. Para obtener más información acerca de las extensiones de PostgreSQL,

vea Extension Files (Archivos de extensión) en Packaging Related Objects into an Extension (Empaquetar objetos relacionados en una extensión) en la documentación de PostgreSQL.

Salida

Esta función devuelve `OK` en caso de éxito y `NULL` en caso de error.

- `OK`: la extensión TLE se ha instalado correctamente en la base de datos.
- `NULL`: la extensión TLE no se ha instalado correctamente en la base de datos.

Ejemplo de uso

```
SELECT pgtle.install_extension(
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

La función `install_update_path` proporciona una ruta de actualización entre dos versiones diferentes de una extensión TLE. Esta función permite a los usuarios de la extensión TLE actualizar su versión mediante la sintaxis `ALTER EXTENSION ... UPDATE`.

Prototipo de función

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

Rol

`pgtle_admin`

Argumentos

- `name`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.

- `fromvers`: versión de origen de la extensión TLE utilizada para la actualización.
- `tovers`: versión de destino de la extensión TLE utilizada para la actualización.
- `ext`: contenido de la actualización. Este valor contiene objetos como funciones.

Salida

Ninguna.

Ejemplo de uso

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

La función `register_feature` añade la característica interna de PostgreSQL especificada a la tabla `pgtle.feature_info`. Los enlaces de PostgreSQL son un ejemplo de una característica interna de PostgreSQL. El kit de desarrollo de Extensiones de lenguaje de confianza admite el uso de enlaces de PostgreSQL. Actualmente, esta función admite la siguiente característica.

- `passcheck`: registra el enlace de comprobación de contraseñas con su procedimiento o función que personaliza el comportamiento de comprobación de contraseñas de PostgreSQL.

Prototipo de función

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

Rol

`pgtle_admin`

Argumentos

- `proc`: nombre de un procedimiento o función almacenados que se utilizarán en la característica.
- `feature`: nombre de una característica `pg_tle` (como `passcheck`) para registrarla con la función.

Salida

Ninguna.

Ejemplo de uso

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

La función `pgtle.register_feature_if_not_exists` añade la función de PostgreSQL especificada a la tabla `pgtle.feature_info` e identifica la extensión TLE u otro procedimiento o función que utilice la característica. Para obtener más información sobre los enlaces y las extensiones de lenguaje de confianza, consulteUso de enlaces de PostgreSQL con sus extensiones TLE.

Prototipo de función

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

Rol

`pgtle_admin`

Argumentos

- `proc`: nombre de una función procedimiento almacenado que contiene la lógica (código) que se utilizará como una característica de la extensión TLE. Por ejemplo, el código `pw_hook`.
- `feature`: nombre de una la característica de PostgreSQL para registrarla para la función TLE. Actualmente, la única característica disponible es el enlace `passcheck`. Para obtener más información, consulte Enlace de comprobación de contraseñas (passcheck).

Salida

Devuelve `true` después de registrar la característica para la extensión especificada. Devuelve `false` si la característica ya está registrada.

Ejemplo de uso

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

La función `set_default_version` le permite especificar un `default_version` para su extensión TLE. Puede utilizar esta función para definir una ruta de actualización y designar la versión como la predeterminada para la extensión TLE. Cuando los usuarios de la base de datos especifican la extensión TLE en los comandos `CREATE EXTENSION` y `ALTER EXTENSION ... UPDATE`, esa versión de la extensión TLE se crea en la base de datos para ese usuario.

Esta función devuelve `true` en caso de realizarse correctamente. Si la extensión TLE especificada en el argumento `name` no existe, la función devuelve un error. Del mismo modo, si el `version` de la extensión TLE no existe, devuelve un error.

Prototipo de función

```
pgtle.set_default_version(name text, version text)
```

Rol

`pgtle_admin`

Argumentos

- `name`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.
- `version`: versión de la extensión TLE para establecer la predeterminada.

Salida

- `true`: cuando la configuración de la versión predeterminada se realiza correctamente, la función devuelve `true`.

- `ERROR`: devuelve un mensaje de error si no existe una extensión TLE con el nombre o la versión especificados.

Ejemplo de uso

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension(name)

La función `uninstall_extension` elimina todas las versiones de una extensión TLE de una base de datos. Esta función evita futuras llamadas de `CREATE EXTENSION` para evitar instalar la extensión TLE. Si la extensión TLE no existe en la base de datos, se genera un error.

La función `uninstall_extension` no elimina una extensión TLE que esté activa actualmente en la base de datos. Para eliminar una extensión TLE que está activa actualmente, debes llamar explícitamente a `DROP EXTENSION` para eliminarla.

Prototipo de función

```
pgtle.uninstall_extension(extname text)
```

Rol

`pgtle_admin`

Argumentos

- `extname`: nombre de la extensión TLE que se va a desinstalar. Este nombre es el mismo que se usó con `CREATE EXTENSION` para cargar la extensión TLE para usarla en una base de datos determinada.

Salida

Ninguna.

Ejemplo de uso

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension(nombre, versión)

La función `uninstall_extension(name, version)` elimina la versión especificada de la extensión TLE de la base de datos. Esta función impide a `CREATE EXTENSION` y `ALTER EXTENSION` instalar o actualizar una extensión TLE a la versión especificada. Esta función también elimina todas las rutas de actualización posibles de la extensión TLE especificada. Esta función no desinstala la extensión TLE si actualmente está activa en la base de datos. Debe llamar explícitamente a `DROP EXTENSION` para eliminar la extensión TLE. Para desinstalar todas las versiones de una extensión TLE, consulte pgtle.uninstall_extension(name).

Prototipo de función

```
pgtle.uninstall_extension(extname text, version text)
```

Rol

`pgtle_admin`

Argumentos

- `extname`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.
- `version`: versión de la extensión TLE que se va a desinstalar de la base de datos.

Salida

Ninguna.

Ejemplo de uso

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

La función `uninstall_extension_if_exists` elimina todas las versiones de una extensión TLE de una base de datos determinada. Si la extensión TLE no existe, la función la devuelve en silencio (no se genera ningún mensaje de error). Si la extensión especificada está activa actualmente en una base de datos, esta función no la elimina. Debe llamar explícitamente a `DROP EXTENSION` para eliminar la extensión TLE antes de utilizar esta función para desinstalar sus artefactos.

Prototipo de función

```
pgtle.uninstall_extension_if_exists(extname text)
```

Rol

`pgtle_admin`

Argumentos

- `extname`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.

Salida

La función `uninstall_extension_if_exists` de vuelve `true` después de desinstalar la extensión especificada. Si la extensión especificada no existe, la función devuelve `false`.

- `true`: devuelve `true` después de desinstalar la extensión TLE.
- `false`: devuelve `false` cuando la extensión TLE no existe en la base de datos.

Ejemplo de uso

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

La función `uninstall_update_path` elimina la ruta de actualización específica de una extensión TLE. Esto impide que `ALTER EXTENSION ... UPDATE TO` se utilice como ruta de actualización.

Si una de las versiones de esta ruta de actualización utiliza actualmente la extensión TLE, permanecerá en la base de datos.

Si la ruta de actualización especificada no existe, esta función genera un error.

Prototipo de función

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

Rol

`pgtle_admin`

Argumentos

- `extname`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.
- `fromvers`: versión de origen de la extensión TLE utilizada en la ruta de actualización.
- `tovers`: versión de destino de la extensión TLE utilizada en la ruta de actualización.

Salida

Ninguna.

Ejemplo de uso

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

La función `uninstall_update_path_if_exists` es similar a `uninstall_update_path` en el sentido de que elimina la ruta de actualización especificada de una extensión TLE. Sin embargo, si la ruta de actualización no existe, esta función no generará ningún mensaje de error. En su lugar, la función devuelve `false`.

Prototipo de función

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

Rol

`pgtle_admin`

Argumentos

- `extname`: nombre de la extensión TLE. Este valor se utiliza cuando se llama a `CREATE EXTENSION`.

- `fromvers`: versión de origen de la extensión TLE utilizada en la ruta de actualización.
- `tovers`: versión de destino de la extensión TLE utilizada en la ruta de actualización.

Salida

- `true`: la función ha actualizado correctamente la ruta de la extensión TLE.
- `false`: la función no ha podido actualizar la ruta de la extensión TLE.

Ejemplo de uso

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

La función `unregister_feature` proporciona una forma de eliminar las funciones que se han registrado para usar características `pg_tle`, como los enlaces. Para obtener información sobre el registro de una característica, consulte pgtle.register_feature.

Prototipo de función

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

Rol

`pgtle_admin`

Argumentos

- `proc`: nombre de una función almacenada para registrarse en una característica de `pg_tle`.
- `feature`: nombre de la característica `pg_tle` para registrarla con la función. Por ejemplo, `passcheck` es una característica que se puede registrar para que la utilicen las extensiones de lenguaje de confianza que desarrolle. Para obtener más información, consulte Enlace de comprobación de contraseñas (passcheck).

Salida

Ninguna.

Ejemplo de uso

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

La función `unregister_feature` proporciona una forma de eliminar las funciones que se registraron para usar funciones `pg_tle`, como los enlaces. Para obtener más información, consulte Uso de enlaces de PostgreSQL con sus extensiones TLE. Devuelve `true` después de anular satisfactoriamente el registro de la función. Devuelve `false` si la función no se ha registrado.

Para obtener información sobre el registro de funciones `pg_tle` para sus extensiones TLE, consulte pgtle.register_feature.

Prototipo de función

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

Rol

`pgtle_admin`

Argumentos

- `proc`: nombre de la función almacenada que se registró para incluir una función `pg_tle`.
- `feature`: nombre de la función `pg_tle` que se registró con la extensión de lenguaje de confianza.

Salida

Devuelve `true` o `false`, de la siguiente manera.

- `true`: la función ha cancelado satisfactoriamente el registro de la función de la extensión.
- `false`: la función no ha podido anular el registro de la función de la extensión TLE.

Ejemplo de uso

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

Referencia de enlaces para Extensiones de lenguaje de confianza para PostgreSQL

Extensiones de lenguaje de confianza para PostgreSQL admite los enlaces de PostgreSQL. Un enlace es un mecanismo interno de devolución de llamada disponible para los desarrolladores para ampliar la funcionalidad principal de PostgreSQL. Mediante el uso de enlaces, los desarrolladores pueden implementar sus propias funciones o procedimientos para utilizarlos durante diversas operaciones de bases de datos, modificando así el comportamiento de PostgreSQL de alguna manera. Por ejemplo, puede utilizar un enlace `passcheck` para personalizar la forma en que PostgreSQL gestiona las contraseñas proporcionadas al crear o cambiar las contraseñas de los usuarios (roles).

Consulte la siguiente documentación para obtener información sobre el enlace de passcheck disponible para sus extensiones TLE. Para obtener más información sobre los enlaces disponibles, incluido el enlace de autenticación del cliente, consulte los enlaces de Extensiones de lenguaje de confianza.

Enlace de comprobación de contraseñas (passcheck)

El enlace `passcheck` se utiliza para personalizar el comportamiento de PostgreSQL durante el proceso de comprobación de contraseñas para los siguientes comandos SQL y el metacomando `psql`.

- `CREATE ROLE` *`username`* `...PASSWORD`: para obtener más información, consulte CREATE ROLE en la documentación de PostgreSQL.
- `ALTER ROLE` *`username`*`...PASSWORD`: para obtener más información, consulte ALTER ROLE en la documentación de PostgreSQL.
- `\password` *`username`*: este metacomando `psql` interactivo cambia de forma segura la contraseña del usuario especificado mediante un hash de la contraseña antes de utilizar la sintaxis `ALTER ROLE ... PASSWORD` de forma transparente. El metacomando es un contenedor seguro para el comando `ALTER ROLE ... PASSWORD`, por lo que el enlace se aplica al comportamiento del metacomando `psql`.

Para ver un ejemplo, consulta Lista de códigos del enlace password_check.

Contenido

- Prototipo de función

- Argumentos
- Configuración
- Notas de uso

Prototipo de función

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

Argumentos

La función de enlace `passcheck` acepta los argumentos siguientes:

- `username`: el nombre (como texto) del rol (nombre de usuario) que establece una contraseña.
- `password`: la contraseña en texto sin formato o con hash. La contraseña introducida debe coincidir con el tipo especificado en `password_type`.
- `password_type`: especifique el formato `pgtle.password_type` de la contraseña. Este formato puede ser uno de los siguientes:
  - `PASSWORD_TYPE_PLAINTEXT`: una contraseña sin formato.
  - `PASSWORD_TYPE_MD5`: una contraseña que se ha cifrado con hash mediante el algoritmo MD5 (resumen de mensaje 5).
  - `PASSWORD_TYPE_SCRAM_SHA_256`: una contraseña que se ha cifrado con hash mediante el algoritmo SCRAM-SHA-256.
- `valid_until`: especifica el momento en que la contraseña deja de ser válida. Este argumento es opcional. Si utiliza este argumento, especifique la hora como valor `timestamptz`.
- `valid_null`: si este valor booleano está establecido en `true`, la opción `valid_until` se establece en `NULL`.

Configuración

La función `pgtle.enable_password_check` controla si el enlace de passcheck está activo. El enlace de passcheck tiene tres ajustes posibles.

- `off`: desactiva el enlace de comprobación de contraseñas `passcheck`. Este es el valor predeterminado.
- on: activa el enlace de comprobación de contraseñas `passcode` para cotejarlas con la tabla.

- `require`: requiere que se defina un enlace de comprobación de contraseñas.

Notas de uso

Para activar o desactivar el enlace `passcheck`, debe modificar el grupo de parámetros de base de datos personalizado de la instancia del escritor de su clúster de base de datos de Aurora PostgreSQL.

Para Linux, macOS o Unix:

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-parameter-group ^
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Referencia de Amazon Aurora PostgreSQL

En los siguientes temas, encontrará información sobre las intercalaciones, las funciones, los parámetros y los eventos de espera en Amazon Aurora PostgreSQL.

Temas

- Intercalaciones de Aurora PostgreSQL para EBCDIC y otras migraciones de mainframe
- Intercalaciones admitidas en Aurora PostgreSQL
- Referencia de las funciones de Aurora PostgreSQL
- Parámetros de Amazon Aurora PostgreSQL.
- Eventos de espera de Amazon Aurora PostgreSQL

Intercalaciones de Aurora PostgreSQL para EBCDIC y otras migraciones de mainframe

Migración de aplicaciones de mainframe a nuevas plataformas, como AWS conserva idealmente el comportamiento de las aplicaciones. Para preservar el comportamiento de las aplicaciones en una nueva plataforma exactamente igual que en el mainframe, se requiere que los datos migrados se recopilen utilizando las mismas reglas de intercalación y clasificación. Por ejemplo, muchas soluciones de migración de Db2 cambian los valores nulos a u0180 (posición Unicode 0180), por lo que estas intercalaciones clasifican u0180 primero. Este es un ejemplo de cómo las intercalaciones pueden variar de su fuente de mainframe y por qué es necesario elegir una intercalación que se adapte mejor a la intercalación EBCDIC original.

Aurora PostgreSQL 14.3 y versiones posteriores proporcionan muchas intercalaciones de ICU y EBCDIC para admitir dicha migración a AWS usando el servicio AWS Mainframe Modernization. Para obtener más información acerca de este servicio, consulte el tema sobre qué es AWS Mainframe Modernization.

En la siguiente tabla, encontrará intercalaciones provistas por Aurora PostgreSQL. Estas intercalaciones siguen las reglas de EBCDIC y garantizan que las aplicaciones de mainframe funcionen igual en AWS como en el entorno de mainframe. El nombre de la intercalación incluye la página de códigos correspondiente, (cpnnnn), para que pueda elegir la intercalación adecuada para su origen de mainframe. Por ejemplo, use `en-US-cp037-x-icu` para lograr el comportamiento de intercalación para los datos EBCDIC que se originaron en una aplicación de mainframe que utilizaba la página de códigos 037.

Intercalaciones EBCDIC	Intercalaciones AWS Blu Age	Intercalaciones AWS Micro Focus
da-DK-cp1142-x-icu	da-DK-cp1142m-x-icu	da-DK-cp1142m-x-icu
da-DK-cp277-x-icu	da-DK-cp277b-x-icu	–
de-DE-cp1141-x-icu	de-DE-cp1141b-x-icu	de-DE-cp1141m-x-icu
de-DE-cp273-x-icu	de-DE-cp273b-x-icu	–
en-GB-cp1146-x-icu	en-GB-cp1146b-x-icu	en-GB-cp1146m-x-icu
en-GB-cp285-x-icu	en-GB-cp285b-x-icu	–
en-US-cp037-x-icu	en-US-cp037b-x-icu	–
en-US-cp1140-x-icu	en-US-cp1140b-x-icu	en-US-cp1140m-x-icu
es-ES-cp1145-x-icu	es-ES-cp1145b-x-icu	es-ES-cp1145m-x-icu
es-ES-cp284-x-icu	es-ES-cp284b-x-icu	–
fi-FI-cp1143-x-icu	fi-FI-cp1143b-x-icu	fi-FI-cp1143m-x-icu
fi-FI-cp278-x-icu	fi-FI-cp278b-x-icu	–
fr-FR-cp1147-x-icu	fr-FR-cp1147b-x-icu	fr-FR-cp1147m-x-icu
fr-FR-cp297-x-icu	fr-FR-cp297b-x-icu	–
it-IT-cp1144-x-icu	it-IT-cp1144b-x-icu	it-IT-cp1144m-x-icu
it-IT-cp280-x-icu	it-IT-cp280b-x-icu	–
nl-BE-cp1148-x-icu	nl-BE-cp1148b-x-icu	nl-BE-cp1148m-x-icu
nl-BE-cp500-x-icu	nl-BE-cp500b-x-icu	–

Para obtener más información sobre AWS Blu Age, consulte el tutorial sobre el tiempo de ejecución administrado para AWS Blu Age en la guía del usuario de AWS Mainframe Modernization.

Para obtener información acerca del uso de AWS Micro Focus, consulte el tutorial sobre el tiempo de ejecución administrado para Micro Focus en la guía del usuario de AWS Mainframe Modernization.

Para obtener más información sobre la administración de intercalaciones en PostgreSQL, consulte compatibilidad de las intercalaciones en la documentación de PostgreSQL.

Intercalaciones admitidas en Aurora PostgreSQL

Las intercalaciones son un conjunto de reglas que determinan cómo se ordenan y comparan las cadenas de caracteres almacenadas en la base de datos. Las intercalaciones desempeñan un papel fundamental en el sistema de computación y se incluyen como parte del sistema operativo. Las intercalaciones cambian con el tiempo cuando se añaden nuevos caracteres a los lenguajes o cuando cambian las reglas de ordenación.

Las bibliotecas de intercalaciones definen reglas y algoritmos específicos para una intercalación. Las bibliotecas de intercalaciones más populares utilizadas en PostgreSQL son GNU C (glibc) y los componentes de internacionalización de Unicode (ICU). De forma predeterminada, Aurora PostgreSQL utiliza la intercalación glibc, que incluye ordenaciones de caracteres Unicode para secuencias de caracteres de varios bytes.

Al crear un nuevo clúster de base de datos de Aurora PostgreSQL, se comprueba en el sistema operativo la intercalación disponible. Los parámetros de PostgreSQL del comando `CREATE DATABASE` `LC_COLLATE` y `LC_CTYPE` se utilizan para especificar una intercalación, que es la intercalación predeterminada en esa base de datos. Como alternativa, también puede utilizar el parámetro `LOCALE` en `CREATE DATABASE` para establecer estos parámetros. Esto determina la intercalación predeterminada de las cadenas de caracteres de la base de datos y las reglas para clasificar los caracteres como letras, números o símbolos. También puede elegir una intercalación para utilizarla en una columna, un índice o una consulta.

Aurora PostgreSQL depende de la biblioteca glibc del sistema operativo para admitir las intercalaciones. La instancia de Aurora PostgreSQL se actualiza periódicamente con las versiones más recientes del sistema operativo. Estas actualizaciones a veces incluyen una versión más reciente de la biblioteca glibc. En raras ocasiones, las versiones más recientes de glibc cambian la ordenación o la intercalación de algunos caracteres, lo que puede provocar que los datos se ordenen de forma diferente o generar entradas de índice no válidas. Si durante una actualización detecta problemas de ordenación para la intercalación, es posible que tenga que volver a generar los índices.

Para reducir el posible impacto de las actualizaciones de glibc, Aurora PostgreSQL incluye ahora una biblioteca de intercalaciones predeterminada independiente. Esta biblioteca de intercalaciones está disponible en Aurora PostgreSQL 14.6, 13.9, 12.13, 11.18 y versiones secundarias posteriores. Es compatible con glibc 2.26-59.amzn2 y proporciona estabilidad en la ordenación para evitar resultados de consultas incorrectos.

Referencia de las funciones de Aurora PostgreSQL

A continuación, encontrará una lista de las funciones de Aurora PostgreSQL disponibles para los clústeres de base de datos de Aurora que ejecutan el motor de base de datos de edición compatible con Aurora PostgreSQL. Estas funciones de Aurora PostgreSQL se suman a las funciones estándar de PostgreSQL. Para obtener más información acerca de las funciones estándar de PostgreSQL, consulte PostgreSQL: funciones y operadores.

Descripción general

Puede utilizar las siguientes funciones para instancias de base de datos de Amazon RDS que ejecutan Aurora PostgreSQL:

- aurora_db_instance_identifier
- aurora_ccm_status
- aurora_global_db_instance_status
- aurora_global_db_status
- aurora_list_builtins
- aurora_replica_status
- aurora_stat_activity
- aurora_stat_backend_waits
- aurora_stat_bgwriter
- aurora_stat_database
- aurora_stat_dml_activity
- aurora_stat_get_db_commit_latency
- aurora_stat_logical_wal_cache
- aurora_stat_memctx_usage
- aurora_stat_optimized_reads_cache
- aurora_stat_plans

- aurora_stat_reset_wal_cache
- aurora_stat_statements
- aurora_stat_system_waits
- aurora_stat_wait_event
- aurora_stat_wait_type
- aurora_version
- aurora_volume_logical_start_lsn
- aurora_wait_report

aurora_db_instance_identifier

Informa del nombre de la instancia de base de datos a la que está conectado.

Sintaxis

```
aurora_db_instance_identifier()
```

Argumentos

Ninguno

Tipo de retorno

Cadena VARCHAR

Notas de uso

Esta función muestra el nombre de la instancia de base de datos del clúster de Aurora PostgreSQL-Compatible Edition para su conexión de aplicaciones o cliente de base de datos.

Esta función está disponible desde el lanzamiento de las versiones 13.7, 12.11, 11.16 y 10.21 de Aurora PostgreSQL, y para todas las demás versiones posteriores.

Ejemplos

En el ejemplo siguiente se muestra lo que ocurre al llamar a la función `aurora_db_instance_identifier`.

```
=> SELECT aurora_db_instance_identifier();
aurora_db_instance_identifier
------------------------------
 test-my-instance-name
```

Puede unir los resultados de esta función con la función `aurora_replica_status` para obtener detalles sobre la instancia de base de datos de la conexión. aurora_replica_status sola no muestra qué instancia de base de datos está utilizando. El siguiente ejemplo muestra cómo.

```
=> SELECT *
    FROM aurora_replica_status() rt,
         aurora_db_instance_identifier() di
    WHERE rt.server_id = di;
-[ RECORD 1 ]----------------------+----------------------
server_id                         | test-my-instance-name
session_id                        | MASTER_SESSION_ID
durable_lsn                       | 88492069
highest_lsn_rcvd                  |
current_read_lsn                  |
cur_replay_latency_in_usec        |
active_txns                       |
is_current                        | t
last_transport_error              | 0
last_error_timestamp              |
last_update_timestamp             | 2022-06-03 11:18:25+00
feedback_xmin                     |
feedback_epoch                    |
replica_lag_in_msec               |
log_stream_speed_in_kib_per_second | 0
log_buffer_sequence_number        | 0
oldest_read_view_trx_id           |
oldest_read_view_lsn              |
pending_read_ios                  | 819
```

aurora_ccm_status

Muestra el estado del administrador de caché del clúster.

Sintaxis

```
aurora_ccm_status()
```

Argumentos

Ninguna.

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `buffers_sent_last_minute`: el número de búferes enviados al lector designado en el último minuto.
- `buffers_found_last_minute`: el número de búferes de acceso frecuente identificados durante el último minuto.
- `buffers_sent_last_scan`: la cantidad de búferes enviados al lector designado durante el último análisis completo de la caché del búfer.
- `buffers_found_last_scan`: el número de búferes de acceso frecuente enviados durante la última exploración completa de la caché del búfer. Los búferes que ya están almacenados en la caché en el lector designado no se envían.
- `buffers_sent_current_scan`: el número de búferes enviados durante el análisis actual.
- `buffers_found_current_scan`: el número de búferes de acceso frecuente identificados en el análisis actual.
- `current_scan_progress`: el número de búferes visitados hasta el momento durante el análisis actual.

Notas de uso

Puede utilizar esta función para comprobar y supervisar la función de administración de caché del clúster (CCM). Esta función solo funciona si CCM está activo en el clúster de base de datos Aurora PostgreSQL. Para utilizar esta función, conéctese a la instancia de base de datos de escritura del clúster de base de datos Aurora PostgreSQL.

Puede activar CCM para un clúster de base de datos Aurora PostgreSQL configurando `apg_ccm_enabled` en 1 del grupo de parámetros del clúster de base de datos personalizado del clúster. Para aprender a hacerlo, consulte Configuración de la administración de la caché del clúster.

La administración de caché de clúster está activa en un clúster de base de datos Aurora PostgreSQL cuando el clúster tiene una instancia de Aurora Reader configurada de la siguiente forma:

- La instancia de Aurora Reader utiliza el mismo tipo y tamaño de clase de instancia de base de datos que la instancia Writer del clúster.
- La instancia de Aurora Reader se configura como nivel 0 para el clúster. Si el clúster tiene más de un lector, este es su único lector de nivel 0.

Al configurar más de un lector en el nivel 0, se deshabilita CCM. Cuando CCM está deshabilitado, al llamar a esta función se devuelve el siguiente mensaje de error:

```
ERROR: Cluster Cache Manager is disabled
```

También puede utilizar la extensión pg_buffercache de PostgreSQL para analizar la caché del búfer. Para obtener más información consulte pg_buffercache, en la documentación de PostgreSQL.

Para obtener más información, consulte Introducción a la administración de caché de clúster de Aurora PostgreSQL.

Ejemplos

En el ejemplo siguiente se muestran los resultados de llamar a la función `aurora_ccm_status`. En este primer ejemplo se muestran las estadísticas de CCM.

```
=> SELECT * FROM aurora_ccm_status();
 buffers_sent_last_minute | buffers_found_last_minute | buffers_sent_last_scan |
 buffers_found_last_scan | buffers_sent_current_scan | buffers_found_current_scan |
 current_scan_progress
--------------------------+---------------------------+------------------------
+-------------------------+---------------------------+---------------------------
+------------------------
                  2242000 |                   2242003 |               17920442 |
         17923410 |                  14098000 |                   14100964 |
   15877443
```

Para obtener más detalles, puede utilizar la pantalla ampliada, como se muestra a continuación:

```
\x
Expanded display is on.
SELECT *  FROM aurora_ccm_status();
[ RECORD 1 ]----------------------+---------
buffers_sent_last_minute          | 2242000
buffers_found_last_minute         | 2242003
```

```
buffers_sent_last_scan             | 17920442
buffers_found_last_scan            | 17923410
buffers_sent_current_scan          | 14098000
buffers_found_current_scan         | 14100964
current_scan_progress              | 15877443
```

En este ejemplo se muestra cómo comprobar la tasa de calentamiento y el porcentaje de calentamiento.

```
=> SELECT buffers_sent_last_minute * 8/60 AS warm_rate_kbps,
100 * (1.0-buffers_sent_last_scan/buffers_found_last_scan) AS warm_percent
FROM aurora_ccm_status ();
 warm_rate_kbps | warm_percent
----------------+--------------
 16523 |        100.0
```

aurora_global_db_instance_status

Muestra el estado de todas las instancias de Aurora, incluidas las réplicas de un clúster de base de datos global de Aurora.

Sintaxis

```
aurora_global_db_instance_status()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `server_id`: el identificador de la instancia de base de datos.
- `session_id`: un identificador único de la sesión actual. Un valor `MASTER_SESSION_ID` identifica la instancia de base de datos de Writer (principal).
- `aws_region`: la Región de AWS en la que se ejecuta esta instancia de base de datos global. Para obtener una lista de regiones, consulte Disponibilidad por región.
- `durable_lsn`: el número de secuencia de registro (LSN) hecho duradero en el almacenamiento. Un número de secuencia de registro (LSN) es un número secuencial único que identifica un

registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción posterior.

- `highest_lsn_rcvd`: LSN más alto recibido por la instancia de base de datos de la instancia de base de datos de escritor.
- `feedback_epoch`: fecha de inicio que utiliza la instancia de base de datos cuando genera información en espera activa. Una espera activa es una instancia de base de datos que admite conexiones y consultas mientras la base de datos principal está en modo de recuperación o espera. La información de espera activa incluye la época (punto en el tiempo) y otros detalles sobre la instancia de base de datos que se utiliza como reserva activa. Para obtener más información, consulte la documentación de PostgreSQL sobre Espera activa.
- `feedback_xmin`: ID de transacción activo mínimo (más antiguo) utilizado por la instancia de base de datos.
- `oldest_read_view_lsn`: LSN más antiguo utilizado por la instancia de base de datos para leer desde el almacenamiento.
- `visibility_lag_in_msec`: tiempo que esta instancia de base de datos se está quedando por detrás de la instancia de base de datos de escritor en milisegundos.

Notas de uso

Esta función muestra estadísticas de replicación para un clúster de base de datos Aurora. Para cada instancia de base de datos Aurora PostgreSQL del clúster, la función muestra una fila de datos que incluye cualquier réplica entre regiones en una configuración de base de datos global.

Puede ejecutar esta función desde cualquier instancia de un clúster de base de datos Aurora PostgreSQL o una base de datos global de Aurora PostgreSQL. La función devuelve detalles sobre el retraso de todas las instancias de réplica.

Para obtener más información sobre el retraso de la supervisión mediante esta función (`aurora_global_db_instance_status`) o utilizando `aurora_global_db_status`, consulteSupervisión de bases de datos globales basadas en Aurora PostgreSQL.

Para obtener información sobre las bases de datos globales de Aurora, consulte Información general sobre la base de datos global de Amazon Aurora.

Para empezar a utilizar bases de datos globales de Aurora, consulte Introducción a la base de datos global de Amazon Aurora o consulte Preguntas frecuentes de Amazon Aurora.

Ejemplos

En este ejemplo se muestran estadísticas de instancias entre regiones.

```
=> SELECT *
  FROM aurora_global_db_instance_status();
               server_id                |              session_id              |
  aws_region  | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------+--------------------------------------
+-------------+-------------+------------------+----------------+---------------
+----------------------+------------------------
 db-119-001-instance-01                  | MASTER_SESSION_ID                    | eu-
west-1    |  2534560273 |           [NULL] |         [NULL] |        [NULL] |
     [NULL] |                 [NULL]
 db-119-001-instance-02                  | 4ecff34d-d57c-409c-ba28-278b31d6fc40 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                    6
 db-119-001-instance-03                  | 3e8a20fc-be86-43d5-95e5-bdf19d27ad6b | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                    6
 db-119-001-instance-04                  | fc1b0023-e8b4-4361-bede-2a7e926cead6 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                   23
 db-119-001-instance-05                  | 30319b74-3f08-4e13-9728-e02aa1aa8649 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                   23
 db-119-001-global-instance-1            | a331ffbb-d982-49ba-8973-527c96329c60 | eu-
central-1 |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                  996
 db-119-001-global-instance-1            | e0955367-7082-43c4-b4db-70674064a9da | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                   14
 db-119-001-global-instance-1-eu-west-2a  | 1248dc12-d3a4-46f5-a9e2-85850491a897 | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                    0
```

En este ejemplo se muestra cómo comprobar el retraso de réplica global en milisegundos.

```
=> SELECT CASE
          WHEN 'MASTER_SESSION_ID' = session_id THEN 'Primary'
         ELSE 'Secondary'
      END AS global_role,
```

```
      aws_region,
      server_id,
      visibility_lag_in_msec
  FROM aurora_global_db_instance_status()
  ORDER BY 1, 2, 3;
   global_role |  aws_region  |               server_id                |
 visibility_lag_in_msec
-------------+--------------+----------------------------------------
+------------------------
 Primary     | eu-west-1    | db-119-001-instance-01                 |
 [NULL]
 Secondary   | eu-central-1 | db-119-001-global-instance-1           |
     13
 Secondary   | eu-west-1    | db-119-001-instance-02                 |
     10
 Secondary   | eu-west-1    | db-119-001-instance-03                 |
      9
 Secondary   | eu-west-1    | db-119-001-instance-04                 |
      2
 Secondary   | eu-west-1    | db-119-001-instance-05                 |
     18
 Secondary   | eu-west-2    | db-119-001-global-instance-1           |
     14
 Secondary   | eu-west-2    | db-119-001-global-instance-1-eu-west-2a |
     13
```

En este ejemplo se muestra cómo comprobar el retraso mínimo, máximo y medio por Región de AWS desde la configuración de base de datos global.

```
=> SELECT 'Secondary' global_role,
       aws_region,
       min(visibility_lag_in_msec) min_lag_in_msec,
       max(visibility_lag_in_msec) max_lag_in_msec,
       round(avg(visibility_lag_in_msec),0) avg_lag_in_msec
  FROM aurora_global_db_instance_status()
 WHERE aws_region NOT IN (SELECT   aws_region
                            FROM aurora_global_db_instance_status()
                           WHERE session_id='MASTER_SESSION_ID')
                        GROUP BY aws_region
UNION ALL
SELECT  'Primary' global_role,
        aws_region,
        NULL,
```

```
        NULL,
        NULL
    FROM aurora_global_db_instance_status()
   WHERE session_id='MASTER_SESSION_ID'
ORDER BY 1, 5;
 global_role |  aws_region  | min_lag_in_msec | max_lag_in_msec | avg_lag_in_msec
-------------+--------------+-----------------+-----------------+-----------------
 Primary     | eu-west-1    |          [NULL] |          [NULL] |          [NULL]
 Secondary   | eu-central-1 |             133 |             133 |             133
 Secondary   | eu-west-2    |               0 |             495 |             248
```

aurora_global_db_status

Muestra información sobre varios aspectos del retardo de la base de datos global de Aurora, específicamente, el retardo del almacenamiento de Aurora subyacente (llamado retardo de durabilidad) y el retardo entre el objetivo de punto de recuperación (RPO).

Sintaxis

```
aurora_global_db_status()
```

Argumentos

Ninguna.

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `aws_region`: la Región de AWS donde está este clúster de base de datos. Para ver una lista completa de Regiones de AWS por motor, consulte Regiones y zonas de disponibilidad.
- `highest_lsn_written`: el número de secuencia de registro (LSN) más alto que existe actualmente en este clúster de base de datos. Un número de secuencia de registro (LSN) es un número secuencial único que identifica un registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción posterior.
- `durability_lag_in_msec`: la diferencia en los valores de marca temporal entre el `highest_lsn_written` de un clúster de base de datos secundario y el

`highest_lsn_written` del clúster de base de datos principal. Un valor de -1 identifica el clúster de base de datos principal de una base de datos global de Aurora.

- `rpo_lag_in_msec`: el retraso del objetivo de punto de recuperación (RPO). El retardo de RPO es el tiempo que tarda la transacción de usuario más reciente en almacenarse en un clúster de base de datos secundario después de almacenarse en el clúster de base de datos principal de una base de datos global de Aurora. Un valor de -1 indica el clúster de base de datos principal (y, por lo tanto, el retardo no es relevante).

  En términos sencillos, esta métrica calcula el objetivo del punto de recuperación de cada clúster de base de datos de Aurora PostgreSQL de una base de datos global de Aurora, es decir, cuántos datos podrían perderse si se produce una interrupción. Al igual que con el retraso, el RPO se mide en tiempo.
- `last_lag_calculation_time`: la marca temporal que especifica cuándo se calcularon por última vez los valores para `durability_lag_in_msec` y `rpo_lag_in_msec`. Un valor temporal como `1970-01-01 00:00:00+00` significa que este es el clúster de base de datos principal.
- `feedback_epoch`: la fecha de inicio que el clúster de base de datos secundario usa cuando genera información en espera activa. Una espera activa es una instancia de base de datos que admite conexiones y consultas mientras la base de datos principal está en modo de recuperación o espera. La información de espera activa incluye la época (punto en el tiempo) y otros detalles sobre la instancia de base de datos que se utiliza como reserva activa. Para obtener más información, consulte la documentación de PostgreSQL sobre Espera activa.
- `feedback_xmin`: el ID de transacción activa mínima (más antigua) utilizada por el clúster de base de datos secundario.

Notas de uso

Esta función es compatible con todas las versiones de Aurora PostgreSQL disponibles actualmente. Esta función muestra estadísticas de replicación para una base de datos global de Aurora. Muestra una fila para cada clúster de base de datos de una base de datos global Aurora PostgreSQL. Puede ejecutar esta función desde cualquier instancia de una base de datos global de Aurora PostgreSQL.

Para evaluar el retraso de replicación de la base de datos global de Aurora, que es el retraso visible de los datos, consulte aurora_global_db_instance_status.

Para obtener más información sobre el uso de `aurora_global_db_status` y `aurora_global_db_instance_status` para supervisar el retraso global de la base de datos Aurora, consulte Supervisión de bases de datos globales basadas en Aurora PostgreSQL. Para

obtener información sobre las bases de datos globales de Aurora, consulte Información general sobre la base de datos global de Amazon Aurora.

Ejemplos

En este ejemplo se muestra cómo visualizar estadísticas de almacenamiento entre regiones.

```
=> SELECT CASE
          WHEN '-1' = durability_lag_in_msec THEN 'Primary'
          ELSE 'Secondary'
       END AS global_role,
       *
  FROM aurora_global_db_status();
 global_role | aws_region | highest_lsn_written | durability_lag_in_msec |
 rpo_lag_in_msec | last_lag_calculation_time  | feedback_epoch | feedback_xmin
-------------+------------+---------------------+------------------------
+-----------------+----------------------------+----------------+---------------
 Primary     | eu-west-1  |           131031557 |                     -1 |
 -1 | 1970-01-01 00:00:00+00     |             0 |              0
 Secondary   | eu-west-2  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
 Secondary   | eu-west-3  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
```

aurora_list_builtins

Enumera todas las funciones incorporadas de Aurora PostgreSQL disponibles, junto con descripciones breves y detalles de funciones.

Sintaxis

```
aurora_list_builtins()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF

Ejemplos

En el ejemplo siguiente se muestran los resultados de llamar a la función `aurora_list_builtins`.

```
=> SELECT *
FROM aurora_list_builtins();

               Name                | Result data type |                  Argument data
 types                     | Type | Volatility |  Parallel  | Security |
             Description
-----------------------------------+------------------
+-----------------------------------------------------------+------+------------
+------------+----------
+-------------------------------------------------------------------------
 aurora_version                    | text             |
                           | func | stable     | safe       | invoker  | Amazon Aurora
 PostgreSQL-Compatible Edition version string
 aurora_stat_wait_type             | SETOF record     | OUT type_id smallint, OUT
 type_name text                 | func | volatile   | restricted | invoker  | Lists all
 supported wait types
 aurora_stat_wait_event            | SETOF record     | OUT type_id smallint, OUT
 event_id integer, OUT event_na.| func | volatile   | restricted | invoker  | Lists all
 supported wait events
                                   |                  |.me text
                           |      |            |            |          |
 aurora_list_builtins              | SETOF record     | OUT "Name" text, OUT "Result
 data type" text, OUT "Argum.| func | stable     | safe       | invoker  | Lists all
 Aurora built-in functions
                                   |                  |.ent data types" text, OUT
 "Type" text, OUT "Volatility" .|      |            |            |          |
                                   |                  |.text, OUT "Parallel" text, OUT
 "Security" text, OUT "Des.|      |            |            |          |
                                   |                  |.cription" text
                           |      |            |            |          |
 .
 .
 .
 aurora_stat_file                  | SETOF record     | OUT filename text, OUT
 allocated_bytes bigint, OUT used_.| func | stable     | safe       | invoker  | Lists
 all files present in Aurora storage
                                   |                  |.bytes bigint
                           |      |            |            |          |
```

```
aurora_stat_get_db_commit_latency | bigint           | oid
                          | func | stable     | restricted | invoker  | Per DB commit
latency in microsecs
```

aurora_replica_status

Muestra el estado de todos los nodos de lector de Aurora PostgreSQL.

Sintaxis

```
aurora_replica_status()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `server_id`: identificador de la instancia de base de datos.
- `session_id`: un identificador único para la sesión actual, devuelto para la instancia principal y las instancias de lectura de la siguiente manera:
  - Para la instancia principal, `session_id` es siempre `` `MASTER_SESSION_ID' ``.
  - Para instancias de lectura, `session_id` es siempre el `UUID` (identificador único universal) de la instancia de lector.
- `durable_lsn`: número de secuencia de registro (LSN) que se guarda en el almacenamiento.
  - Para el volumen principal, es el LSN duradero del volumen (VDL) principal efectivo actualmente.
  - Para cualquier volumen secundario, es el VDL principal al que se ha aplicado correctamente el secundario.

> **Note**
>
> Un número de secuencia de registro (LSN) es un número secuencial único que identifica un registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción que ha tenido lugar más tarde en la secuencia.

- `highest_lsn_rcvd`: LSN más alto (más reciente) recibido por la instancia de base de datos de la instancia de base de datos de escritor.
- `current_read_lsn`: LSN de la instantánea más reciente que se ha aplicado a todos los lectores.
- `cur_replay_latency_in_usec`: número de microsegundos que se espera que tarden en reproducir el registro en el secundario.
- `active_txns`: número de transacciones activas actualmente.
- `is_current`: no se usa.
- `last_transport_error`: último código de error de replicación.
- `last_error_timestamp`: marca temporal del último error de replicación.
- `last_update_timestamp`: marca temporal de la última actualización del estado de la réplica. En Aurora PostgreSQL 13.9, el valor `last_update_timestamp` de la instancia de base de datos a la que está conectado se establece en `NULL`.
- `feedback_xmin`: la espera activa feedback_xmin de la réplica. ID de transacción activo mínimo (más antiguo) utilizado por la instancia de base de datos.
- `feedback_epoch`: fecha de inicio que utiliza la instancia de base de datos cuando genera información en espera activa.
- `replica_lag_in_msec`: tiempo en que esa instancia de lector se queda detrás de la instancia de escritor, en milisegundos.
- `log_stream_speed_in_kib_per_second`: rendimiento del flujo de registro en kilobytes por segundo.
- `log_buffer_sequence_number`: número de secuencia de búfer de registro.
- `oldest_read_view_trx_id`: no se usa.
- `oldest_read_view_lsn`: LSN más antiguo utilizado por la instancia de base de datos para leer desde el almacenamiento.
- `pending_read_ios`: lecturas de la página pendientes que aún deben replicarse.
- `read_ios`: número total de lecturas de página en la réplica.
- `iops`: no se usa.
- `cpu`: Uso de la CPU del daemon de almacenamiento de Aurora para cada nodo del clúster. Para obtener más información acerca del uso de CPU por parte de la instancia, consulte Métricas de nivel de instancia para Amazon Aurora.

Notas de uso

Esta función es compatible con todas las versiones de Aurora PostgreSQL disponibles actualmente. La función `aurora_replica_status` devuelve valores del gestor de estado de réplica de un clúster de base de datos de Aurora PostgreSQL. Puede utilizar esta función para obtener información sobre el estado de la replicación en el clúster de base de datos de Aurora PostgreSQL, incluidas las métricas de todas las instancias de base de datos del clúster de base de datos de Aurora. Por ejemplo, puede hacer lo siguiente:

- Obtenga información sobre el tipo de instancia (escritor, lector) en el clúster de base de datos de Aurora PostgreSQL: puede obtener esta información comprobando los valores de las columnas siguientes:
  - `server_id`: contiene el nombre de la instancia especificada cuando creó la instancia. En algunos casos, como para la instancia principal (escritor), el nombre se crea normalmente anexando -instance-1 al nombre que cree para su clúster de base de datos de Aurora PostgreSQL.
  - `session_id`: el campo `session_id` indica si la instancia es un lector o un escritor. Para una instancia de escritor, `session_id` está siempre configurado en `"MASTER_SESSION_ID"`. Para una instancia de lector, `session_id` está configurado con el `UUID` del lector específico.
- Diagnosticar problemas de replicación comunes, como un retraso de réplica: el retraso de réplica es el tiempo en milisegundos que la caché de página de una instancia de lector está por detrás del de la instancia de escritor. Este retraso se produce porque los clústeres de Aurora utilizan replicación asíncrona, tal como se describe en Replicación con Amazon Aurora. Se muestra en la columna `replica_lag_in_msec` en los resultados devueltos por esta función. El retraso también puede producirse cuando se cancela una consulta debido a conflictos con la recuperación en un servidor en espera. Puede comprobar `pg_stat_database_conflicts()` para verificar que dicho conflicto está provocando el retraso de la réplica (o no). Para obtener más información, consulte el tema sobre el recopilador de estadísticas en la documentación de PostgreSQL. Para obtener más información acerca de la alta disponibilidad y la replicación, consulte la sección de Preguntas frecuentes de Amazon Aurora.

  Amazon CloudWatch almacena los resultados de `replica_lag_in_msec` a lo largo del tiempo, como la métrica `AuroraReplicaLag`. Para obtener más información acerca de las métricas de CloudWatch para Aurora, consulte Supervisión de métricas de Amazon Aurora con Amazon CloudWatch.

Para obtener más información sobre la solución de problemas de réplicas de lectura y reinicios de Aurora, consulte la pregunta sobre ¿por qué mi réplica de lectura de Amazon Aurora se quedó atrás y se reinició? en el AWS SupportCenter.

Ejemplos

En el siguiente ejemplo se muestra cómo obtener el estado de replicación de todas las instancias de un clúster de base de datos de Aurora PostgreSQL:

```
=> SELECT *
FROM aurora_replica_status();
```

En el ejemplo siguiente se muestra la instancia de escritor en el clúster de base de datos de Aurora PostgreSQL `docs-lab-apg-main`:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id = 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
 db-119-001-instance-01 | writer
```

En el ejemplo siguiente se enumeran todas las instancias de lector de un clúster:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id <> 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
db-119-001-instance-02  | reader
db-119-001-instance-03  | reader
db-119-001-instance-04  | reader
db-119-001-instance-05  | reader
```

```
(4 rows)
```

En el siguiente ejemplo se enumeran todas las instancias, hasta qué punto está rezagada cada instancia con respecto al escritor y cuánto tiempo ha pasado desde la última actualización:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role,
    replica_lag_in_msec AS replica_lag_ms,
    round(extract (epoch FROM (SELECT age(clock_timestamp(), last_update_timestamp))) *
 1000) AS last_update_age_ms
FROM aurora_replica_status()
ORDER BY replica_lag_in_msec NULLS FIRST;
       server_id        | instance_role | replica_lag_ms | last_update_age_ms
------------------------+---------------+----------------+--------------------
 db-124-001-instance-03 | writer        |         [NULL] |               1756
 db-124-001-instance-01 | reader        |             13 |               1756
 db-124-001-instance-02 | reader        |             13 |               1756
(3 rows)
```

aurora_stat_activity

Devuelve una fila por proceso del servidor, que muestra información relacionada con la actividad actual de ese proceso.

Sintaxis

```
aurora_stat_activity();
```

Argumentos

Ninguno

Tipo de retorno

Devuelve una fila por proceso de servidor. Además de las columnas `pg_stat_activity`, se agrega el siguiente campo:

- planid - identificador del plan

Notas de uso

Una vista complementaria de `pg_stat_activity` devolviendo las mismas columnas con una columna `plan_id` adicional que muestra el plan de ejecución de la consulta actual.

`aurora_compute_plan_id` debe estar activado para que la vista devuelva un plan_id.

Esta función está disponible desde Aurora PostgreSQL versiones 14.10, 15.5, y para todas las demás versiones posteriores.

Ejemplos

El ejemplo de consulta que aparece a continuación agrega la carga superior mediante query_id y plan_id.

```
db1=# select count(*), query_id, plan_id
db1-# from aurora_stat_activity() where state = 'active'
db1-# and pid <> pg_backend_pid()
db1-# group by query_id, plan_id
db1-# order by 1 desc;

count |  query_id             |  plan_id
-------+----------------------+-------------
 11    | -5471422286312252535 | -2054628807
 3     | -6907107586630739258 | -815866029
 1     | 5213711845501580017  |  300482084
(3 rows)
```

Si el plan utilizado para query_id cambia, aurora_stat_activity indicará un nuevo plan_id.

```
count  |  query_id             |  plan_id
-------+----------------------+-------------
 10    | -5471422286312252535 | 1602979607
 1     | -6907107586630739258 | -1809935983
 1     | -2446282393000597155 | -207532066
(3 rows)
```

aurora_stat_backend_waits

Muestra estadísticas de la actividad de espera de un proceso de backend específico.

Sintaxis

```
aurora_stat_backend_waits(pid)
```

Argumentos

`pid`: el ID del proceso de backend. Puede obtener los ID de proceso mediante la vista de `pg_stat_activity`.

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `type_id`: un número que indica el tipo de evento de espera, como 1 para un bloqueo ligero (`LWLock`), 3 para un bloqueo o 6 para una sesión de cliente, por nombrar algunos ejemplos. Estos valores se vuelven significativos cuando se unen los resultados de esta función con columnas de la función `aurora_stat_wait_type`, tal y como se muestra en los Ejemplos.
- `event_id`: número de identificación del evento de espera. Una este valor con las columnas de `aurora_stat_wait_event` para obtener nombres de eventos significativos.
- `waits`: recuento del número de esperas acumuladas para el ID del proceso especificado.
- `wait_time`: tiempo de espera en milisegundos.

Notas de uso

Puede utilizar esta función para analizar eventos de espera específicos de backend (sesión) ocurridos desde que se abrió una conexión. Para obtener información más significativa sobre los nombres y tipos de eventos de espera, puede combinar esta función `aurora_stat_wait_type` y `aurora_stat_wait_event`, utilizando JOIN como se muestra en los ejemplos.

Ejemplos

En este ejemplo se muestran todas las esperas, tipos y nombres de eventos de un ID de proceso de backend 3027.

```
=> SELECT type_name, event_name, waits, wait_time
        FROM aurora_stat_backend_waits(3027)
```

```
NATURAL JOIN aurora_stat_wait_type()
NATURAL JOIN aurora_stat_wait_event();
type_name |       event_name       | waits | wait_time
----------+------------------------+-------+-----------
 LWLock    | ProcArrayLock          |     3 |         27
 LWLock    | wal_insert             |   423 |      16336
 LWLock    | buffer_content         | 11840 |    1033634
 LWLock    | lock_manager           | 23821 |    5664506
 Lock      | tuple                  | 10258 |  152280165
 Lock      | transactionid          | 78340 | 1239808783
 Client    | ClientRead             | 34072 |   17616684
 IO        | ControlFileSyncUpdate  |     2 |          0
 IO        | ControlFileWriteUpdate |     4 |         32
 IO        | RelationMapRead        |     2 |        795
 IO        | WALWrite               | 36666 |      98623
 IO        | XactSync               |  4867 |    7331963
```

En este ejemplo se muestran los tipos de espera actuales y acumulados y los eventos de espera de todas las sesiones activas (`pg_stat_activity state <> 'idle'`) (pero sin la sesión actual que invoca la función (`pid <> pg_backend_pid()`).

```
=> SELECT a.pid,
            a.usename,
            a.app_name,
            a.current_wait_type,
            a.current_wait_event,
            a.current_state,
            wt.type_name AS wait_type,
            we.event_name AS wait_event,
            a.waits,
            a.wait_time
       FROM (SELECT pid,
                    usename,
                    left(application_name,16) AS app_name,
                    coalesce(wait_event_type,'CPU') AS current_wait_type,
                    coalesce(wait_event,'CPU') AS current_wait_event,
                    state AS current_state,
                    (aurora_stat_backend_waits(pid)).*
               FROM pg_stat_activity
              WHERE pid <> pg_backend_pid()
                AND state <> 'idle') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we;
```

```
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |       wait_event       | waits | wait_time
-------+----------+----------+-------------------+--------------------+---------------
+----------+-----------------------+-------+-----------
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1937 |     29975
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22903 |    760498
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | lock_manager           | 10012 |    223207
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple                  | 20315 |  63081529
 .
 .
 .
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 IO        | WALWrite               | 93293 |    237440
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 IO        | XactSync               | 13010 |  19525143
 30100 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | ProcArrayLock          |     6 |        53
 30100 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1913 |     25450
 30100 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22874 |    778005
 .
 .
 .
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | ProcArrayLock          |     3 |        71
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | wal_insert             |  1940 |     27741
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | buffer_content         | 22962 |    776352
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | lock_manager           |  9879 |    218826
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | tuple                  | 20401 |  63581306
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | transactionid          | 50769 | 211645008
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 Client    | ClientRead             | 89901 |  44192439
```

En este ejemplo se muestran los tres (3) eventos de tipo de espera acumulativos y de espera principales para todas las sesiones activas (`pg_stat_activity state <> 'idle'`) excluyendo la sesión actual (`pid <>pg_backend_pid()`).

```
=> SELECT top3.*
       FROM (SELECT a.pid,
                    a.usename,
                    a.app_name,
                    a.current_wait_type,
                    a.current_wait_event,
                    a.current_state,
                    wt.type_name AS wait_type,
                    we.event_name AS wait_event,
                    a.waits,
                    a.wait_time,
                    RANK() OVER (PARTITION BY pid ORDER BY a.wait_time DESC)
               FROM (SELECT pid,
                            usename,
                            left(application_name,16) AS app_name,
                            coalesce(wait_event_type,'CPU') AS current_wait_type,
                            coalesce(wait_event,'CPU') AS current_wait_event,
                            state AS current_state,
                            (aurora_stat_backend_waits(pid)).*
                       FROM pg_stat_activity
                      WHERE pid <> pg_backend_pid()
                        AND state <> 'idle') a
       NATURAL JOIN aurora_stat_wait_type() wt
       NATURAL JOIN aurora_stat_wait_event() we) top3
 WHERE RANK <=3;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |   wait_event    |  waits  | wait_time  | rank
-------+----------+----------+-------------------+--------------------+--------------
+-----------+-----------------+---------+------------+------
 20567 | postgres | psql     | CPU               | CPU                | active        |
 LWLock    | wal_insert      |   25000 |   67512003 |    1
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | WALWrite        | 3071758 |    1016961 |    2
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | BufFileWrite    |   20750 |     184559 |    3
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | transactionid   |  237350 | 1265580011 |    1
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple           |   93641 |  341472318 |    2
```

```
27743 | postgres | pgbench  | Lock              | transactionid      | active        |
Client    | ClientRead      |  417556 |  204796837 |    3
.
.
.
27745 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | transactionid   |  238068 | 1265816822 |    1
27745 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | tuple           |   93210 |  338312247 |    2
27745 | postgres | pgbench  | IO                | XactSync           | active        |
Client    | ClientRead      |  419157 |  207836533 |    3
27746 | postgres | pgbench  | Lock              | transactionid      | active        |
Lock      | transactionid   |  237621 | 1264528811 |    1
27746 | postgres | pgbench  | Lock              | transactionid      | active        |
Lock      | tuple           |   93563 |  339799310 |    2
27746 | postgres | pgbench  | Lock              | transactionid      | active        |
Client    | ClientRead      |  417304 |  208372727 |    3
```

aurora_stat_bgwriter

`aurora_stat_bgwriter` es una vista de estadísticas que muestra información sobre las lecturas optimizadas y las escrituras en caché.

Sintaxis

```
aurora_stat_bgwriter()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con todas las columnas `pg_stat_bgwriter` y las siguientes columnas adicionales. Para obtener más información sobre las columnas `pg_stat_bgwriter`, consulte `pg_stat_bgwriter`.

Puede restablecer las estadísticas de esta función utilizando `pg_stat_reset_shared("bgwriter")`.

- `orcache_blks_written`: número total de lecturas optimizadas escritas en caché y bloques de datos.

- `orcache_blk_write_time`: si se habilita `track_io_timing`, se registra el tiempo total dedicado a escribir lecturas optimizadas en bloques de archivos de datos de caché, en milisegundos. Para obtener más información, consulte track_io_timing.

Notas de uso

Esta función está disponible en las siguientes versiones de Aurora PostgreSQL:

- Versión 15.4 y versiones posteriores a la 15
- Versión 14.9 y versiones posteriores a la 14

Ejemplos

```
=> select * from aurora_stat_bgwriter();
-[ RECORD 1 ]-----------------+-----------
orcache_blks_written          | 246522
orcache_blk_write_time        | 339276.404
```

aurora_stat_database

Contiene todas las columnas de pg_stat_database y, al final, añade nuevas columnas.

Sintaxis

```
aurora_stat_database()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con todas las columnas `pg_stat_database` y las siguientes columnas adicionales. Para obtener más información sobre las columnas `pg_stat_database`, consulte `pg_stat_database`.

- `storage_blks_read`: número total de bloques compartidos leídos desde el almacenamiento de Aurora en esta base de datos.

- `orcache_blks_hit`: número total de visitas a la caché de lecturas optimizadas de esta base de datos.
- `local_blks_read`: número total de bloques locales leídos en esta base de datos.
- `storage_blk_read_time`: si se habilita `track_io_timing`, se registra el tiempo total dedicado a leer bloques de archivos de datos desde el almacenamiento de Aurora (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.
- `local_blk_read_time`: si se habilita `track_io_timing`, se registra el tiempo total dedicado a leer bloques de archivos de datos locales, en milisegundos; de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.
- `orcache_blk_read_time`: si se habilita `track_io_timing`, se registra el tiempo total dedicado a leer bloques de archivos de datos desde la caché de lecturas optimizadas (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.

> **Note**
>
> El valor de `blks_read` es la suma de`storage_blks_read` `orcache_blks_hit` y `local_blks_read`.
> El valor de `blk_read_time` es la suma de`storage_blk_read_time` `orcache_blk_read_time` y `local_blk_read_time`.

Notas de uso

Esta función está disponible en las siguientes versiones de Aurora PostgreSQL:

- Versión 15.4 y versiones posteriores a la 15
- Versión 14.9 y versiones posteriores a la 14

Ejemplos

El siguiente ejemplo muestra cómo incluye todas las columnas `pg_stat_database` y añade 6 columnas nuevas al final:

```
=> select * from aurora_stat_database() where datid=14717;
-[ RECORD 1 ]------------+------------------------------
datid                    | 14717
```

```
datname                  | postgres
numbackends              | 1
xact_commit              | 223
xact_rollback            | 4
blks_read                | 1059
blks_hit                 | 11456
tup_returned             | 27746
tup_fetched              | 5220
tup_inserted             | 165
tup_updated              | 42
tup_deleted              | 91
conflicts                | 0
temp_files               | 0
temp_bytes               | 0
deadlocks                | 0
checksum_failures        |
checksum_last_failure    |
blk_read_time            | 3358.689
blk_write_time           | 0
session_time             | 1076007.997
active_time              | 3684.371
idle_in_transaction_time | 0
sessions                 | 10
sessions_abandoned       | 0
sessions_fatal           | 0
sessions_killed          | 0
stats_reset              | 2023-01-12 20:15:17.370601+00
orcache_blks_hit         | 425
orcache_blk_read_time    | 89.934
storage_blks_read        | 623
storage_blk_read_time    | 3254.914
local_blks_read          | 0
local_blk_read_time      | 0
```

aurora_stat_dml_activity

Informa la actividad acumulativa para cada tipo de operación de lenguaje de manipulación de datos (DML) en una base de datos en un clúster de Aurora PostgreSQL.

Sintaxis

```
aurora_stat_dml_activity(database_oid)
```

Argumentos

database_oid

ID de objeto (OID) de la base de datos en el clúster de Aurora PostgreSQL.

Tipo de retorno

Registro SETOF

Notas de uso

La función `aurora_stat_dml_activity` solo está disponible con Aurora PostgreSQL versión 3.1 compatible con el motor de PostgreSQL 11.6 y versiones posteriores.

Utilice esta función en clústeres de Aurora PostgreSQL con un gran número de bases de datos para identificar qué bases de datos tienen más actividad DML o más lenta, o ambas.

La función `aurora_stat_dml_activity` devuelve el número de veces que se ejecutaron las operaciones y la latencia acumulada en microsegundos para las operaciones SELECT, INSERT, UPDATE y DELETE. El informe solo incluye operaciones DML correctas.

Puede restablecer esta estadística mediante la función de acceso a estadísticas de PostgreSQL `pg_stat_reset`. Puede comprobar la última vez que se restableció esta estadística mediante la opción de función `pg_stat_get_db_stat_reset_time`. Para obtener más información acerca de las funciones de acceso a las estadísticas de PostgreSQL, consulte Recopilador de estadísticas en la documentación de PostgreSQL.

Ejemplos

En el siguiente ejemplo se muestra cómo informar de las estadísticas de actividad de DML para la base de datos conectada.

```
--Define the oid variable from connected database by using \gset
=> SELECT oid,
          datname
     FROM pg_database
    WHERE datname=(select current_database()) \gset
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
select_count | select_latency_microsecs | insert_count | insert_latency_microsecs |
 update_count | update_latency_microsecs | delete_count | delete_latency_microsecs
```

```
--------------+--------------------------+--------------+--------------------------
+--------------+--------------------------+--------------+--------------------------
       178957 |                 66684115 |       171065 |                 28876649 |
    519538 |            1454579206167 |            1 |                      53027


-- Showing the same results with expanded display on
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
-[ RECORD 1 ]------------+--------------
select_count             | 178957
select_latency_microsecs | 66684115
insert_count             | 171065
insert_latency_microsecs | 28876649
update_count             | 519538
update_latency_microsecs | 1454579206167
delete_count             | 1
delete_latency_microsecs | 53027
```

En el ejemplo siguiente se muestran las estadísticas de actividad de DML para todas las bases de datos del clúster de Aurora PostgreSQL. Este clúster tiene dos bases de datos: `postgres` y `mydb`. La lista separada por comas corresponde a los campos `select_count`, `select_latency_microsecs`, `insert_count`, `insert_latency_microsecs`, `update_count`, `update_latency_microsecs`, `delete_count` y `delete_latency_microsecs`.

Aurora PostgreSQL crea y utiliza una base de datos del sistema llamada `rdsadmin` para admitir operaciones administrativas como copias de seguridad, restauraciones, comprobaciones de estado, replicación, etc. Estas operaciones DML no tienen ningún impacto en su clúster de Aurora PostgreSQL.

```
=> SELECT oid,
    datname,
    aurora_stat_dml_activity(oid)
    FROM pg_database;
oid  |    datname     |                   aurora_stat_dml_activity
-------+---------------
+-------------------------------------------------------------------
 14006 | template0      | (,,,,,,,)
 16384 | rdsadmin       | (2346623,1211703821,4297518,817184554,0,0,0,0)
     1 | template1      | (,,,,,,,)
```

```
 14007 | postgres       |
(178961,66716329,171065,28876649,519538,1454579206167,1,53027)
 16401 | mydb           | (200246,64302436,200036,107101855,600000,83659417514,0,0)
```

En el siguiente ejemplo se muestran las estadísticas de actividad de DML para todas las bases de datos, organizadas en columnas para una mejor legibilidad.

```
SELECT db.datname,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 1), '()') AS select_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 2), '()') AS select_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 3), '()') AS insert_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 4), '()') AS insert_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 5), '()') AS update_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 6), '()') AS update_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 7), '()') AS delete_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 8), '()') AS delete_latency_microsecs
FROM  (SELECT datname,
              aurora_stat_dml_activity(oid) AS asdmla
         FROM pg_database
      ) AS db;

    datname     | select_count | select_latency_microsecs | insert_count |
 insert_latency_microsecs | update_count | update_latency_microsecs | delete_count |
 delete_latency_microsecs
----------------+--------------+--------------------------+--------------
+--------------------------+--------------+--------------------------+--------------
+--------------------------
 template0      |              |                          |              |
             |              |                          |              |
 rdsadmin       | 4206523      | 2478812333               | 7009414      | 1338482258
             | 0            | 0                        | 0            | 0
 template1      |              |                          |              |
             |              |                          |              |
 fault_test     | 66           | 452099                   | 0            | 0
             | 0            | 0                        | 0            | 0
 db_access_test | 1            | 5982                     | 0            | 0
             | 0            | 0                        | 0            | 0
 postgres       | 42035        | 95179203                 | 5752         | 2678832898
             | 21157        | 441883182488             | 2            | 1520
 mydb           | 71           | 453514                   | 0            | 0
             | 1            | 190                      | 1            | 152
```

En el ejemplo siguiente se muestra la latencia acumulativa media (latencia acumulada dividida por recuento) para cada operación DML de la base de datos con el OID16401.

```
=> SELECT select_count,
          select_latency_microsecs,
          select_latency_microsecs/NULLIF(select_count,0) select_latency_per_exec,
          insert_count,
          insert_latency_microsecs,
          insert_latency_microsecs/NULLIF(insert_count,0) insert_latency_per_exec,
          update_count,
          update_latency_microsecs,
          update_latency_microsecs/NULLIF(update_count,0) update_latency_per_exec,
          delete_count,
          delete_latency_microsecs,
          delete_latency_microsecs/NULLIF(delete_count,0) delete_latency_per_exec
     FROM aurora_stat_dml_activity(16401);
-[ RECORD 1 ]------------+-------------
select_count             | 451312
select_latency_microsecs | 80205857
select_latency_per_exec  | 177
insert_count             | 451001
insert_latency_microsecs | 123667646
insert_latency_per_exec  | 274
update_count             | 1353067
update_latency_microsecs | 200900695615
update_latency_per_exec  | 148478
delete_count             | 12
delete_latency_microsecs | 448
delete_latency_per_exec  | 37
```

aurora_stat_get_db_commit_latency

Obtiene la latencia de confirmación acumulada en microsegundos para las bases de datos de Aurora PostgreSQL. La latencia de confirmación se mide como el tiempo transcurrido entre el momento en que un cliente envía una solicitud de confirmación y el momento en que recibe el acuse de recibo de confirmación.

Sintaxis

```
aurora_stat_get_db_commit_latency(database_oid)
```

Argumentos

database_oid

ID de objeto (OID) de la base de datos de Aurora PostgreSQL.

Tipo de retorno

Registro SETOF

Notas de uso

Amazon CloudWatch utiliza esta función para calcular la latencia media de confirmación. Para obtener más información acerca de las métricas de Amazon CloudWatch y cómo solucionar problemas de latencia de confirmación alta, consulte Consulta de métricas en la consola de Amazon RDS y Cómo tomar mejores decisiones sobre Amazon RDS con las métricas de Amazon CloudWatch.

Puede restablecer esta estadística mediante la función de acceso a estadísticas de PostgreSQL `pg_stat_reset`. Puede comprobar la última vez que se restableció esta estadística mediante la opción de función `pg_stat_get_db_stat_reset_time`. Para obtener más información acerca de las funciones de acceso a las estadísticas de PostgreSQL, consulte Recopilador de estadísticas en la documentación de PostgreSQL.

Ejemplos

En el siguiente ejemplo se obtiene la latencia de confirmación acumulada de cada base de datos del clúster.`pg_database`.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database;

  oid  |    datname     | aurora_stat_get_db_commit_latency
-------+----------------+-----------------------------------
 14006 | template0      |                                 0
 16384 | rdsadmin       |                         654387789
     1 | template1      |                                 0
 16401 | mydb           |                            229556
 69768 | postgres       |                             22011
```

En el siguiente ejemplo se obtiene la latencia de confirmación acumulativa de la base de datos conectada actualmente. Antes de llamar al método `aurora_stat_get_db_commit_latency`, el ejemplo primero usa `\gset` para definir una variable para el argumento `oid` y establece su valor desde la base de datos conectada.

```
--Get the oid value from the connected database before calling
 aurora_stat_get_db_commit_latency
=> SELECT oid
     FROM pg_database
    WHERE datname=(SELECT current_database()) \gset
=> SELECT *
     FROM aurora_stat_get_db_commit_latency(:oid);

 aurora_stat_get_db_commit_latency
-----------------------------------
                        1424279160
```

En el siguiente ejemplo se obtiene la latencia de confirmación acumulada para la base de datos `mydb` en el clúster `pg_database`. A continuación, se restablece esta estadística mediante la función `pg_stat_reset` y muestra los resultados. Por último, utiliza el rol `pg_stat_get_db_stat_reset_time` para comprobar la última vez que se restableció esta estadística.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database
    WHERE datname = 'mydb';

  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                           3320370


=> SELECT pg_stat_reset();
 pg_stat_reset
---------------


=> SELECT oid,
          datname,
          aurora_stat_get_db_commit_latency(oid)
```

```
     FROM pg_database
    WHERE datname = 'mydb';
  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                                 6


=> SELECT *
     FROM pg_stat_get_db_stat_reset_time(16427);

 pg_stat_get_db_stat_reset_time
--------------------------------
 2021-04-29 21:36:15.707399+00
```

aurora_stat_logical_wal_cache

Se muestra el uso de la caché del registro de escritura (WAL) lógica por ranura.

Sintaxis

```
SELECT * FROM aurora_stat_logical_wal_cache()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `name`: el nombre de la ranura de replicación.
- `active_pid`: el identificador del proceso walsender.
- `cache_hit`: el número total de aciertos de caché de wal desde el último restablecimiento.
- `cache_miss`: el número total de fallos de caché de wal desde el último restablecimiento.
- `blks_read`: el número total de solicitudes de lectura de la caché de wal.
- `hit_rate`: la tasa de aciertos de caché de WAL (cache_hit/blks_read).
- `last_reset_timestamp`: la última vez que se restableció el contador.

Notas de uso

Esta función está disponible para las siguientes versiones de Aurora PostgreSQL:

- Versiones 15.2 y posteriores
- Versión 14.7 y posteriores
- Versión 13.8 y posteriores
- Versión 12.12 y posteriores
- Versión 11.17 y posteriores

Ejemplos

En el siguiente ejemplo, se muestran dos ranuras de replicación con una sola función `aurora_stat_logical_wal_cache` activa.

```
=> SELECT *
     FROM aurora_stat_logical_wal_cache();
    name    | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
------------+------------+-----------+------------+-----------+----------
+-------------------------------
 test_slot1 |      79183 |        24 |          0 |        24 | 100.00%  | 2022-08-05
 17:39:56.830635+00
 test_slot2 |            |         1 |          0 |         1 | 100.00%  | 2022-08-05
 17:34:04.036795+00
(2 rows)
```

aurora_stat_memctx_usage

Informa del uso del contexto de memoria para cada proceso de PostgreSQL.

Sintaxis

```
aurora_stat_memctx_usage()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `pid`: el ID del proceso.
- `name`: el nombre del contexto.
- `allocated`: el número de bytes obtenidos del subsistema de memoria subyacente por el contexto de memoria.
- `used`: el número de bytes asignados a los clientes del contexto de memoria.
- `instances`: el recuento de contextos de este tipo existentes actualmente.

Notas de uso

Esta función muestra el uso del contexto de memoria para cada proceso de PostgreSQL. Algunos procesos están etiquetados `anonymous`. Los procesos no están expuestos porque contienen palabras clave restringidas.

Esta función está disponible a partir de las siguientes versiones de Aurora PostgreSQL:

- Versión 15.3 y versiones posteriores a la 15
- Versión 14.8 y versiones posteriores a la 14
- Versión 13.11 y versiones posteriores a la 13
- Versión 12.15 y versiones posteriores a la 12
- Versión 11.20 y versiones posteriores a la 11

Ejemplos

En el ejemplo siguiente se muestran los resultados de llamar a la función `aurora_stat_memctx_usage`.

```
=> SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                             | allocated |   used  | instances
-------+----------------------------------+-----------+---------+-----------
123864 | Miscellaneous                    |     19520 |   15064 |         3
123864 | Aurora File Context              |      8192 |     616 |         1
123864 | Aurora WAL Context               |      8192 |     296 |         1
```

```
123864 | CacheMemoryContext              |    524288 |  422600 |         1
123864 | Catalog tuple context           |     16384 |   13736 |         1
123864 | ExecutorState                   |     32832 |   28304 |         1
123864 | ExprContext                     |      8192 |    1720 |         1
123864 | GWAL record construction        |      1024 |     832 |         1
123864 | MdSmgr                          |      8192 |     296 |         1
123864 | MessageContext                  |    532480 |  353832 |         1
123864 | PortalHeapMemory                |      1024 |     488 |         1
123864 | PortalMemory                    |      8192 |     576 |         1
123864 | printtup                        |      8192 |     296 |         1
123864 | RelCache hash table entries     |      8192 |    8152 |         1
123864 | RowDescriptionContext           |      8192 |    1344 |         1
123864 | smgr relation context           |      8192 |     296 |         1
123864 | Table function arguments        |      8192 |     352 |         1
123864 | TopTransactionContext           |      8192 |     632 |         1
123864 | TransactionAbortContext         |     32768 |     296 |         1
123864 | WAL record construction         |     50216 |   43904 |         1
123864 | hash table                      |     65536 |   52744 |         6
123864 | Relation metadata               |    191488 |  124240 |        87
104992 | Miscellaneous                   |      9280 |    7728 |         3
104992 | Aurora File Context             |      8192 |     376 |         1
104992 | Aurora WAL Context              |      8192 |     296 |         1
104992 ||Autovacuum Launcher             |      8192 |     296 |         1
104992 | Autovacuum database list        |     16384 |     744 |         2
104992 | CacheMemoryContext              |    262144 |  140288 |         1
104992 | Catalog tuple context           |      8192 |     296 |         1
104992 | GWAL record construction        |      1024 |     832 |         1
104992 | MdSmgr                          |      8192 |     296 |         1
104992 | PortalMemory                    |      8192 |     296 |         1
104992 | RelCache hash table entries     |      8192 |     296 |         1
104992 | smgr relation context           |      8192 |     296 |         1
104992 | Autovacuum start worker (tmp)   |      8192 |     296 |         1
104992 | TopTransactionContext           |     16384 |     592 |         2
104992 | TransactionAbortContext         |     32768 |     296 |         1
104992 | WAL record construction         |     50216 |   43904 |         1
104992 | hash table                      |     49152 |   34024 |         4
(39 rows)
```

Algunas palabras clave restringidas se ocultarán y el resultado tendrá el siguiente aspecto:

```
postgres=>SELECT *
     FROM aurora_stat_memctx_usage();
```

```
    pid| name                            | allocated |   used  | instances
-------+---------------------------------+-----------+---------+-----------
  5482 | anonymous                       |      8192 |     456 |         1
  5482 | anonymous                       |      8192 |     296 |         1
```

aurora_stat_optimized_reads_cache

Esta función muestra las estadísticas de la caché por niveles.

Sintaxis

```
aurora_stat_optimized_reads_cache()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- `total_size`: tamaño total de la caché de lecturas optimizadas.
- `used_size`: tamaño de página utilizado en la caché de lecturas optimizadas.

Notas de uso

Esta función está disponible en las siguientes versiones de Aurora PostgreSQL:

- Versión 15.4 y versiones posteriores a la 15
- Versión 14.9 y versiones posteriores a la 14

Ejemplos

A continuación se muestra la salida en una instancia r6gd.8xlarge:

```
=> select pg_size_pretty(total_size) as total_size, pg_size_pretty(used_size)
               as used_size from aurora_stat_optimized_reads_cache();
total_size | used_size
-----------+-----------
1054 GB    | 975 GB
```

aurora_stat_plans

Devuelve una fila por cada plan de ejecución rastreado.

Sintaxis

```
aurora_stat_plans(
    showtext
)
```

Argumentos

- showtext: muestra el texto de la consulta y del plan. Los valores válidos son NULL, true y false. True mostrará el texto de la consulta y del plan.

Tipo de retorno

Devuelve una fila para cada plan rastreado que contiene todas las columnas de `aurora_stat_statements` y las siguientes columnas adicionales.

- planid: identificador del plan
- explain_plan: explica el texto del plan
- plan_type:
  - `no plan` - no se capturó ningún plan
  - `estimate` - plan capturado con costos estimados
  - `actual` - plan capturado con EXPLAIN ANALYZE
- plan_captured_time: última vez que se capturó un plan

Notas de uso

`aurora_compute_plan_id` debe estar activado y `pg_stat_statements` debe ser `shared_preload_libraries` para que se realice un seguimiento de los planes.

El número de planes disponibles se controla mediante el valor establecido en el parámetro `pg_stat_statements.max`. Cuando `aurora_compute_plan_id` está habilitado, puede realizar un seguimiento de los planes hasta el valor especificado en `aurora_stat_plans`.

Esta función está disponible desde Aurora PostgreSQL versiones 14.10, 15.5, y para todas las demás versiones posteriores.

Ejemplos

En el ejemplo siguiente, se capturan los dos planes que corresponden al identificador de consulta -5471422286312252535 y el planid rastrea las estadísticas de los estados.

```
db1=# select calls, total_exec_time, planid, plan_captured_time, explain_plan
db1-# from aurora_stat_plans(true)
db1-# where queryid = '-5471422286312252535'

calls    |  total_exec_time   |   planid    |      plan_captured_time       |
                explain_plan
---------+--------------------+-------------+-------------------------------
+-----------------------------------------------------------------
 1532632 |  3209846.097107853 |  1602979607 | 2023-10-31 03:27:16.925497+00 | Update on
 pgbench_branches                                        +
         |                    |             |                               |   ->
 Bitmap Heap Scan on pgbench_branches                       +
         |                    |             |                               |
 Recheck Cond: (bid = 76)                                     +
         |                    |             |                               |         -
>  Bitmap Index Scan on pgbench_branches_pkey          +
         |                    |             |                               |
     Index Cond: (bid = 76)
   61365 | 124078.18012200127 | -2054628807 | 2023-10-31 03:20:09.85429+00  | Update on
 pgbench_branches                                        +
         |                    |             |                               |   ->
 Index Scan using pgbench_branches_pkey on pgbench_branches+
         |                    |             |                               |
 Index Cond: (bid = 17)
```

aurora_stat_reset_wal_cache

Restablece el contador de la memoria caché de wal lógica.

Sintaxis

Para restablecer una ranura específica

```
SELECT * FROM aurora_stat_reset_wal_cache('slot_name')
```

Para restablecer todas las ranuras

```
SELECT * FROM aurora_stat_reset_wal_cache(NULL)
```

Argumentos

`NULL` o `slot_name`

Tipo de retorno

Mensaje de estado, cadena de texto

- Restablece el contador de la memoria caché de wal lógica: mensaje de éxito. Este texto se devuelve cuando la función se ejecuta correctamente.
- No se encontró la ranura de replicación. Inténtelo de nuevo. - Mensaje de error Este texto se devuelve cuando la función no se ejecuta correctamente.

Notas de uso

Esta función está disponible para las siguientes versiones:

- Aurora PostgreSQL 14.5 y versiones posteriores
- Aurora PostgreSQL versión 13.8 y posteriores
- Aurora PostgreSQL versión 12.12 y posteriores
- Aurora PostgreSQL versión 11.7 y posteriores

Ejemplos

En el siguiente ejemplo, se utiliza la función `aurora_stat_reset_wal_cache` para restablecer una ranura denominada `test_results` y, a continuación, se intenta restablecer una ranura que no existe.

```
=> SELECT *
     FROM aurora_stat_reset_wal_cache('test_slot');
aurora_stat_reset_wal_cache
-------------------------------------
```

```
 Reset the logical wal cache counter.
(1 row)
=> SELECT *
     FROM aurora_stat_reset_wal_cache('slot-not-exist');
aurora_stat_reset_wal_cache
-----------------------------------------------
 Replication slot not found. Please try again.
(1 row)
```

aurora_stat_resource_usage

Informa del uso de los recursos en tiempo real, que consiste en las métricas de los recursos de backend y el uso de CPU para todos los procesos de backend de Aurora PostgreSQL.

Sintaxis

```
aurora_stat_resource_usage()
```

Argumentos

Ninguno

Tipo de devolución

Registro SETOF con columnas:

- pid: identificador de procesos
- allocated_memory: memoria total asignada por proceso en bytes
- used_memory: memoria realmente utilizada por el proceso en bytes
- cpu_usage_percent: porcentaje de uso de CPU del proceso

Notas de uso

Esta función muestra el uso de recursos de backend para cada proceso de backend de Aurora PostgreSQL.

Esta función está disponible a partir de las siguientes versiones de Aurora PostgreSQL:

- Aurora PostgreSQL 17.5 y versiones posteriores a la 17
- Aurora PostgreSQL 16.9 y versiones posteriores a la 16

- Aurora PostgreSQL 15.13 y versiones posteriores a la 15
- Aurora PostgreSQL 14.18 y versiones posteriores a la 14
- Aurora PostgreSQL 13.21 y versiones posteriores a la 13

Ejemplos

En el siguiente ejemplo se muestra el resultado de la función `aurora_stat_resource_usage`.

```
=> select * from aurora_stat_resource_usage();
 pid  | allocated_memory | used_memory |   cpu_usage_percent
------+------------------+-------------+-----------------------
  666 |          1074032 |      333544 |   0.00729274882897963
  667 |           787312 |      287360 | 0.0029263928146372746
  668 |          3076776 |     1563488 |  0.006013116835953961
  684 |           803744 |      307480 |  0.002226855426881142
 2401 |          1232992 |      943144 |                     0
  647 |             8000 |         944 |   0.48853387812429855
  659 |           319344 |      243000 | 0.0004135602076683591
  663 |           262000 |      185736 |  0.008181301476644002
  664 |             9024 |        1216 |   0.10992313082653653
(9 rows)
```

aurora_stat_statements

Muestra todas las columnas `pg_stat_statements` y añade más columnas al final.

Sintaxis

```
aurora_stat_statements(showtext boolean)
```

Argumentos

showtext boolean

Tipo de retorno

Registro SETOF con todas las columnas `pg_stat_statements` y las siguientes columnas adicionales. Para obtener más información sobre las columnas `pg_stat_statements`, consulte `pg_stat_statements`.

Puede restablecer las estadísticas de esta función utilizando `pg_stat_statements_reset()`.

- `storage_blks_read`: número total de bloques compartidos leídos desde el almacenamiento de Aurora por esta instrucción.
- `orcache_blks_hit`: número total de visitas a la caché de lecturas optimizadas por esta instrucción.
- `storage_blk_read_time`: si se habilita `track_io_timing`, rastrea el tiempo total que la instrucción ha dedicado a leer bloques compartidos desde el almacenamiento de Aurora (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.
- `local_blk_read_time`: si se habilita `track_io_timing`, rastrea el tiempo total que la instrucción ha dedicado a leer bloques locales (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.
- `orcache_blk_read_time`: si se habilita `track_io_timing`, rastrea el tiempo total que la instrucción ha dedicado a leer bloques compartidos desde la caché de lecturas optimizadas (en milisegundos); de lo contrario, el valor es cero. Para obtener más información, consulte track_io_timing.
- `total_plan_peakmem`: suma total de los valores máximos de memoria durante la fase de planificación para todas las llamadas a esta instrucción. Para ver el promedio de picos de memoria durante la planificación de la instrucción, divida este valor por el número de llamadas.
- `min_plan_peakmem`: el valor máximo de memoria más bajo registrado durante la planificación en todas las llamadas incluidas en esta instrucción.
- `max_plan_peakmem`: el valor máximo de memoria más alto registrado durante la planificación en todas las llamadas a esta instrucción.
- `total_exec_peakmem`: suma total de los valores máximos de memoria durante la fase de ejecución para todas las llamadas a esta instrucción. Para ver el promedio de picos de memoria durante la ejecución de la instrucción, divida este valor por el número de llamadas.
- `min_exec_peakmem`: el valor máximo de memoria más bajo, en bytes, visto durante la ejecución en todas las llamadas a esta instrucción.
- `max_exec_peakmem`: el valor máximo de memoria más alto, en bytes, visto durante la ejecución en todas las llamadas a esta instrucción.

> **Note**
>
> `total_plan_peakmen`, `min_plan_peakmem` y `max_plan_peakmem` solo se supervisan cuando `pg_stat_statements.track_planning` está activado.

Notas de uso

Para utilizar la función aurora_stat_statements(), debe incluir la extensión `pg_stat_statements` en el parámetro `shared_preload_libraries`.

Esta función está disponible en las siguientes versiones de Aurora PostgreSQL:

- Versión 15.4 y versiones posteriores a la 15
- Versión 14.9 y versiones posteriores a la 14

Las columnas que muestran los picos de memoria están disponibles a partir de las siguientes versiones:

- Versión 16.3 y posteriores
- Versión 15.7 y posteriores
- Versión 14.12 y posteriores

Ejemplos

El siguiente ejemplo muestra cómo incluye todas las columnas pg_stat_statements y añade 11 columnas nuevas al final:

```
=> select * from aurora_stat_statements(true) where query like 'with window_max%';
-[ RECORD 1 ]----------
+-----------------------------------------------------------------------------------------------
userid                 | 16409
dbid                   | 5
toplevel               | t
queryid                | -8347523682669847482
query                  | with window_max as (select custid, max(scratch) over (order by
 scratch rows between $1 preceding
and $2 following) wmax from ts) select sum(wmax), max(custid) from window_max
plans                  | 0
```

```
total_plan_time        | 0
min_plan_time          | 0
max_plan_time          | 0
mean_plan_time         | 0
stddev_plan_time       | 0
calls                  | 4
total_exec_time        | 254.105121
min_exec_time          | 57.503164000000005
max_exec_time          | 68.687418
mean_exec_time         | 63.52628025
stddev_exec_time       | 5.150765359979643
rows                   | 4
shared_blks_hit        | 200192
shared_blks_read       | 0
shared_blks_dirtied    | 0
shared_blks_written    | 0
local_blks_hit         | 0
local_blks_read        | 0
local_blks_dirtied     | 0
local_blks_written     | 0
temp_blks_read         | 0
temp_blks_written      | 0
blk_read_time          | 0
blk_write_time         | 0
temp_blk_read_time     | 0
temp_blk_write_time    | 0
wal_records            | 0
wal_fpi                | 0
wal_bytes              | 0
jit_functions          | 0
jit_generation_time    | 0
jit_inlining_count     | 0
jit_inlining_time      | 0
jit_optimization_count | 0
jit_optimization_time  | 0
jit_emission_count     | 0
jit_emission_time      | 0
storage_blks_read      | 0
orcache_blks_hit       | 0
storage_blk_read_time  | 0
local_blk_read_time    | 0
orcache_blk_read_time  | 0
total_plan_peakmem     | 0
min_plan_peakmem       | 0
```

```
max_plan_peakmem       | 0
total_exec_peakmem     | 6356224
min_exec_peakmem       | 1589056
max_exec_peakmem       | 1589056
```

aurora_stat_system_waits

Informa la información de eventos de espera para la instancia de base de datos de Aurora PostgreSQL.

Sintaxis

```
aurora_stat_system_waits()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF

Notas de uso

Esta función devuelve el número acumulativo de esperas y el tiempo de espera acumulativo para cada evento de espera generado por la instancia de base de datos a la que está conectado actualmente.

El conjunto de registros devuelto incluye los siguientes campos:

- `type_id` – El ID del tipo de evento de espera.
- `event_id` – El ID del evento de espera.
- `waits` – El número de veces que se ha producido el evento de espera.
- `wait_time` – La cantidad total de tiempo en microsegundos transcurrido esperando este evento.

Las estadísticas devueltas por esta función se restablecen cuando se reinicia una instancia de base de datos.

Ejemplos

En el ejemplo siguiente se muestran los resultados de llamar a la función `aurora_stat_system_waits`.

```
=> SELECT *
    FROM aurora_stat_system_waits();
 type_id | event_id  |   waits   |  wait_time
---------+-----------+-----------+--------------
       1 |  16777219 |        11 |        12864
       1 |  16777220 |       501 |       174473
       1 |  16777270 |     53171 |     23641847
       1 |  16777271 |        23 |       319668
       1 |  16777274 |        60 |        12759
.
.
.
      10 | 167772231 |    204596 |    790945212
      10 | 167772232 |         2 |        47729
      10 | 167772234 |         1 |          888
      10 | 167772235 |         2 |           64
```

En el siguiente ejemplo se muestra cómo se puede usar esta función junto con `aurora_stat_wait_event` y `aurora_stat_wait_type` para producir resultados más legibles.

```
=> SELECT type_name,
            event_name,
            waits,
            wait_time
        FROM aurora_stat_system_waits()
NATURAL JOIN aurora_stat_wait_event()
NATURAL JOIN aurora_stat_wait_type();

 type_name |       event_name       |  waits   |  wait_time
-----------+------------------------+----------+--------------
 LWLock    | XidGenLock             |       11 |        12864
 LWLock    | ProcArrayLock          |      501 |       174473
 LWLock    | buffer_content         |    53171 |     23641847
 LWLock    | rdsutils               |        2 |        12764
 Lock      | tuple                  |    75686 |   2033956052
 Lock      | transactionid          |  1765147 |  47267583409
 Activity  | AutoVacuumMain         |   136868 |  56305604538
 Activity  | BgWriterHibernate      |     7486 |  55266949471
```

```
 Activity  | BgWriterMain           |     7487 |   1508909964
.
.
.
 IO        | SLRURead               |        3 |        11756
 IO        | WALWrite               | 52544463 |    388850428
 IO        | XactSync               |   187073 |    597041642
 IO        | ClogRead               |        2 |        47729
 IO        | OutboundCtrlRead       |        1 |          888
 IO        | OutboundCtrlWrite      |        2 |           64
```

aurora_stat_wait_event

Enumera todos los eventos de espera admitidos para Aurora PostgreSQL. Para obtener información acerca de los eventos de espera de Aurora PostgreSQL, consulte Eventos de espera de Amazon Aurora PostgreSQL.

Sintaxis

```
aurora_stat_wait_event()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- type_id: ID del tipo de evento de espera
- event_id: ID del evento de espera
- type_name: nombre de tipo de espera
- event_name: nombre de evento de espera

Notas de uso

Para ver nombres de eventos con tipos de eventos en lugar de los ID, utilice esta función junto con otras funciones como `aurora_stat_wait_type` y `aurora_stat_system_waits`. Los nombres de los eventos de espera devueltos por esta función son los mismos que los devueltos por la función `aurora_wait_report`.

Ejemplos

En el ejemplo siguiente se muestran los resultados de llamar a la función `aurora_stat_wait_event`.

```
=>  SELECT *
    FROM aurora_stat_wait_event();

 type_id | event_id  |               event_name
---------+-----------+-----------------------------------------
       1 |  16777216 | <unassigned:0>
       1 |  16777217 | ShmemIndexLock
       1 |  16777218 | OidGenLock
       1 |  16777219 | XidGenLock
.
.
.
       9 | 150994945 | PgSleep
       9 | 150994946 | RecoveryApplyDelay
      10 | 167772160 | BufFileRead
      10 | 167772161 | BufFileWrite
      10 | 167772162 | ControlFileRead
.
.
.
      10 | 167772226 | WALInitWrite
      10 | 167772227 | WALRead
      10 | 167772228 | WALSync
      10 | 167772229 | WALSyncMethodAssign
      10 | 167772230 | WALWrite
      10 | 167772231 | XactSync
.
.
.
      11 | 184549377 | LsnAllocate
```

En el siguiente ejemplo se une `aurora_stat_wait_type` y `aurora_stat_wait_event` para devolver nombres de tipo y nombres de eventos para mejorar la legibilidad.

```
=> SELECT *
    FROM aurora_stat_wait_type() t
    JOIN aurora_stat_wait_event() e
      ON t.type_id = e.type_id;
```

```
 type_id | type_name | type_id | event_id  |                event_name
---------+-----------+---------+-----------+-------------------------------------------
       1 | LWLock    |       1 |  16777216 | <unassigned:0>
       1 | LWLock    |       1 |  16777217 | ShmemIndexLock
       1 | LWLock    |       1 |  16777218 | OidGenLock
       1 | LWLock    |       1 |  16777219 | XidGenLock
       1 | LWLock    |       1 |  16777220 | ProcArrayLock
.
.
.
       3 | Lock      |       3 |  50331648 | relation
       3 | Lock      |       3 |  50331649 | extend
       3 | Lock      |       3 |  50331650 | page
       3 | Lock      |       3 |  50331651 | tuple
.
.
.
      10 | IO        |      10 | 167772214 | TimelineHistorySync
      10 | IO        |      10 | 167772215 | TimelineHistoryWrite
      10 | IO        |      10 | 167772216 | TwophaseFileRead
      10 | IO        |      10 | 167772217 | TwophaseFileSync
.
.
.
      11 | LSN       |      11 | 184549376 | LsnDurable
```

aurora_stat_wait_type

Muestra todos los tipos de espera admitidos para Aurora PostgreSQL.

Sintaxis

```
aurora_stat_wait_type()
```

Argumentos

Ninguno

Tipo de retorno

Registro SETOF con las siguientes columnas:

- type_id: ID del tipo de evento de espera
- type_name: nombre de tipo de espera

Notas de uso

Para ver nombres de eventos de espera con tipos de eventos de espera en lugar de los ID, utilice esta función junto con otras funciones como `aurora_stat_wait_event` y `aurora_stat_system_waits`. Los nombres de tipo de espera devueltos por esta función son los mismos que los devueltos por la función `aurora_wait_report`.

Ejemplos

En el ejemplo siguiente se muestran los resultados de llamar a la función `aurora_stat_wait_type`.

```
=> SELECT *
     FROM aurora_stat_wait_type();
 type_id | type_name
---------+-----------
       1 | LWLock
       3 | Lock
       4 | BufferPin
       5 | Activity
       6 | Client
       7 | Extension
       8 | IPC
       9 | Timeout
      10 | IO
      11 | LSN
```

aurora_version

Devuelve el valor de cadena del número de versión de la edición compatible con Amazon Aurora PostgreSQL.

Sintaxis

```
aurora_version()
```

Argumentos

Ninguno

Tipo de retorno

Cadena CHAR o VARCHAR

Notas de uso

Esta función muestra la versión del motor de base de datos de la edición compatible con Amazon Aurora PostgreSQL. El número de versión se devuelve como una cadena con el formato *major*.*minor*.*patch*. Para obtener más información acerca de los números de versión de Aurora PostgreSQL, consulte Número de versión de Aurora.

Puede elegir cuándo aplicar actualizaciones de versiones secundarias configurando el período de mantenimiento para el clúster de base de datos de Aurora PostgreSQL. Para aprender a hacerlo, consulte Mantenimiento de un clúster de base de datos de Amazon Aurora.

A partir del lanzamiento de las versiones 13.3, 12.8, 11.13 y 10.18 de Aurora PostgreSQL, y para todas las demás versiones posteriores, los números de versión de Aurora se ajustan a los números de versión de PostgreSQL. Para obtener más información acerca de todas las versiones de Aurora PostgreSQL, consulte el tema sobre actualizaciones de Amazon Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

Ejemplos

En el siguiente ejemplo se muestran los resultados de la llamada a la función `aurora_version` en un clúster de base de datos de Aurora PostgreSQL que ejecuta la versión 12.7 de PostgreSQL, 4.2 de Aurora PostgreSQL y, a continuación, la misma función en un clúster que ejecuta la versión 13.3 de Aurora PostgreSQL.

```
=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
```

En este ejemplo se muestra cómo utilizar la función con varias opciones para obtener más detalles sobre la versión de Aurora PostgreSQL. En este ejemplo se incluye un número de versión de Aurora distinto del número de versión de PostgreSQL.

```
=> SHOW SERVER_VERSION;
 server_version
----------------
 12.7
(1 row)

=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
(1 row)

=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 12.7
(1 row)

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------
 PostgreSQL 12.7 on aarch64-unknown-linux-gnu, compiled by aarch64-unknown-linux-gnu-
gcc (GCC) 7.4.0, 64-bit
(1 row)

=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------
 Aurora: 4.2.2 Compatible with PostgreSQL: 12.7
(1 row)
```

En el siguiente ejemplo se utiliza la función con las mismas opciones del ejemplo anterior. En este ejemplo se incluye un número de versión de Aurora distinto del número de versión de PostgreSQL.

```
=> SHOW SERVER_VERSION;
server_version
----------------
 13.3

=> SELECT * FROM aurora_version();
aurora_version
---------------
 13.3.0
=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 13.3

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------------
 Aurora: 13.3.0 Compatible with PostgreSQL: 13.3
```

aurora_volume_logical_start_lsn

Devuelve el número de secuencia de registro (LSN, por sus siglas en inglés) utilizado para identificar el principio de un registro en la transmisión lógica de registro anticipado (WAL, por sus siglas en inglés) del volumen del clúster de Aurora.

Sintaxis

```
aurora_volume_logical_start_lsn()
```

Argumentos

Ninguno

Tipo de retorno

`pg_lsn`

Notas de uso

Esta función identifica el principio del registro en el flujo WAL lógico para un volumen de clúster de Aurora determinado. Puede utilizar esta función al realizar una actualización de la versión principal mediante la replicación lógica y la clonación rápida de Aurora para determinar el LSN en el que se toma una instantánea o un clon de una base de datos. A continuación, puede utilizar la replicación lógica para transmitir de forma continua los datos más recientes registrados después del LSN y sincronizar los cambios del publicador al suscriptor.

Para obtener más información sobre el uso de la replicación lógica para una actualización de versión principal, consulte Uso de la replicación lógica para realizar una actualización de la versión principal para Aurora PostgreSQL.

Esta función está disponible para las siguientes versiones de Aurora PostgreSQL:

- Versión 15.2 y versiones posteriores a la 15
- Versión 14.3 y versiones posteriores a la 14
- Versión 13.6 y versiones posteriores a la 13
- Versión 12.10 y versiones posteriores a la 12
- Versión 11.15 y versiones posteriores a la 11
- Versión 10.20 y versiones posteriores a la 10

Ejemplos

Puede obtener el número de secuencia de registro (LSN) mediante la siguiente consulta:

```
postgres=> SELECT aurora_volume_logical_start_lsn();

aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

aurora_wait_report

Esta función muestra la actividad del evento de espera durante un período de tiempo.

Sintaxis

```
aurora_wait_report([time])
```

Argumentos

time (opcional)

Tiempo en segundos. El valor predeterminado es de 10 segundos.

Tipo de retorno

Registro SETOF con las siguientes columnas:

- type_name: nombre de tipo de espera
- event_name: nombre de evento de espera
- wait: número de esperas
- wait_time: tiempo de espera en milisegundos
- ms_per_wait: promedio de milisegundos por el número de una espera
- waits_per_xact: promedio de esperas por el número de una transacción
- ms_per_wait: promedio de milisegundos por el número de transacciones

Notas de uso

Esta función está disponible a partir de la versión 1.1 de Aurora PostgreSQL compatible con PostgreSQL 9.6.6 y versiones superiores.

Para utilizar esta función, primero debe crear la extensión `aurora_stat_utils` de Aurora PostgreSQL, según se indica:

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Para obtener más información acerca de las versiones disponibles de la extensión de Aurora PostgreSQL, consulte el tema sobre versiones de extensiones para Amazon Aurora PostgreSQL en las Notas de la versión de Aurora PostgreSQL.

Esta función calcula los eventos de espera en el nivel de instancia comparando dos instantáneas de datos estadísticos de la función aurora_stat_system_waits() y las vistas de estadísticas de PostgreSQL pg_stat_database.

Para obtener más información sobre `aurora_stat_system_waits()` y `pg_stat_database`, consulte el tema sobre el recopilador de estadísticas en la documentación de PostgreSQL.

Cuando se ejecuta, esta función toma una instantánea inicial, espera el número de segundos especificado y, a continuación, toma una segunda instantánea. La función compara las dos instantáneas y devuelve la diferencia. Esta diferencia representa la actividad de la instancia durante ese intervalo de tiempo.

En la instancia de escritor, la función también muestra el número de transacciones confirmadas y TPS (transacciones por segundo). Esta función devuelve información en el nivel de instancia e incluye todas las bases de datos de la instancia.

Ejemplos

En este ejemplo se muestra cómo crear la extensión aurora_stat_utils para poder utilizar la función aurora_wait_report.

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

En este ejemplo se muestra cómo comprobar el informe de espera durante 10 segundos.

```
=> SELECT *
     FROM aurora_wait_report();
NOTICE:  committed 34 transactions in 10 seconds (tps 3)
 type_name |    event_name     | waits | wait_time | ms_per_wait | waits_per_xact |
 ms_per_xact
-----------+-------------------+-------+-----------+-------------+----------------
+-------------
 Client    | ClientRead        |    26 |  30003.00 |    1153.961 |           0.76 |
 882.441
 Activity  | WalWriterMain     |    50 |  10051.32 |     201.026 |           1.47 |
 295.627
 Timeout   | PgSleep           |     1 |  10049.52 |   10049.516 |           0.03 |
 295.574
 Activity  | BgWriterHibernate |     1 |  10048.15 |   10048.153 |           0.03 |
 295.534
```

```
 Activity  | AutoVacuumMain    |    18 |   9941.66 |     552.314 |          0.53 |
 292.402
 Activity  | BgWriterMain      |     1 |    201.09 |     201.085 |          0.03 |
   5.914
 IO        | XactSync          |    15 |     25.34 |       1.690 |          0.44 |
   0.745
 IO        | RelationMapRead   |    12 |      0.54 |       0.045 |          0.35 |
   0.016
 IO        | WALWrite          |    84 |      0.21 |       0.002 |          2.47 |
   0.006
 IO        | DataFileExtend    |     1 |      0.02 |       0.018 |          0.03 |
   0.001
```

En este ejemplo se muestra cómo comprobar el informe de espera durante 60 segundos.

```
=> SELECT *
     FROM aurora_wait_report(60);
NOTICE:  committed 1544 transactions in 60 seconds (tps 25)
 type_name |       event_name       |  waits  | wait_time | ms_per_wait |
 waits_per_xact | ms_per_xact
-----------+------------------------+---------+-----------+-------------
+----------------+-------------
 Lock      | transactionid          |    6422 | 477000.53 |      74.276 |
 4.16 |     308.938
 Client    | ClientRead             |    8265 | 270752.99 |      32.759 |
 5.35 |     175.358
 Activity  | CheckpointerMain       |       1 |  60100.25 |   60100.246 |
 0.00 |      38.925
 Timeout   | PgSleep                |       1 |  60098.49 |   60098.493 |
 0.00 |      38.924
 Activity  | WalWriterMain          |     296 |  60010.99 |     202.740 |
 0.19 |      38.867
 Activity  | AutoVacuumMain         |     107 |  59827.84 |     559.139 |
 0.07 |      38.749
 Activity  | BgWriterMain           |     290 |  58821.83 |     202.834 |
 0.19 |      38.097
 IO        | XactSync               |    1295 |  55220.13 |      42.641 |
 0.84 |      35.764
 IO        | WALWrite               | 6602259 |  47810.94 |       0.007 |
 4276.07 |      30.966
 Lock      | tuple                  |     473 |  29880.67 |      63.173 |
 0.31 |      19.353
```

```
LWLock    | buffer_mapping         |     142 |   3540.13 |      24.930 |
0.09 |       2.293
Activity  | BgWriterHibernate      |     290 |   1124.15 |       3.876 |
0.19 |       0.728
IO        | BufFileRead            |    7615 |    618.45 |       0.081 |
4.93 |       0.401
LWLock    | buffer_content         |      73 |    345.93 |       4.739 |
0.05 |       0.224
LWLock    | lock_manager           |      62 |    191.44 |       3.088 |
0.04 |       0.124
IO        | RelationMapRead        |      72 |      5.16 |       0.072 |
0.05 |       0.003
LWLock    | ProcArrayLock          |       1 |      2.01 |       2.008 |
0.00 |       0.001
IO        | ControlFileWriteUpdate |       2 |      0.03 |       0.013 |
0.00 |       0.000
IO        | DataFileExtend         |       1 |      0.02 |       0.018 |
0.00 |       0.000
IO        | ControlFileSyncUpdate  |       1 |      0.00 |       0.000 |
0.00 |       0.000
```

Parámetros de Amazon Aurora PostgreSQL.

El clúster de base de datos de Amazon Aurora se administra de la misma manera que las instancias de base de datos de Amazon RDS, mediante el uso de parámetros en un grupo de parámetros de base de datos. Sin embargo, Amazon Aurora se diferencia de Amazon RDS en que un clúster de base de datos de Aurora tiene varias instancias de base de datos. Algunos de los parámetros que se usan para administrar el clúster de base de datos de Amazon Aurora se aplican a todo el clúster, mientras que otros parámetros se aplican solo a una instancia de base de datos determinada en el clúster de base de datos, como se indica a continuación:

- Grupo de parámetros del clúster de base de datos: un grupo de parámetros del clúster de base de datos contiene el conjunto de parámetros de configuración del motor que se aplican en todo el clúster de base de datos de Aurora. Por ejemplo, la administración de la caché del clúster es una característica de un clúster de base de datos de Aurora que se controla mediante el parámetro `apg_ccm_enabled`, que forma parte del grupo de parámetros del clúster de base de datos. El grupo de parámetros del clúster de base de datos también contiene la configuración predeterminada del grupo de parámetros de base de datos para las instancias de base de datos que componen el clúster.

- Grupo de parámetros de base de datos: un grupo de parámetros de base de datos es el conjunto de valores de configuración del motor que se aplican a una instancia de base de datos específica de ese tipo de motor. Los grupos de parámetros de base de datos del motor de base de datos de PostgreSQL se usan en una instancia de base de datos de RDS para PostgreSQL y en el clúster de base de datos de Aurora PostgreSQL. Estos ajustes de configuración se aplican a propiedades que pueden variar entre las instancias de base de datos dentro de un clúster de Aurora, como los tamaños de los búferes de memoria.

El usuario administra los parámetros de nivel de clúster de los grupos de parámetros de clúster de base de datos. El usuario administra los parámetros de nivel de instancia de los grupos de parámetros de base de datos. Puede administrar los parámetros con la consola de Amazon RDS, la AWS CLI o la API de Amazon RDS. Hay comandos independientes para administrar los parámetros de nivel de clúster y los parámetros de nivel de instancia.

- Para administrar parámetros de nivel de clúster en un grupo de parámetros en un clúster de base de datos, utilice el comando modify-db-cluster-parameter-group de la AWS CLI.
- Para administrar parámetros de nivel de instancia en un grupo de parámetros de base de datos para una instancia de base de datos del clúster de base de datos, utilice el comando modify-db-parameter-group de la AWS CLI.

Para obtener más información sobre la AWS CLI, consulte la sección Using the AWS CLI (Uso de la AWS CLI) en la Guía del usuario de la AWS Command Line Interface.

Para obtener más información acerca de los grupos de parámetros, consulte Grupos de parámetros para Amazon Aurora.

Visualización de los parámetros del clúster y de la base de datos de Aurora PostgreSQL

Puede ver todos los grupos de parámetros predeterminados disponibles para las instancias de RDS para instancias de base de datos de PostgreSQL y para los clústeres de base de datos de Aurora PostgreSQL en la Consola de administración de AWS. Los grupos de parámetros predeterminados para todos los motores de base de datos y tipos y versiones de clústeres de base de datos se enumeran para cada región de AWS. También se enumeran los grupos de parámetros personalizados.

En lugar de verlo en la Consola de administración de AWS, también puede enumerar los parámetros contenidos en los grupos de parámetros de clústeres de bases de datos y grupos de parámetros de bases de datos utilizando la AWS CLI o la API de Amazon RDS. Por ejemplo, para enumerar los parámetros de un grupo de parámetros de un clúster de base de datos, utilice el comando de la AWS CLI describe-db-cluster-parameters como se indica a continuación:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12
```

El comando muestra descripciones JSON detalladas de cada parámetro. Para reducir la cantidad de información devuelta, puede especificar lo que desea con la opción `--query`. Por ejemplo, puede obtener el nombre del parámetro, su descripción y los valores permitidos del grupo de parámetros del clúster de base de datos de Aurora PostgreSQL 12 predeterminado de la siguiente manera:

Para Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Para Windows:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Un grupo de parámetros de un clúster de base de datos de Aurora incluye el grupo de parámetros de instancia de base de datos y los valores predeterminados de un motor de base de datos de Aurora determinado. Puede obtener la lista de parámetros de base de datos del mismo grupo de parámetros predeterminados de Aurora PostgreSQL con el comando AWS CLI describe-db-parameters como se muestra a continuación.

Para Linux, macOS o Unix:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Para Windows:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Los comandos anteriores muestran listas de parámetros del grupo de parámetros de base de datos o del clúster de base de datos con descripciones y otros detalles especificados en la consulta. A continuación, se muestra un ejemplo de respuesta.

```
[
    [
        {
            "ParameterName": "apg_enable_batch_mode_function_execution",
            "ApplyType": "dynamic",
            "Description": "Enables batch-mode functions to process sets of rows at a
 time.",
            "AllowedValues": "0,1"
        }
    ],
    [
        {
            "ParameterName": "apg_enable_correlated_any_transform",
            "ApplyType": "dynamic",
            "Description": "Enables the planner to transform correlated ANY Sublink
 (IN/NOT IN subquery) to JOIN when possible.",
            "AllowedValues": "0,1"
        }
    ],...
```

A continuación se muestran las tablas que contienen los valores del parámetro del clúster de base de datos y del parámetro de la base de datos predeterminados de la versión 14 de Aurora PostgreSQL.

Parámetros de nivel de clúster de Aurora PostgreSQL

Puede ver los parámetros de nivel de clúster disponibles para una versión de Aurora PostgreSQL específica usando la consola de administración de AWS, la CLI de AWS o la API de Amazon RDS. Para obtener información sobre cómo ver los parámetros en grupos de parámetros del clúster de base de datos de Aurora PostgreSQL en la consola de RDS, consulte Visualización de los valores de parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Algunos parámetros en el nivel de clúster no están disponibles en todas las versiones y otros están en desuso. Para obtener información sobre cómo ver los parámetros de una versión específica de Aurora PostgreSQL, consulte Visualización de los parámetros del clúster y de la base de datos de Aurora PostgreSQL.

Por ejemplo, la siguiente tabla muestra los parámetros disponibles en el grupo de parámetros del clúster de base de datos predeterminado de la versión 14 de Aurora PostgreSQL. Si crea un clúster de base de datos de Aurora PostgreSQL sin especificar su propio grupo de parámetros de base de datos personalizado, el clúster de base de datos se crea con el grupo de parámetros de clúster de base de datos de Aurora predeterminado de la versión elegida, como `default.aurora-postgresql14`, `default.aurora-postgresql13`, etc.

Para obtener una lista de los parámetros de instancia de base de datos del mismo grupo de parámetros de base de datos predeterminado, consulte Parámetros de nivel de instancia de Aurora PostgreSQL.

Nombre del parámetro	Descripción	Predeterminado
ansi_constraint_trigger_ordering	Cambia el orden de disparo de los desencadenadores de restricciones para que sean compatibles con el estándar ANSI SQL.	–
ansi_force_foreign_key_checks	Garantiza que se produzcan siempre acciones referenciales, como la eliminación en cascada o la actualización en cascada, independientemente de los distintos contextos de desencadenamiento que existan para la acción.	–

Nombre del parámetro	Descripción	Predeterminado
ansi_qualified_update_set_target	Admite los calificadores de tabla y de esquema en las instrucciones UPDATE… Instrucciones SET.	–
apg_ccm_enabled	Habilita o desactiva la administración de la caché del clúster.	–
apg_enable_batch_mode_function_execution	Habilita las funciones en modo lote para procesar conjuntos de filas a la vez.	–
apg_enable_correlated_any_transform	Permite que el planificador transforme la subconsulta ANY correlacionada (IN/NOT IN) en JOIN cuando sea posible.	–
apg_enable_function_migration	Permite al planificador migrar las funciones escalares elegibles a la cláusula FROM.	–
apg_enable_not_in_transform	Permite al planificador transformar la subconsulta NOT IN en ANTI JOIN cuando sea posible.	–
apg_enable_remove_redundant_inner_joins	Permite al planificador eliminar las uniones internas redundantes.	–
apg_enable_semijoin_push_down	Permite el uso de filtros semijoin para las uniones hash.	–
apg_plan_mgmt.capture_plan_baselines	Modo de línea base del plan de captura. manual - habilitar la captura del plan para cualquier sentencia SQL, desactivado - deshabilitar la captura del plan, automático - habilitar la captura del plan para las instrucciones en pg_stat_statements que cumplen los criterios de elegibilidad.	off

Nombre del parámetro	Descripción	Predeterminado
apg_plan_mgmt.max_databases	Establece el número máximo de bases de datos que se pueden administrar con apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Establece el número máximo de planes que se pueden almacenar en caché mediante apg_plan_mgmt.	10000
apg_plan_mgmt.plan_retention_period	Número máximo de días desde que un plan se usó por última vez antes de que un plan se elimine automáticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Costo total estimado del plan por debajo del cual se ejecutará un plan no aprobado.	0
apg_plan_mgmt.use_plan_baselines	Use solo planes aprobados o fijos para las instrucciones administradas.	false
application_name	Define el nombre de la aplicación sobre la que informarán las estadísticas y los registros.	–
array_nulls	Permite la entrada de elementos NULL en matrices.	–
aurora_compute_plan_id	Supervisa los planes de ejecución de consultas para detectar los planes de ejecución que contribuyen a la carga actual de la base de datos y para hacer un seguimiento de las estadísticas de rendimiento de los planes de ejecución a lo largo del tiempo. Para obtener más información, consulte Monitorización de planes de ejecución de consultas para Aurora PostgreSQL.	on

Nombre del parámetro	Descripción	Predeterminado
authentication_timeout	Establece el tiempo máximo permitido para completar una autenticación del cliente.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE para el registro de planes.	–
auto_explain.log_buffers	Usa los búferes de registro.	–
auto_explain.log_format	Usa el formato EXPLAIN para el registro del plan.	–
auto_explain.log_min_duration	Establece el tiempo mínimo de ejecución por encima del cual se registrarán los planes.	–
auto_explain.log_nested_statements	Registra las instrucciones anidadas.	–
auto_explain.log_timing	Recopila datos de cronometraje, no solo recuentos de filas.	–
auto_explain.log_triggers	Incluye estadísticas de desencadenado en los planes.	–
auto_explain.log_verbose	Usa EXPLAIN VERBOSE para el registro de planes.	–
auto_explain.sample_rate	Permite procesar una fracción de las consultas.	–
autovacuum	Inicia el subproceso de autovacuum.	–
autovacuum_analyze_scale_factor	Número de inserciones de tuplas, actualizaciones o borrados antes del análisis como fracción de retuplas.	0.05

Nombre del parámetro	Descripción	Predeterminado
autovacuum_analyze_threshold	Número mínimo de inserciones, actualizaciones o eliminaciones de tuplas antes del análisis.	–
autovacuum_freeze_max_age	Antigüedad con la que se debe aplicar autovacuum a una tabla para impedir el reinicio de los ID.	–
autovacuum_max_workers	Define el número máximo de procesos de empleados de autovacuum que se ejecutan simultáneamente.	GREATEST(DBInstanceClassMemory/64371566592,3)
autovacuum_multixact_freeze_max_age	Antigüedad de multiexact con la que se autovacía una tabla para evitar el reinicio de multiexact.	–
autovacuum_naptime	(s) Tiempo de espera entre ejecuciones de autovacuum.	5
autovacuum_vacuum_cost_delay	(ms) Retardo del costo del vacío en milisegundos, para autovacuum.	5
autovacuum_vacuum_cost_limit	Importe del costo del vacío disponible antes del periodo de reposo para autovacuum.	GREATEST(log(DBInstanceClassMemory/21474836480)*600,200)
autovacuum_vacuum_insert_scale_factor	Número de inserciones de tuplas antes del vacío como fracción de retuplas.	–
autovacuum_vacuum_insert_threshold	Número mínimo de inserciones de tuplas antes de vacuum o -1 para desactivar los vacíos de inserción.	–

Nombre del parámetro	Descripción	Predeterminado
autovacuum_vacuum_scale_factor	Número de actualizaciones o borrados de tuplas antes del vacío como fracción de reltuples.	0.1
autovacuum_vacuum_threshold	Número mínimo de actualizaciones o borrados de tuplas antes del vacío.	–
autovacuum_work_mem	(kB) Establece la memoria máxima que puede utilizar cada proceso de empleado de autovacuum.	GREATEST(DBInstanceClassMemory/32768,131072)
babelfishpg_tds.default_server_name	Nombre predeterminado del servidor de Babelfish	Microsoft SQL Server
babelfishpg_tds.listen_addresses	Establece el nombre de host o las direcciones IP para escuchar el TDS.	*
babelfishpg_tds.port	Establece el puerto TCP de TDS en el que escucha el servidor.	1433
babelfishpg_tds.tds_debug_log_level	Establece el nivel de registro en TDS, 0 desactiva el registro	1
babelfishpg_tds.tds_default_numeric_precision	Establece la precisión predeterminada del tipo numérico que se enviará en los metadatos de la columna TDS si el motor no especifica ninguna.	38
babelfishpg_tds.tds_default_numeric_scale	Establece la escala predeterminada del tipo numérico que se enviará en los metadatos de la columna TDS si el motor no especifica una.	8
babelfishpg_tds.tds_default_packet_size	Establece el tamaño de paquete predeterminado para todos los clientes de SQL Server que se conectan.	4096

Nombre del parámetro	Descripción	Predeterminado
babelfishpg_tds.tds_default_protocol_version	Establece la versión del protocolo TDS predeterminada para todos los clientes que se conectan	DEFAULT
babelfishpg_tds.tds_ssl_encrypt	Establece la opción de cifrado SSL	0
babelfishpg_tds.tds_ssl_max_protocol_version	Establece la versión máxima del protocolo SSL/TLS a usar para la sesión tds	TLSv1.2
babelfishpg_tds.tds_ssl_min_protocol_version	Establece la versión mínima del protocolo SSL/TLS que se puede usar para la sesión TDS.	TLSv1.2 de Aurora PostgreSQL versión 16, TLSv1 para versiones anteriores a Aurora PostgreSQL versión 16
babelfishpg_tsql.default_locale	Configuración regional predeterminada que se usará para las colaciones creadas por CREATE COLLATION.	en-US
babelfishpg_tsql.migration_mode	Define si se admiten múltiples bases de datos de usuarios	Varias bases de datos de Aurora PostgreSQL versión 16, una sola base de datos para versiones anteriores a Aurora PostgreSQL versión 16
babelfishpg_tsql.server_collation_name	Nombre de la intercalación de servidores predeterminada	sql_latin1_general_cp1_ci_as

Nombre del parámetro	Descripción	Predeterminado
babelfishpg_tsql.version	Establece la salida de la variable @@VERSION	predeterminado
backend_flush_after	(8Kb) Número de páginas después de las cuales las escrituras realizadas anteriormente se vacían al disco.	–
backslash_quote	Establece si se permite el uso de \\ en cadenas literales.	–
backtrace_functions	Backtrace de registro de errores en estas funciones.	–
bytea_output	Establece el formato de salida para bytea.	–
check_function_bodies	Comprueba los cuerpos de las funciones durante la ejecución de CREATE FUNCTION.	–
client_connection_check_interval	Establece el intervalo de tiempo entre las comprobaciones de desconexión mientras se ejecutan las consultas.	–
client_encoding	Define la codificación del conjunto de caracteres del cliente.	UTF8
client_min_messages	Establece los niveles de mensajes que se envían al cliente.	–
compute_query_id	Computa los identificadores de consultas.	auto
config_file	Establece el archivo de configuración principal del servidor.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Habilita el planificador para que use restricciones con el fin de optimizar las consultas.	–

Nombre del parámetro	Descripción	Predeterminado
cpu_index_tuple_cost	Establece la estimación del planificador del costo de procesar cada entrada de índice durante un examen del índice.	–
cpu_operator_cost	Establece la estimación del planificador del costo de procesar cada operador o llamada a una función.	–
cpu_tuple_cost	Establece la estimación del planificador del costo de procesar cada tupla (fila).	–
cron.database_name	Establece la base de datos para almacenar las tablas de metadatos de pg_cron	postgres
cron.log_run	Registra todas las ejecuciones de trabajos en la tabla job_run_details	on
cron.log_statement	Registra todas las instrucciones cron antes de la ejecución.	off
cron.max_running_jobs	Número máximo de trabajos que pueden ejecutarse simultáneamente.	5
cron.use_background_workers	Habilita empleados en segundo plano para pg_cron	on
cursor_tuple_fraction	Establece la estimación de los planificadores de la fracción de filas de un cursor que se recuperará.	–
data_directory	Establece el directorio de datos del servidor.	/rdsdbdata/db
datestyle	Define el formato de visualización para los valores de fecha y hora.	–

Nombre del parámetro	Descripción	Predeterminado
db_user_namespace	Habilita los nombres de usuario por base de datos.	–
deadlock_timeout	(ms) Establece el tiempo de espera de un bloqueo antes de comprobar si hay un bloqueo.	–
debug_pretty_print	Aplica una sangría a las visualizaciones del árbol de análisis y de planificación.	–
debug_print_parse	Registra el árbol de análisis de cada consulta.	–
debug_print_plan	Registra el plan de ejecución de cada consulta.	–
debug_print_rewritten	Registra el árbol de análisis reescrito de cada consulta.	–
default_statistics_target	Define el objetivo de estadística predeterminado.	–
default_tablespace	Define el espacio de tabla predeterminado en el que se deben crear las tablas y los índices.	–
default_toast_compression	Establece el método de compresión predeterminado para los valores comprimibles.	–
default_transaction_deferrable	Define el estado diferible predeterminado de las nuevas transacciones.	–
default_transaction_isolation	Define el nivel de aislamiento de cada nueva transacción.	–
default_transaction_read_only	Define el estado de solo lectura predeterminado de las nuevas transacciones.	–
efective_cache_size	(8kB) Establece la suposición de los planificadores sobre el tamaño de la caché del disco.	SUM(DBInstanceClassMemory/12038,-50003)

Nombre del parámetro	Descripción	Predeterminado
effective_io_concurrency	Número de solicitudes simultáneas que el subsistema del disco puede gestionar de un modo eficiente.	–
enable_async_append	Habilita el uso de planes de append asíncronos por parte de los planificadores.	–
enable_bitmapscan	Habilita el uso de planes de examen de mapas de bits por parte de los planificadores.	–
enable_gathermerge	Habilita el uso de planes de fusión de recopilación por parte de los planificadores.	–
enable_hashagg	Habilita el uso de planes de agregación con hash por parte de los planificadores.	–
enable_hashjoin	Habilita el uso de planes de combinación con hash por parte de los planificadores.	–
enable_incremental_sort	Habilita el uso de pasos de ordenación incrementales por parte de los planificadores.	–
enable_indexonlyscan	Habilita el uso de planes de examen solo de índice por parte de los planificadores.	–
enable_indexscan	Habilita el uso de planes de examen de índice por parte de los planificadores.	–
enable_material	Habilita el uso de materialización por parte de los planificadores.	–
enable_memoize	Habilita el uso de memorización para los planificadores	–
enable_mergejoin	Habilita el uso de planes de combinación por fusión por parte de los planificadores.	–

Nombre del parámetro	Descripción	Predeterminado
enable_nestloop	Habilita el uso de planes de combinación de bucles anidados por parte de los planificadores.	–
enable_parallel_append	Habilita el uso de planes de append en paralelo por parte de los planificadores.	–
enable_parallel_hash	Habilita el uso de planes hash paralelos por parte de los planificadores.	–
enable_partition_pruning	Habilita la depuración de particiones en tiempo de planificación y de ejecución.	–
enable_partitionwise_aggregate	Habilita la agregación y agrupación de particiones.	–
enable_partitionwise_join	Habilita la combinación de particiones.	–
enable_seqscan	Habilita el uso de planes de examen secuencial por parte de los planificadores.	–
enable_sort	Habilita el uso de pasos de ordenación explícitos por parte de los planificadores.	–
enable_tidscan	Habilita el uso de planes de examen TID por parte de los planificadores.	–
escape_string_warning	Advierte sobre los escapes de barra invertida en los literales de cadena ordinarios.	–
exit_on_error	Termina la sesión en caso de error.	–
extra_float_digits	Define el número de dígitos que se muestran para los valores de punto flotante.	–
force_parallel_mode	Fuerza el uso de las facilidades de consulta paralela.	–

Nombre del parámetro	Descripción	Predeterminado
from_collapse_limit	Define el tamaño de la lista FROM por encima del cual las subconsultas no se contraen.	–
geqo	Habilita la optimización de consultas geneticas.	–
geqo_effort	GEQO: esfuerzo que se usa para definir el valor predeterminado de otros parámetros de GEQO.	–
geqo_generations	GEQO: número de iteraciones del algoritmo.	–
geqo_pool_size	GEQO: número de individuos de la población.	–
geqo_seed	GEQO: valor de inicialización para la selección de ruta aleatoria.	–
geqo_selection_bias	GEQO: presión selectiva dentro de la población .	–
geqo_threshold	Define el umbral de los elementos FROM por encima de los cuales se usa GEQO.	–
gin_fuzzy_search_limit	Define el resultado máximo permitido para la búsqueda exacta por GIN.	–
gin_pending_list_limit	(kB) Establece el tamaño máximo de la lista pendiente para el índice de GIN.	–
hash_mem_multiplier	Múltiplo de work_mem que se utilizará para las tablas hash.	–
hba_file	Establece el archivo de configuración de HBA de servidores.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Permite la retroalimentación de una espera activa al primario que evitará conflictos de consulta.	on

Nombre del parámetro	Descripción	Predeterminado
huge_pages	Reduce la sobrecarga cuando una instancia de base de datos trabaja con grandes fragmentos contiguos de memoria, como los utilizados por los búferes compartidos. Se activa de forma predeterminada para todas las clases de instancia de base de datos que no sean las siguientes: t3.medium, db.t3.large, db.t4g.medium y db.t4g.large.	on
ident_file	Establece el archivo de configuración de ident de servidores.	/rdsdbdata/config/pg_hba.conf
idle_in_transaction_session_timeout	(ms) Establece la duración máxima permitida de cualquier transacción en espera.	86400000
idle_session_timeout	Finaliza cualquier sesión que haya estado inactiva (es decir, a la espera de una consulta por parte de un cliente), pero no dentro de una transacción abierta, durante más tiempo del especificado	–
intervalstyle	Define el formato de visualización para los valores de intervalo.	–
join_collapse_limit	Define el tamaño de la lista FROM por encima del cual las construcciones JOIN no se aplanan.	–

Nombre del parámetro	Descripción	Predeterminado
krb_caseins_users	Establece si los nombres de usuario de la GSSAPI (Generic Security Service API) deben distinguir entre mayúsculas y minúsculas (verdadero) o no. De forma predeterminada, este parámetro está configurado en falso, por lo que Kerberos espera que los nombres de usuario distingan entre mayúsculas y minúsculas. Para obtener más información, consulte GSSAPI Authentication (Autentic ación de GSSAPI) en la documentación de PostgreSQL.	false
lc_messages	Define el idioma en el que se muestran los mensajes.	–
lc_monetary	Define la configuración regional para el formato de las cantidades monetarias.	–
lc_numeric	Define la configuración regional para el formato de los números.	–
lc_time	Define la configuración regional para el formato de los valores de fecha y hora.	–
listen_addresses	Establece el nombre de host o las direcciones IP para escuchar.	*
lo_compat_privileges	Habilita el modo de compatibilidad con versiones pasadas para las comprobaciones de privilegios en objetos grandes.	0
log_autovacuum_min_duration	(ms) Establece el tiempo mínimo de ejecución a partir del cual se registrarán las acciones de autovacuum.	10000

Nombre del parámetro	Descripción	Predeterminado
log_connections	Registra cada conexión realizada correctamente.	–
log_destination	Establece el destino de la salida del registro del servidor.	stderr
log_directory	Establece el directorio de destino para los archivos de registro.	/rdsdbdata/log/error
log_disconnections	Registra el final de una sesión, incluida su duración.	–
log_duration	Registra la duración de cada instrucción de SQL completada.	–
log_error_verbosity	Define el detalle de los mensajes registrados.	–
log_executor_stats	Escribe las estadísticas de rendimiento del ejecutor en el registro del servidor.	–
log_file_mode	Establece los permisos de los archivos de registro.	0644
log_filename	Define el patrón del nombre de archivo de los archivos de registro.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Inicia un subproceso para capturar el resultado de stderr o csvlogs en archivos de registro.	1
log_hostname	Registra el nombre del host en los registros de conexión.	0
logical_decoding_work_mem	(kB) Cada búfer de reordenamiento interno puede usar esta cantidad de memoria antes de volcar en el disco.	–

Nombre del parámetro	Descripción	Predeterminado
log_line_prefix	Controla la información prefijada en cada línea de registro.	%t: %r: %u@%d: %p]:
log_lock_waits	Registra las esperas de bloqueo largas.	–
log_min_duration_sample	(ms) Establece el tiempo mínimo de ejecución a partir del cual se registrará una muestra de instrucciones. El muestreo se determina mediante log_statement_sample_rate.	–
log_min_duration_statement	(ms) Establece el tiempo mínimo de ejecución por encima del cual se registrarán las instrucciones.	–
log_min_error_statement	Hace que se registren todas las instrucciones que generen un error por encima de este nivel.	–
log_min_messages	Define los niveles de los mensajes que se registran.	–
log_parameter_max_length	(B) Al registrar instrucciones limita los valores de los parámetros registrados a los primeros N bytes.	–
log_parameter_max_length_on_error	(B) Cuando se informa de un error, limita los valores de los parámetros registrados a los primeros N bytes.	–
log_parser_stats	Escribe las estadísticas de rendimiento del analizador en el registro del servidor.	–
log_planner_stats	Escribe las estadísticas de rendimiento del planificador en el registro del servidor.	–
log_replication_commands	Registra cada comando de replicación.	–

Nombre del parámetro	Descripción	Predeterminado
log_rotation_age	(min) La rotación automática del archivo de registro se producirá después de N minutos.	60
log_rotation_size	(kB) La rotación automática del archivo de registro se producirá después de N kilobytes.	100000
log_statement	Define el tipo de declaraciones que se deben registrar.	–
log_statement_sample_rate	Fracción de instrucciones que exceden la muestra de log_min_duration que se registrará.	–
log_statement_stats	Escribe las estadísticas de rendimiento acumulativas en el registro del servidor.	–
log_temp_files	(kB) Registra el uso de archivos temporales mayores a este número de kilobytes.	–
log_timezone	Establece la zona horaria que se usará en los mensajes de registro.	UTC
log_transaction_sample_rate	Establece la fracción de transacciones que se tienen que registrar para las nuevas transacciones.	–
log_truncate_on_rotation	Permite truncar los archivos de registro existentes con el mismo nombre durante la rotación del registro.	0
maintenance_io_concurrency	Una variante de effective_io_concurrency que se utiliza para trabajos de mantenimiento.	1
maintenance_work_mem	(kB) Establece la memoria máxima que se puede usar para las operaciones de mantenimiento	GREATEST(DBInstanceClassMemory/63963136*1024, 65536)

Nombre del parámetro	Descripción	Predeterminado
max_connections	Define el número máximo de conexiones simultáneas.	LEAST(DBInstanceClassMemory/9531392, 5000)
max_files_per_process	Define el número máximo de archivos abiertos simultáneamente para cada proceso del servidor.	–
max_locks_per_transaction	Define el número máximo de bloqueos por transacción.	64
max_logical_replication_workers	Número máximo de procesos de empleados de replicación lógica.	–
max_parallel_maintenance_workers	Establece el número máximo de procesos en paralelo por operación de mantenimiento.	–
max_parallel_workers	Establece el número máximo de empleados en paralelo que pueden estar activos a la vez.	GREATEST($DBInstanceVCPU/2, 8)
max_parallel_workers_per_gather	Establece el número máximo de procesos en paralelo por nodo ejecutor.	–
max_pred_locks_per_page	Establece el número máximo de tuplas bloqueadas por predicado por página.	–
max_pred_locks_per_relation	Establece el número máximo de páginas y tuplas bloqueadas por predicado por relación.	–
max_pred_locks_per_transaction	Define el número máximo de bloqueos de predicado por transacción.	–
max_prepared_transactions	Define el número máximo de transacciones preparadas simultáneamente.	0

Nombre del parámetro	Descripción	Predeterminado
max_replication_slots	Establece el número máximo de ranuras de replicación que puede admitir el servidor.	20
max_slot_wal_keep_size	(MB) Las ranuras de replicación se marcarán como fallidas, y los segmentos se liberarán para su eliminación o reciclaje si esta cantidad de espacio está ocupada por la WAL en el disco.	–
max_stack_depth	(kB) Establece la profundidad máxima de la pila, en kilobytes.	6144
max_standby_streaming_delay	(ms) Establece el retardo máximo antes de cancelar las consultas cuando un servidor en espera activa está procesando datos de WAL en flujo.	14000
max_sync_workers_per_subscription	Número máximo de empleados de sincronización por suscripción	2
max_wal_senders	Establece el número máximo de procesos emisores de WAL que se ejecutan simultáneamente.	10
max_worker_processes	Establece el número máximo de procesos de empleados que se ejecutan simultáneamente.	GREATEST($DBInstanceVCPU*2, 8)
min_dynamic_shared_memory	(MB) Cantidad de memoria compartida dinámica reservada al inicio.	–
min_parallel_index_scan_size	(8kB) Establece la cantidad mínima de datos de índice para un examen en paralelo.	–
min_parallel_table_scan_size	(8kB) Establece la cantidad mínima de datos de tabla para un examen en paralelo.	–

Nombre del parámetro	Descripción	Predeterminado
old_snapshot_threshold	(min) Tiempo antes de que una instantánea se vuelva demasiado antigua para leer las páginas modificadas después de la captura de la instantánea.	–
orafce.nls_date_format	Emula el comportamiento de salida de fecha de Oracle.	–
orafce.timezone	Especifica la zona horaria utilizada para la función sysdate.	–
parallel_leader_participation	Controla si Gather y Gather Merge también ejecutan subplanes.	–
parallel_setup_cost	Establece la estimación de los planificadores del costo de iniciar procesos de empleados para la consulta paralela.	–
parallel_tuple_cost	Establece la estimación de los planificadores del costo de pasar cada tupla (fila) del empleado al backend maestro.	–
password_encryption	Codifica las contraseñas.	–
pgaudit.log	Especifica qué clases de instrucciones se registrarán mediante el registro de auditoría de sesión.	–
pgaudit.log_catalog	Especifica que el registro de sesión se debe habilitar en el caso de que todas las relaciones de una instrucción se encuentren en pg_catalog.	–
pgaudit.log_level	Especifica el nivel de registro que se usará para las entradas de registro.	–

Nombre del parámetro	Descripción	Predeterminado
pgaudit.log_parameter	Especifica que el registro de auditoría debe incluir los parámetros que se pasaron con la instrucción.	–
pgaudit.log_relation	Especifica si el registro de auditoría de sesión debe crear una entrada de registro separada para cada relación (TABLE, VIEW, etc.) referenciada en una instrucción SELECT o DML.	–
pgaudit.log_statement_once	Especifica si el registro incluirá el texto de la instrucción y los parámetros con la primera entrada de registro para una combinación de instrucción o subinstrucción o con cada entrada.	–
pgaudit.role	Especifica el rol maestro que se usará para el registro de auditoría de objetos.	–
pg_bigm.enable_recheck	Especifica si se realiza una recomprobación, que es un proceso interno de búsqueda de texto completo.	on
pg_bigm.gin_key_limit	Especifica el número máximo de bigramas de la palabra clave de búsqueda que se utilizará para la búsqueda de texto completo.	0
pg_bigm.last_update	Informa de la última fecha de actualización del módulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Especifica el umbral mínimo que usa la búsqueda de similitud.	0.3
pg_hint_plan.debug_print	Registra los resultados del análisis sintáctico de las pistas.	–

Nombre del parámetro	Descripción	Predeterminado
pg_hint_plan.enable_hint	Fuerza al planificador a usar los planes especificados en el comentario de la pista que precede a la consulta.	–
pg_hint_plan.enable_hint_table	Fuerza al planificador a no obtener la sugerencia mediante el uso de búsquedas en tablas.	–
pg_hint_plan.message_level	Nivel de mensaje de los mensajes de depuración.	–
pg_hint_plan.parse_messages	Nivel de mensaje de los errores de análisis.	–
pglogical.batch_inserts	Inserciones por lotes si es posible	–
pglogical.conflict_log_level	Establece el nivel de registro utilizado para registrar los conflictos resueltos.	–
pglogical.conflict_resolution	Establece el método que se usa para la resolución de conflictos de los conflictos resolubles.	–
pglogical.extra_connection_options	opciones de conexión para agregar a todas las conexiones de los nodos pares	–
pglogical.synchronous_commit	valor de confirmación sincrónica específico de pglogical	–
pglogical.use_spi	Usa SPI en lugar de la API de bajo nivel para aplicar los cambios	–
pgtle.clientauth_databases_to_skip	Lista de bases de datos que se deben omitir para utilizar la característica clientauth.	–

Nombre del parámetro	Descripción	Predeterminado
pgtle.clientauth_db_name	Controla qué base de datos se utiliza para la característica clientauth.	–
pgtle.clientauth_num_parallel_workers	Número de trabajadores en segundo plano utilizados para la característica clientauth.	–
pgtle.clientauth_users_to_skip	Lista de bases de datos que se deben omitir para la característica clientauth.	–
pgtle.enable_clientauth	Hablita la característica clientauth.	–
pgtle.passcheck_db_name	Establece qué base de datos se utiliza para característica passcheck en todo el clúster.	–
pg_prewarm.autoprewarm	Inicia el empleado de precalentamiento automático.	–
pg_prewarm.autoprewarm_interval	Establece el intervalo entre volcados de búferes compartidos	–
pg_similarity.block_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.block_threshold	Establece el umbral utilizado por la función de similitud Bloquear.	–
pg_similarity.block_tokenizer	Establece el tokenizador para la función de similitud de bloques.	–
pg_similarity.cosine_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.cosine_threshold	Establece el umbral utilizado por la función de similitud coseno.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.cosine_tokenizer	Establece el tokenizador para la función de similitud de coseno.	–
pg_similarity.dice_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.dice_threshold	Establece el umbral utilizado por la medida de similitud de dados.	–
pg_similarity.dice_tokenizer	Establece el tokenizador para la medida de similitud de dados.	–
pg_similarity.euclidean_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.euclidean_threshold	Establece el umbral utilizado por la medida de similitud euclidiana.	–
pg_similarity.euclidean_tokenizer	Establece el tokenizador para la medida de similitud euclidiana.	–
pg_similarity.hamming_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.hamming_threshold	Establece el umbral utilizado por la métrica de similitud de bloques.	–
pg_similarity.jaccard_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.jaccard_threshold	Establece el umbral utilizado por la medida de similitud Jaccard.	–
pg_similarity.jaccard_tokenizer	Establece el tokenizador para la medida de similitud de Jaccard.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.jaro_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.jaro_threshold	Establece el umbral utilizado por la medida de similitud Jaro.	–
pg_similarity.jarowinkler_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.jarowinkler_threshold	Establece el umbral utilizado por la medida de similitud Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.levenshtein_threshold	Establece el umbral utilizado por la medida de similitud Levenshtein.	–
pg_similarity.matching_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.matching_threshold	Establece el umbral utilizado por la medida de similitud de coeficiente de coincidencia.	–
pg_similarity.matching_tokenizer	Establece el tokenizador para la medida de similitud de coeficiente de coincidencia.	–
pg_similarity.mongeelkan_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.mongeelkan_threshold	Establece el umbral utilizado por la medida de similitud Monge-Elkan.	–
pg_similarity.mongeelkan_tokenizer	Establece el tokenizador para la medida de similitud Monge-Elkan.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.nw_gap_penalty	Establece la penalización de separación utilizada por la medida de similitud Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.nw_threshold	Establece el umbral utilizado por la medida de similitud Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.overlap_threshold	Establece el umbral utilizado por la medida de similitud de coeficiente de superposición.	–
pg_similarity.overlap_tokenizer	Establece el tokenizador para la medida de similitud del coeficiente de superposición.	–
pg_similarity.qgram_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.qgram_threshold	Establece el umbral utilizado por la medida de similitud Q-Gram.	–
pg_similarity.qgram_tokenizer	Establece el tokenizador para la medida Q-Gram.	–
pg_similarity.swg_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.swg_threshold	Establece el umbral utilizado por la medida de similitud Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Establece si el valor del resultado está normalizado o no.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.sw_threshold	Establece el umbral utilizado por la medida de similitud de Smith-Waterman.	–
pg_stat_statements.max	Establece el número máximo de instrucciones rastreadas por pg_stat_statements.	–
pg_stat_statements.save	Guarda las estadísticas de pg_stat_statements a través de los apagados del servidor.	–
pg_stat_statements.track	Selecciona qué instrucciones rastrea pg_stat_statements.	–
pg_stat_statements.track_planning	Selecciona si la duración de la planificación se rastrea con pg_stat_statements.	–
pg_stat_statements.track_utility	Selecciona si los comandos de utilidad se rastrean mediante pg_stat_statements.	–
plan_cache_mode	Controla la selección del planificador del plan personalizado o genérico.	–
puerto	Establece el puerto TCP en el que escucha el servidor.	EndPointPort
postgis.gdal_enabled_drivers	Habilita o desactiva los controladores GDAL que se usan con PostGIS en Postgres 9.3.5 y versiones posteriores.	ENABLE_ALL
quote_all_identifiers	Al generar fragmentos de SQL, cite todos los identificadores.	–
random_page_cost	Establece la estimación de los planificadores del costo de una página de disco que no se recupera secuencialmente.	–

Nombre del parámetro	Descripción	Predeterminado
rdkit.dice_threshold	Umbral inferior de similitud de Dice. Las moléculas con una similitud inferior al umbral no son similares según la operación #.	–
rdkit.do_chiral_sss	En caso de que se tenga en cuenta la estereoquímica en la coincidencia de subestructuras. Si es falso, no se usa ninguna información de estereoquímica en las coincidencias de subestructuras.	–
rdkit.tanimoto_threshold	Umbral inferior de similitud de Tanimoto. Las moléculas con una similitud inferior al umbral no son similares según la operación %.	–
rds.accepted_password_auth_method	Fuerza la autenticación para las conexiones con contraseñas almacenadas localmente.	md5+scram
rds.adaptive_autovacuum	Parámetro RDS para habilitar o desactivar el autovacuum adaptativo.	1
rds.babelfish_status	Parámetro de RDS para habilitar o desactivar Babelfish for Aurora PostgreSQL.	off
rds.enable_plan_management	Habilita o desactiva la extensión apg_plan_mgmt.	0

Nombre del parámetro	Descripción	Predeterminado
rds.extensions	Lista de extensiones proporcionadas por RDS	address_standardizer, address_standardizer_data_us, apg_plan_mgmt, aurora_stat_utils, amcheck, autoinc, aws_commons, aws_ml, aws_s3, aws_lambda, bool_plperl, bloom, btree_gin, btree_gist, citext, cube, dblink, dict_int, dict_xsyn, earthdistance, fuzzystrmatch, hll, hstore, hstore_plperl, insert_username, intagg, intarray, ip4r, isn, jsonb_plperl, lo, log_fdw, ltree, moddatetime, old_snapshot, oracle_fdw, orafce, pgaudit, pgcrypto, pglogical, pgrouting, pgrowlocks, pgstattuple, pgtap, pg_bigm, pg_buffercache, pg_cron, pg_freespacemap, pg_hint_plan, pg_partman, pg_prewarm, pg_proctab,

Nombre del parámetro	Descripción	Predeterminado
		pg_repack, pg_similarity, pg_stat_statements, pg_trgm, pg_visibility, plcoffee, plls, plperl, plpgsql, plprofiler, pltcl, plv8, postgis, postgis_tiger_geocoder, postgis_raster, postgis_topology, postgres_fdw, prefix, rdkit, rds_tools, refint, sslinfo, tablefunc, tds_fdw, test_parser, tsm_system_rows, tsm_system_time, unaccent, uuid-ossp
rds.force_admin_logging_level	Consulta los mensajes de registro de las acciones del usuario administrador de RDS en las bases de datos de los clientes.	–
rds.force_autovacuum_logging_level	Consulta los mensajes de registro relacionados con las operaciones de autovacuum.	ADVERTENCIA
rds.force_ssl	Fuerza las conexiones SSL.	0

Nombre del parámetro	Descripción	Predeterminado
show rds.global_db_rpo;	(s) Umbral objetivo del punto de recuperación, en segundos, que bloquea las confirmaciones del usuario cuando se infringe. **Important** Este parámetro está destinado a bases de datos globales basadas en Aurora PostgreSQL. Para una base de datos no global, déjelo en el valor predeterminado. Para obtener más información sobre el uso de este parámetro, consulte the section called “Administración de RPO (Aurora PostgreSQL)”.	–
rds.logical_replication	Habilita la decodificación lógica.	0
rds.logically_replicate_unlogged_tables	Las tablas no registradas se replican de forma lógica.	1
rds.log_retention_period	Amazon RDS eliminará los registros de PostgreSQL que tengan una antigüedad superior a N minutos.	4320
rds.pg_stat_ramdisk_size	Tamaño del disco de estadísticas en MB. Un valor distinto de cero configurará el disco ramdisk. Este parámetro solo está disponible en la versión 14 e inferiores de Aurora PostgreSQL.	0

Nombre del parámetro	Descripción	Predeterminado
rds.rds_superuser_reserved_connections	Establece el número de ranuras de conexión reservadas para rds_superusers. Este parámetro solo está disponible en la versión 15 y anteriores. Para obtener más información, consulte la documentación de PostgreSQL sobre reserved_connections.	2
rds.restrict_password_commands	Restringe los comandos relacionados con la contraseña a los miembros de rds_password	–
rds.superuser_variables	Lista de variables solo para superusuarios para las que elevamos las declaraciones de modificación de rds_superuser.	session_replication_role
recovery_init_sync_method	Establece el método para sincronizar el directorio de datos antes de la recuperación en caso de bloqueo.	syncfs
remove_temp_files_after_crash	Elimina los archivos temporales después de una caída del backend.	0
restart_after_crash	Reinicia el servidor después de una caída del backend.	–
row_security	Habilita la seguridad de las filas.	–
search_path	Define el orden de búsqueda del esquema para los nombres que no cumplen los requisitos del esquema.	–
seq_page_cost	Establece la estimación de los planificadores del costo de una página de disco obtenida secuencialmente.	–

Nombre del parámetro	Descripción	Predeterminado
session_replication_role	Define el comportamiento de las sesiones para los desencadenadores y las reglas de reescritura.	–
shared_buffers	(8kB) Establece el número de búferes de memoria compartida utilizados por el servidor.	SUM(DBInstanceClassMemory/12038,-50003)
shared_preload_libraries	Enumera las bibliotecas compartidas para precargar en el servidor.	pg_stat_statements
ssl	Habilita las conexiones SSL.	1
ssl_ca_file	Ubicación del archivo de autoridad del servidor SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Ubicación del archivo de certificado del servidor SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_ciphers	Establece la lista de cifrados TLS permitidos que se pueden usar en conexiones seguras.	–
ssl_crl_dir	Ubicación del directorio de lista de revocación de certificados SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Ubicación del archivo de clave privada del servidor SSL	/rdsdbdata/rds-metadata/server-key.pem
ssl_max_protocol_version	Establece la versión máxima de protocolo SSL/TLS permitida	–
ssl_min_protocol_version	Establece la versión mínima de protocolo SSL/TLS permitida	TLSv1.2
standard_conforming_strings	Hace que las cadenas ... traten las barras diagonales invertidas literalmente.	–

Nombre del parámetro	Descripción	Predeterminado
statement_timeout	(ms) Establece la duración máxima permitida de cualquier instrucción.	–
stats_temp_directory	Escribe los archivos de estadísticas temporales en el directorio especificado.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Establece el número de ranuras de conexión reservadas para los superusuarios.	3
synchronize_seqscans	Habilita los exámenes secuenciales sincronizados.	–
synchronous_commit	Define el nivel de sincronización de las transacciones actuales.	on
tcp_keepalives_count	Número máximo de retransmisiones de keepalive de TCP.	–
tcp_keepalives_idle	(s) Tiempo entre la emisión de keepalives de TCP.	–
tcp_keepalives_interval	(s) Tiempo entre retransmisiones de keepalives de TCP.	–
temp_buffers	(8kB) Establece el número máximo de búferes temporales utilizados por cada sesión.	–
temp_file_limit	Restringe la cantidad total de espacio en disco en kilobytes que un proceso PostgreSQL dado puede usar para archivos temporales, sin incluir el espacio usado para tablas temporales explícitas	-1
temp_tablespaces	Define los espacios de tabla que se deben usar para las tablas temporales y los archivos de ordenación.	–

Nombre del parámetro	Descripción	Predeterminado
timezone	Define la zona horaria para visualizar e interpretar las marcas temporales.	UTC
track_activities	Recopila información sobre la ejecución de comandos.	–
track_activity_query_size	Define el tamaño reservado para pg_stat_activity.current_query en bytes.	4096
track_commit_timestamp	Recopila el tiempo de confirmación de transacciones.	–
track_counts	Recopila estadísticas sobre la actividad de la base de datos.	–
track_functions	Recopila estadísticas de nivel de función sobre la actividad de la base de datos.	pl
track_io_timing	Recopila estadísticas temporales sobre la actividad de E/S de la base de datos.	1
track_wal_io_timing	Recopila estadísticas temporales de la actividad de E/S de WAL.	–
transform_null_equals	Trata expr=NULL como expr IS NULL.	–
update_process_title	Actualiza el título del proceso para mostrar el comando SQL activo.	–
vacuum_cost_delay	Retardo del costo del vacío en milisegundos.	–
vacuum_cost_limit	Importe del costo del vacío disponible antes del periodo de reposo.	–
vacuum_cost_page_dirty	Costo del vacío para una página ensuciada por el vacío.	–

Nombre del parámetro	Descripción	Predeterminado
vacuum_cost_page_hit	Costo del vacío para una página encontrada en la caché del búfer.	–
vacuum_cost_page_miss	Costo del vacío para una página no encontrada en la caché del búfer.	0
vacuum_defer_cleanup_age	Número de transacciones por las que se debe aplazar la limpieza de VACUUM y HOT, si la hay.	–
vacuum_failsafe_age	Antigüedad con la que VACUUM debe activar el mecanismo a prueba de errores para evitar una interrupción del reinicio.	1200000000
vacuum_freeze_min_age	Antigüedad mínima con la que VACUUM debe congelar una fila de la tabla.	–
vacuum_freeze_table_age	Antigüedad con la que VACUUM debe escanear toda la tabla para congelar las tuplas.	–
vacuum_multixact_failsafe_age	Antigüedad exacta a la que el VACUUM debe activar el sistema de seguridad para evitar un corte de reinicio.	1200000000
vacuum_multixact_freeze_min_age	Antigüedad mínima con la que VACUUM debe congelar un MultiXactId en una fila de la tabla.	–
vacuum_multixact_freeze_table_age	Antigüedad de multixact con la que VACUUM debe escanear toda la tabla para congelar las tuplas.	–
wal_buffers	(8kB) Establece el número de búferes de páginas de disco en la memoria compartida para WAL.	–

Nombre del parámetro	Descripción	Predeterminado
wal_receiver_create_temp_slot	Establece si un receptor de WAL debe crear una ranura de replicación temporal si no se configura una ranura permanente.	0
wal_receiver_status_interval	(s) Establece el intervalo máximo entre los informes de estado del receptor de WAL al primario.	–
wal_receiver_timeout	(ms) Establece el tiempo máximo de espera para recibir datos del primario.	30000
wal_sender_timeout	(ms) Establece el tiempo máximo de espera para la replicación de WAL.	–
work_mem	(kB) Establece la memoria máxima que se utilizará para los espacios de trabajo de consulta.	–
xmlbinary	Define cómo se deben codificar los valores binarios en XML.	–
xmloption	Define si los datos XML de las operaciones implícitas de análisis y serialización se deben considerar documentos o fragmentos de contenido.	–

Parámetros de nivel de instancia de Aurora PostgreSQL

Puede ver los parámetros de nivel de instancia disponibles para una versión de Aurora PostgreSQL específica usando la consola de administración de AWS, la CLI de AWS o la API de Amazon RDS. Para obtener información sobre cómo ver los parámetros en grupos de parámetros de la base de datos de Aurora PostgreSQL en la consola de RDS, consulte Visualización de los valores de parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Algunos parámetros en el nivel de instancia no están disponibles en todas las versiones y otros están en desuso. Para obtener información sobre cómo ver los parámetros de una versión específica de Aurora PostgreSQL, consulte Visualización de los parámetros del clúster y de la base de datos de Aurora PostgreSQL.

Por ejemplo, en la siguiente tabla se enumeran todos los parámetros que afectan a una instancia de base de datos concreta de un clúster de base de datos de Aurora PostgreSQL. Esta lista se generó al ejecutar el comando de AWS CLI describe-db-parameters con `default.aurora-postgresql14` para el valor `--db-parameter-group-name`.

Para obtener una lista de los parámetros de clúster de base de datos del mismo grupo de parámetros de base de datos predeterminado, consulte Parámetros de nivel de clúster de Aurora PostgreSQL.

Nombre del parámetro	Descripción	Predeterminado
apg_enable_batch_mode_function_execution	Habilita las funciones en modo lote para procesar conjuntos de filas a la vez.	–
apg_enable_correlated_any_transform	Permite que el planificador transforme la subconsulta ANY correlacionada (IN/NOT IN) en JOIN cuando sea posible.	–
apg_enable_function_migration	Permite al planificador migrar las funciones escalares elegibles a la cláusula FROM.	–
apg_enable_not_in_transform	Permite al planificador transformar la subconsulta NOT IN en ANTI JOIN cuando sea posible.	–

Nombre del parámetro	Descripción	Predeterminado
apg_enable_remove_redundant_inner_joins	Permite al planificador eliminar las uniones internas redundantes.	–
apg_enable_semijoin_push_down	Permite el uso de filtros semijoin para las uniones hash.	–
apg_plan_mgmt.capture_plan_baselines	Modo de línea base del plan de captura. manual - habilitar la captura del plan para cualquier sentencia SQL, desactivado - deshabilitar la captura del plan, automático - habilitar la captura del plan para las instrucciones en pg_stat_statements que cumplen los criterios de elegibilidad.	off
apg_plan_mgmt.max_databases	Establece el número máximo de bases de datos que se pueden administrar con apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Establece el número máximo de planes que se pueden almacenar en caché mediante apg_plan_mgmt.	10000
apg_plan_mgmt.plan_retention_period	Número máximo de días desde que un plan se usó por última vez antes de que un plan se elimine automáticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Costo total estimado del plan por debajo del cual se ejecutará un plan no aprobado.	0
apg_plan_mgmt.use_plan_baselines	Use solo planes aprobados o fijos para las instrucciones administradas.	false
application_name	Define el nombre de la aplicación sobre la que informarán las estadísticas y los registros.	–

Nombre del parámetro	Descripción	Predeterminado
aurora_compute_plan_id	Supervisa los planes de ejecución de consultas para detectar los planes de ejecución que contribuyen a la carga actual de la base de datos y para hacer un seguimiento de las estadísticas de rendimiento de los planes de ejecución a lo largo del tiempo. Para obtener más información, consulte Monitorización de planes de ejecución de consultas para Aurora PostgreSQL.	on
aurora_temp_space_size	(MB) Establece el tamaño del espacio asignado para los objetos temporales habilitados para lecturas optimizadas en clústeres optimizados para E/S de Aurora con clases de instancias compatibles.	DBInstanceClassMemory/524288
authentication_timeout	Establece el tiempo máximo permitido para completar una autenticación del cliente.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE para el registro de planes.	–
auto_explain.log_buffers	Usa los búferes de registro.	–
auto_explain.log_format	Usa el formato EXPLAIN para el registro del plan.	–
auto_explain.log_min_duration	Establece el tiempo mínimo de ejecución por encima del cual se registrarán los planes.	–
auto_explain.log_nested_statements	Registra las instrucciones anidadas.	–
auto_explain.log_timing	Recopila datos de cronometraje, no solo recuentos de filas.	–

Nombre del parámetro	Descripción	Predeterminado
auto_explain.log_triggers	Incluye estadísticas de desencadenado en los planes.	–
auto_explain.log_verbose	Usa EXPLAIN VERBOSE para el registro de planes.	–
auto_explain.sample_rate	Permite procesar una fracción de las consultas.	–
babelfishpg_tds.listen_addresses	Establece el nombre de host o las direcciones IP para escuchar el TDS.	*
babelfishpg_tds.tds_debug_log_level	Establece el nivel de registro en TDS, 0 desactiva el registro	1
backend_flush_after	(8Kb) Número de páginas después de las cuales las escrituras realizadas anteriormente se vacían al disco.	–
bytea_output	Establece el formato de salida para bytea.	–
check_function_bodies	Comprueba los cuerpos de las funciones durante la ejecución de CREATE FUNCTION.	–
client_connection_check_interval	Establece el intervalo de tiempo entre las comprobaciones de desconexión mientras se ejecutan las consultas.	–
client_min_messages	Establece los niveles de mensajes que se envían al cliente.	–
config_file	Establece el archivo de configuración principal del servidor.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Habilita el planificador para que use restricciones con el fin de optimizar las consultas.	–

Nombre del parámetro	Descripción	Predeterminado
cpu_index_tuple_cost	Establece la estimación del planificador del costo de procesar cada entrada de índice durante un examen del índice.	–
cpu_operator_cost	Establece la estimación del planificador del costo de procesar cada operador o llamada a una función.	–
cpu_tuple_cost	Establece la estimación del planificador del costo de procesar cada tupla (fila).	–
cron.database_name	Establece la base de datos para almacenar las tablas de metadatos de pg_cron	postgres
cron.log_run	Registra todas las ejecuciones de trabajos en la tabla job_run_details	on
cron.log_statement	Registra todas las instrucciones cron antes de la ejecución.	off
cron.max_running_jobs	Número máximo de trabajos que pueden ejecutarse simultáneamente.	5
cron.use_background_workers	Habilita empleados en segundo plano para pg_cron	on
cursor_tuple_fraction	Establece la estimación de los planificadores de la fracción de filas de un cursor que se recuperará.	–
db_user_namespace	Habilita los nombres de usuario por base de datos.	–
deadlock_timeout	(ms) Establece el tiempo de espera de un bloqueo antes de comprobar si hay un bloqueo.	–

Nombre del parámetro	Descripción	Predeterminado
debug_pretty_print	Aplica una sangría a las visualizaciones del árbol de análisis y de planificación.	–
debug_print_parse	Registra el árbol de análisis de cada consulta.	–
debug_print_plan	Registra el plan de ejecución de cada consulta.	–
debug_print_rewritten	Registra el árbol de análisis reescrito de cada consulta.	–
default_statistics_target	Define el objetivo de estadística predeterminado.	–
default_transaction_deferrable	Define el estado diferible predeterminado de las nuevas transacciones.	–
default_transaction_isolation	Define el nivel de aislamiento de cada nueva transacción.	–
default_transaction_read_only	Define el estado de solo lectura predeterminado de las nuevas transacciones.	–
efective_cache_size	(8kB) Establece la suposición de los planificadores sobre el tamaño de la caché del disco.	SUM(DBInstanceClassMemory/12038,-50003
effective_io_concurrency	Número de solicitudes simultáneas que el subsistema del disco puede gestionar de un modo eficiente.	–
enable_async_append	Habilita el uso de planes de append asíncronos por parte de los planificadores.	–
enable_bitmapscan	Habilita el uso de planes de examen de mapas de bits por parte de los planificadores.	–

Nombre del parámetro	Descripción	Predeterminado
enable_gathermerge	Habilita el uso de planes de fusión de recopilación por parte de los planificadores.	–
enable_hashagg	Habilita el uso de planes de agregación con hash por parte de los planificadores.	–
enable_hashjoin	Habilita el uso de planes de combinación con hash por parte de los planificadores.	–
enable_incremental_sort	Habilita el uso de pasos de ordenación incrementales por parte de los planificadores.	–
enable_indexonlyscan	Habilita el uso de planes de examen solo de índice por parte de los planificadores.	–
enable_indexscan	Habilita el uso de planes de examen de índice por parte de los planificadores.	–
enable_material	Habilita el uso de materialización por parte de los planificadores.	–
enable_memoize	Habilita el uso de memorización para los planificadores	–
enable_mergejoin	Habilita el uso de planes de combinación por fusión por parte de los planificadores.	–
enable_nestloop	Habilita el uso de planes de combinación de bucles anidados por parte de los planificadores.	–
enable_parallel_append	Habilita el uso de planes de append en paralelo por parte de los planificadores.	–
enable_parallel_hash	Habilita el uso de planes hash paralelos por parte de los planificadores.	–

Nombre del parámetro	Descripción	Predeterminado
enable_partition_pruning	Habilita la depuración de particiones en tiempo de planificación y de ejecución.	–
enable_partitionwise_aggregate	Habilita la agregación y agrupación de particiones.	–
enable_partitionwise_join	Habilita la combinación de particiones.	–
enable_seqscan	Habilita el uso de planes de examen secuencial por parte de los planificadores.	–
enable_sort	Habilita el uso de pasos de ordenación explícitos por parte de los planificadores.	–
enable_tidscan	Habilita el uso de planes de examen TID por parte de los planificadores.	–
escape_string_warning	Advierte sobre los escapes de barra invertida en los literales de cadena ordinarios.	–
exit_on_error	Termina la sesión en caso de error.	–
force_parallel_mode	Fuerza el uso de las facilidades de consulta paralela.	–
from_collapse_limit	Define el tamaño de la lista FROM por encima del cual las subconsultas no se contraen.	–
geqo	Habilita la optimización de consultas geneticas.	–
geqo_effort	GEQO: esfuerzo que se usa para definir el valor predeterminado de otros parámetros de GEQO.	–
geqo_generations	GEQO: número de iteraciones del algoritmo.	–

Nombre del parámetro	Descripción	Predeterminado
geqo_pool_size	GEQO: número de individuos de la población.	–
geqo_seed	GEQO: valor de inicialización para la selección de ruta aleatoria.	–
geqo_selection_bias	GEQO: presión selectiva dentro de la población .	–
geqo_threshold	Define el umbral de los elementos FROM por encima de los cuales se usa GEQO.	–
gin_fuzzy_search_limit	Define el resultado máximo permitido para la búsqueda exacta por GIN.	–
gin_pending_list_limit	(kB) Establece el tamaño máximo de la lista pendiente para el índice de GIN.	–
hash_mem_multiplic ador	Múltiplo de work_mem que se utilizará para las tablas hash.	–
hba_file	Establece el archivo de configuración de HBA de servidores.	/rdsdbdata/config/ pg_hba.conf
hot_standby_feedback	Permite la retroalimentación de una espera activa al primario que evitará conflictos de consulta.	on
ident_file	Establece el archivo de configuración de ident de servidores.	/rdsdbdata/config/ pg_hba.conf
in_timeout	(ms) Establece la duración máxima permitida de cualquier transacción en espera.	86400000

Nombre del parámetro	Descripción	Predeterminado
idle_session_timeout	Finaliza cualquier sesión que haya estado inactiva (es decir, en espera de una consulta por parte de un cliente), pero no dentro de una transacción abierta, durante más tiempo del especificado	–
join_collapse_limit	Define el tamaño de la lista FROM por encima del cual las construcciones JOIN no se aplanan.	–
lc_messages	Define el idioma en el que se muestran los mensajes.	–
listen_addresses	Establece el nombre de host o las direcciones IP para escuchar.	*
lo_compat_privileges	Habilita el modo de compatibilidad con versiones pasadas para las comprobaciones de privilegios en objetos grandes.	0
log_connections	Registra cada conexión realizada correctam ente.	–
log_destination	Establece el destino de la salida del registro del servidor.	stderr
log_directory	Establece el directorio de destino para los archivos de registro.	/rdsdbdata/log/error
log_disconnections	Registra el final de una sesión, incluida su duración.	–
log_duration	Registra la duración de cada instrucción de SQL completada.	–
log_error_verbosity	Define el detalle de los mensajes registrados.	–

Nombre del parámetro	Descripción	Predeterminado
log_executor_stats	Escribe las estadísticas de rendimiento del ejecutor en el registro del servidor.	–
log_file_mode	Establece los permisos de los archivos de registro.	0644
log_filename	Define el patrón del nombre de archivo de los archivos de registro.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Inicia un subproceso para capturar el resultado de stderr o csvlogs en archivos de registro.	1
log_hostname	Registra el nombre del host en los registros de conexión.	0
logical_decoding_work_mem	(kB) Cada búfer de reordenamiento interno puede usar esta cantidad de memoria antes de volcar en el disco.	–
log_line_prefix	Controla la información prefijada en cada línea de registro.	%t: %r: %u@%d: %p]:
log_lock_waits	Registra las esperas de bloqueo largas.	–
log_min_duration_sample	(ms) Establece el tiempo mínimo de ejecución a partir del cual se registrará una muestra de instrucciones. El muestreo se determina mediante log_statement_sample_rate.	–
log_min_duration_statement	(ms) Establece el tiempo mínimo de ejecución por encima del cual se registrarán las instrucciones.	–
log_min_error_statement	Hace que se registren todas las instrucciones que generen un error por encima de este nivel.	–

Nombre del parámetro	Descripción	Predeterminado
log_min_messages	Define los niveles de los mensajes que se registran.	–
log_parameter_max_length	(B) Al registrar instrucciones limita los valores de los parámetros registrados a los primeros N bytes.	–
log_parameter_max_length_on_error	(B) Cuando se informa de un error, limita los valores de los parámetros registrados a los primeros N bytes.	–
log_parser_stats	Escribe las estadísticas de rendimiento del analizador en el registro del servidor.	–
log_planner_stats	Escribe las estadísticas de rendimiento del planificador en el registro del servidor.	–
log_replication_commands	Registra cada comando de replicación.	–
log_rotation_age	(min) La rotación automática del archivo de registro se producirá después de N minutos.	60
log_rotation_size	(kB) La rotación automática del archivo de registro se producirá después de N kilobytes.	100000
log_statement	Define el tipo de declaraciones que se deben registrar.	–
log_statement_sample_rate	Fracción de instrucciones que exceden la muestra de log_min_duration que se registrará.	–
log_statement_stats	Escribe las estadísticas de rendimiento acumulativas en el registro del servidor.	–

Nombre del parámetro	Descripción	Predeterminado
log_temp_files	(kB) Registra el uso de archivos temporales mayores a este número de kilobytes.	–
log_timezone	Establece la zona horaria que se usará en los mensajes de registro.	UTC
log_truncate_on_rotation	Permite truncar los archivos de registro existentes con el mismo nombre durante la rotación del registro.	0
maintenance_io_concurrency	Una variante de effective_io_concurrency que se utiliza para trabajos de mantenimiento.	1
maintenance_work_mem	(kB) Establece la memoria máxima que se puede usar para las operaciones de mantenimiento	GREATEST(DBInstanceClassMemory/63963136*1024, 65536)
max_connections	Define el número máximo de conexiones simultáneas.	LEAST(DBInstanceClassMemory/9531392, 5000)
max_files_per_process	Define el número máximo de archivos abiertos simultáneamente para cada proceso del servidor.	–
max_locks_per_transaction	Define el número máximo de bloqueos por transacción.	64
max_parallel_maintenance_workers	Establece el número máximo de procesos en paralelo por operación de mantenimiento.	–
max_parallel_workers	Establece el número máximo de empleados en paralelo que pueden estar activos a la vez.	GREATEST($DBInstanceVCPU/2, 8)

Nombre del parámetro	Descripción	Predeterminado
max_parallel_workers_per_gather	Establece el número máximo de procesos en paralelo por nodo ejecutor.	–
max_pred_locks_per_page	Establece el número máximo de tuplas bloqueadas por predicado por página.	–
max_pred_locks_per_relation	Establece el número máximo de páginas y tuplas bloqueadas por predicado por relación.	–
max_pred_locks_per_transaction	Define el número máximo de bloqueos de predicado por transacción.	–
max_slot_wal_keep_size	(MB) Las ranuras de replicación se marcarán como fallidas, y los segmentos se liberarán para su eliminación o reciclaje si esta cantidad de espacio está ocupada por la WAL en el disco.	–
max_stack_depth	(kB) Establece la profundidad máxima de la pila, en kilobytes.	6144
max_standby_streaming_delay	(ms) Establece el retardo máximo antes de cancelar las consultas cuando un servidor en espera activa está procesando datos de WAL en flujo.	14000
max_worker_processes	Establece el número máximo de procesos de empleados que se ejecutan simultáneamente.	GREATEST($DBInstanceVCPU*2, 8)
min_dynamic_shared_memory	(MB) Cantidad de memoria compartida dinámica reservada al inicio.	–
min_parallel_index_scan_size	(8kB) Establece la cantidad mínima de datos de índice para un examen en paralelo.	–

Nombre del parámetro	Descripción	Predeterminado
min_parallel_table_scan_size	(8kB) Establece la cantidad mínima de datos de tabla para un examen en paralelo.	–
old_snapshot_threshold	(min) Tiempo antes de que una instantánea se vuelva demasiado antigua para leer las páginas modificadas después de la captura de la instantánea.	–
parallel_leader_participation	Controla si Gather y Gather Merge también ejecutan subplanes.	–
parallel_setup_cost	Establece la estimación de los planificadores del costo de iniciar procesos de empleados para la consulta paralela.	–
parallel_tuple_cost	Establece la estimación de los planificadores del costo de pasar cada tupla (fila) del empleado al backend maestro.	–
pgaudit.log	Especifica qué clases de instrucciones se registrarán mediante el registro de auditoría de sesión.	–
pgaudit.log_catalog	Especifica que el registro de sesión se debe habilitar en el caso de que todas las relaciones de una instrucción se encuentren en pg_catalog.	–
pgaudit.log_level	Especifica el nivel de registro que se usará para las entradas de registro.	–
pgaudit.log_parameter	Especifica que el registro de auditoría debe incluir los parámetros que se pasaron con la instrucción.	–

Nombre del parámetro	Descripción	Predeterminado
pgaudit.log_relation	Especifica si el registro de auditoría de sesión debe crear una entrada de registro separada para cada relación (TABLE, VIEW, etc.) referenciada en una instrucción SELECT o DML.	–
pgaudit.log_statement_once	Especifica si el registro incluirá el texto de la instrucción y los parámetros con la primera entrada de registro para una combinación de instrucción o subinstrucción o con cada entrada.	–
pgaudit.role	Especifica el rol maestro que se usará para el registro de auditoría de objetos.	–
pg_bigm.enable_recheck	Especifica si se realiza una recomprobación, que es un proceso interno de búsqueda de texto completo.	on
pg_bigm.gin_key_limit	Especifica el número máximo de bigramas de la palabra clave de búsqueda que se utilizará para la búsqueda de texto completo.	0
pg_bigm.last_update	Informa de la última fecha de actualización del módulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Especifica el umbral mínimo que usa la búsqueda de similitud.	0.3
pg_hint_plan.debug_print	Registra los resultados del análisis sintáctico de las pistas.	–
pg_hint_plan.enable_hint	Fuerza al planificador a usar los planes especificados en el comentario de la pista que precede a la consulta.	–

Nombre del parámetro	Descripción	Predeterminado
pg_hint_plan.enable_hint_table	Fuerza al planificador a no obtener la sugerencia mediante el uso de búsquedas en tablas.	–
pg_hint_plan.message_level	Nivel de mensaje de los mensajes de depuración.	–
pg_hint_plan.parse_messages	Nivel de mensaje de los errores de análisis.	–
pglogical.batch_inserts	Inserciones por lotes si es posible	–
pglogical.conflict_log_level	Establece el nivel de registro utilizado para registrar los conflictos resueltos.	–
pglogical.conflict_resolution	Establece el método que se usa para la resolución de conflictos de los conflictos resolubles.	–
pglogical.extra_connection_options	opciones de conexión para agregar a todas las conexiones de los nodos pares	–
pglogical.synchronous_commit	valor de confirmación sincrónica específico de pglogical	–
pglogical.use_spi	Usa SPI en lugar de la API de bajo nivel para aplicar los cambios	–
pg_similarity.block_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.block_threshold	Establece el umbral utilizado por la función de similitud Bloquear.	–
pg_similarity.block_tokenizer	Establece el tokenizador para la función de similitud de bloques.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.cosine_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.cosine_threshold	Establece el umbral utilizado por la función de similitud coseno.	–
pg_similarity.cosine_tokenizer	Establece el tokenizador para la función de similitud de coseno.	–
pg_similarity.dice_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.dice_threshold	Establece el umbral utilizado por la medida de similitud de dados.	–
pg_similarity.dice_tokenizer	Establece el tokenizador para la medida de similitud de dados.	–
pg_similarity.euclidean_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.euclidean_threshold	Establece el umbral utilizado por la medida de similitud euclidiana.	–
pg_similarity.euclidean_tokenizer	Establece el tokenizador para la medida de similitud euclidiana.	–
pg_similarity.hamming_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.hamming_threshold	Establece el umbral utilizado por la métrica de similitud de bloques.	–
pg_similarity.jaccard_is_normalized	Establece si el valor del resultado está normalizado o no.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.jaccard_threshold	Establece el umbral utilizado por la medida de similitud Jaccard.	–
pg_similarity.jaccard_tokenizer	Establece el tokenizador para la medida de similitud de Jaccard.	–
pg_similarity.jaro_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.jaro_threshold	Establece el umbral utilizado por la medida de similitud Jaro.	–
pg_similarity.jarowinkler_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.jarowinkler_threshold	Establece el umbral utilizado por la medida de similitud Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.levenshtein_threshold	Establece el umbral utilizado por la medida de similitud Levenshtein.	–
pg_similarity.matching_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.matching_threshold	Establece el umbral utilizado por la medida de similitud de coeficiente de coincidencia.	–
pg_similarity.matching_tokenizer	Establece el tokenizador para la medida de similitud de coeficiente de coincidencia.	–
pg_similarity.mongeelkan_is_normalized	Establece si el valor del resultado está normalizado o no.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.mongeelkan_threshold	Establece el umbral utilizado por la medida de similitud Monge-Elkan.	–
pg_similarity.mongeelkan_tokenizer	Establece el tokenizador para la medida de similitud Monge-Elkan.	–
pg_similarity.nw_gap_penalty	Establece la penalización de separación utilizada por la medida de similitud Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.nw_threshold	Establece el umbral utilizado por la medida de similitud Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.overlap_threshold	Establece el umbral utilizado por la medida de similitud de coeficiente de superposición.	–
pg_similarity.overlap_tokenizer	Establece el tokenizador para la medida de similitud del coeficiente de superposición.	–
pg_similarity.qgram_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.qgram_threshold	Establece el umbral utilizado por la medida de similitud Q-Gram.	–
pg_similarity.qgram_tokenizer	Establece el tokenizador para la medida Q-Gram.	–
pg_similarity.swg_is_normalized	Establece si el valor del resultado está normalizado o no.	–

Nombre del parámetro	Descripción	Predeterminado
pg_similarity.swg_threshold	Establece el umbral utilizado por la medida de similitud Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Establece si el valor del resultado está normalizado o no.	–
pg_similarity.sw_threshold	Establece el umbral utilizado por la medida de similitud de Smith-Waterman.	–
pg_stat_statements.max	Establece el número máximo de instrucciones rastreadas por pg_stat_statements.	–
pg_stat_statements.save	Guarda las estadísticas de pg_stat_statements a través de los apagados del servidor.	–
pg_stat_statements.track	Selecciona qué instrucciones rastrea pg_stat_statements.	–
pg_stat_statements.track_planning	Selecciona si la duración de la planificación se rastrea con pg_stat_statements.	–
pg_stat_statements.track_utility	Selecciona si los comandos de utilidad se rastrean mediante pg_stat_statements.	–
postgis.gdal_enabled_drivers	Habilita o desactiva los controladores GDAL que se usan con PostGIS en Postgres 9.3.5 y versiones posteriores.	ENABLE_ALL
quote_all_identifiers	Al generar fragmentos de SQL, cite todos los identificadores.	–
random_page_cost	Establece la estimación de los planificadores del costo de una página de disco que no se recupera secuencialmente.	–

Nombre del parámetro	Descripción	Predeterminado
rds.enable_memory_management	Mejora las capacidades de administración de memoria en las versiones 12.17, 13.13, 14.10, 15.5 y superiores de Aurora PostgreSQL para evitar los problemas de estabilidad y los reinicios de las bases de datos causados por la falta de memoria libre. Para obtener más información, consulte Administración de memoria mejorada en Aurora PostgreSQL.	True
rds.force_admin_logging_level	Consulta los mensajes de registro de las acciones del usuario administrador de RDS en las bases de datos de los clientes.	–
rds.log_retention_period	Amazon RDS eliminará los registros de PostgreSQL que tengan una antigüedad superior a N minutos.	4320
rds.memory_allocation_guard	Mejora las capacidades de administración de memoria en las versiones 11.21, 12.16, 13.12, 14.9, 15.4 y anteriores de Aurora PostgreSQL para evitar los problemas de estabilidad y los reinicios de las bases de datos causados por la falta de memoria libre. Para obtener más información, consulte Administración de memoria mejorada en Aurora PostgreSQL.	False
rds.pg_stat_ramdisk_size	Tamaño del disco de estadísticas en MB. Un valor distinto de cero configurará el disco ramdisk.	0

Nombre del parámetro	Descripción	Predeterminado
rds.rds_superuser_reserved_connections	Establece el número de ranuras de conexión reservadas para rds_superusers. Este parámetro solo está disponible en la versión 15 y anteriores. Para obtener más información, consulte la documentación de PostgreSQL sobre reserved_connections.	2
rds.superuser_variables	Lista de variables solo para superusuarios para las que elevamos las declaraciones de modificación de rds_superuser.	session_replication_role
remove_temp_files_after_crash	Elimina los archivos temporales después de una caída del backend.	0
restart_after_crash	Reinicia el servidor después de una caída del backend.	–
row_security	Habilita la seguridad de las filas.	–
search_path	Define el orden de búsqueda del esquema para los nombres que no cumplen los requisitos del esquema.	–
seq_page_cost	Establece la estimación de los planificadores del costo de una página de disco obtenida secuencialmente.	–
session_replication_role	Define el comportamiento de las sesiones para los desencadenadores y las reglas de reescritura.	–
shared_buffers	(8kB) Establece el número de búferes de memoria compartida utilizados por el servidor.	SUM(DBInstanceClassMemory/12038,-50003)

Nombre del parámetro	Descripción	Predeterminado
shared_preload_libraries	Enumera las bibliotecas compartidas para precargar en el servidor.	pg_stat_statements
ssl_ca_file	Ubicación del archivo de autoridad del servidor SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Ubicación del archivo de certificado del servidor SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_crl_dir	Ubicación del directorio de lista de revocación de certificados SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Ubicación del archivo de clave privada del servidor SSL	/rdsdbdata/rds-metadata/server-key.pem
standard_conforming_strings	Hace que las cadenas ... traten las barras diagonales invertidas literalmente.	–
statement_timeout	(ms) Establece la duración máxima permitida de cualquier instrucción.	–
stats_temp_directory	Escribe los archivos de estadísticas temporales en el directorio especificado.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Establece el número de ranuras de conexión reservadas para los superusuarios.	3
synchronize_seqscans	Habilita los exámenes secuenciales sincronizados.	–
tcp_keepalives_count	Número máximo de retransmisiones de keepalive de TCP.	–
tcp_keepalives_idle	(s) Tiempo entre la emisión de keepalives de TCP.	–

Nombre del parámetro	Descripción	Predeterminado
tcp_keepalives_interval	(s) Tiempo entre retransmisiones de keepalives de TCP.	–
temp_buffers	(8kB) Establece el número máximo de búferes temporales utilizados por cada sesión.	–
temp_file_limit	Restringe la cantidad total de espacio en disco en kilobytes que un proceso PostgreSQL dado puede usar para archivos temporales, sin incluir el espacio usado para tablas temporales explícitas	-1
temp_tablespaces	Define los espacios de tabla que se deben usar para las tablas temporales y los archivos de ordenación.	–
track_activities	Recopila información sobre la ejecución de comandos.	–
track_activity_query_size	Define el tamaño reservado para pg_stat_activity.current_query en bytes.	4096
track_counts	Recopila estadísticas sobre la actividad de la base de datos.	–
track_functions	Recopila estadísticas de nivel de función sobre la actividad de la base de datos.	pl
track_io_timing	Recopila estadísticas temporales sobre la actividad de E/S de la base de datos.	1
transform_–_equals	Treats expr=– as expr IS –.	–
update_process_title	Actualiza el título del proceso para mostrar el comando SQL activo.	–

Nombre del parámetro	Descripción	Predeterminado
wal_receiver_status_interval	(s) Establece el intervalo máximo entre los informes de estado del receptor de WAL al primario.	–
work_mem	(kB) Establece la memoria máxima que se utilizará para los espacios de trabajo de consulta.	–
xmlbinary	Define cómo se deben codificar los valores binarios en XML.	–
xmloption	Define si los datos XML de las operaciones implícitas de análisis y serialización se deben considerar documentos o fragmentos de contenido.	–

Eventos de espera de Amazon Aurora PostgreSQL

A continuación, encontrará eventos de espera frecuentes de Aurora PostgreSQL. Para obtener más información sobre los eventos de espera y activar el clúster de bases de datos de Aurora PostgreSQL, consulte Ajuste con eventos de espera de Aurora PostgreSQL.

Activity:ArchiverMain

El proceso de archivador está esperando la actividad.

Activity:AutoVacuumMain

El proceso del iniciador de autovacuum está esperando la actividad.

Activity:BgWriterHibernate

El proceso de escritor en segundo plano está hibernando mientras se espera la actividad.

Activity:BgWriterMain

El proceso de escritor en segundo plano está esperando la actividad.

Activity:CheckpointerMain

El proceso del verificador de puntos está esperando la actividad.

Activity:LogicalApplyMain

El proceso de aplicación de replicación lógica está esperando la actividad.

Activity:LogicalLauncherMain

El proceso del iniciador de replicación lógica está esperando la actividad.

Activity:PgStatMain

El proceso del recopilador de estadísticas está a la espera de actividad.

Activity:RecoveryWalAll

Un proceso está esperando el registro de escritura anticipada (WAL) de una transmisión en recuperación.

Activity:RecoveryWalStream

El proceso de inicio está esperando a que llegue el registro de escritura anticipada (WAL) durante la recuperación del streaming.

Activity:SysLoggerMain

El proceso de syslogger está esperando la actividad.

Activity:WalReceiverMain

El proceso de recepción del registro de escritura anticipada (WAL) está esperando la actividad.

Activity:WalSenderMain

El proceso de envío del registro de escritura anticipada (WAL) está esperando la actividad.

Activity:WalWriterMain

El proceso de escritura del registro de escritura anticipada (WAL) está esperando la actividad.

BufferPin:BufferPin

Un proceso está esperando para adquirir un pin exclusivo en un buffer.

Client:GSSOpenServer

Un proceso está esperando para leer los datos del cliente mientras se establece una sesión de interfaz de programación de aplicaciones del servicio de seguridad genérico (GSSAPI).

Client:ClientRead

Un proceso de backend está esperando para recibir datos de un cliente de PostgreSQL. Para obtener más información, consulte Client:ClientRead.

Client:ClientWrite

Un proceso de backend está esperando para enviar más datos a un cliente de PostgreSQL. Para obtener más información, consulte Client:ClientWrite.

Client:LibPQWalReceiverConnect

Un proceso está esperando en el receptor del registro de escritura anticipada (WAL) para establecer la conexión con el servidor remoto.

Client:LibPQWalReceiverReceive

Un proceso está esperando en el receptor del registro de escritura anticipada (WAL) para recibir datos del servidor remoto.

Client:SSLOpenServer

Un proceso está esperando la Capa de sockets seguros (SSL) mientras se intenta la conexión.

Client:WalReceiverWaitStart

Un proceso está esperando a que el proceso de inicio envíe datos iniciales para la replicación del streaming.

Client:WalSenderWaitForWAL

Un proceso está esperando a que el registro de escritura anticipada (WAL) se vacíe en el proceso de remitente de WAL.

Client:WalSenderWriteData

Un proceso está esperando cualquier actividad al procesar las respuestas del receptor de registro de escritura anticipada (WAL) en el proceso de remitente de WAL.

CPU

Un proceso de backend está activo en la CPU o a la espera de ella. Para obtener más información, consulte CPU.

Extension:extension

Un proceso de backend está esperando una condición definida por una extensión o módulo.

IO:AuroraEnhancedLogicalWALRead

Un proceso de backend consiste en obtener registros de registro del volumen de captura de datos de cambio (CDC).

IO:AuroraOptimizedReadsCacheRead

Un proceso está esperando una lectura de la caché por niveles de lecturas optimizadas porque la página no está disponible en la memoria compartida.

IO:AuroraOptimizedReadsCacheSegmentTruncate

Un proceso está esperando a que se termine un archivo de segmento de caché por niveles de lecturas optimizadas.

IO:AuroraOptimizedReadsCacheWrite

El proceso de escritor en segundo plano está esperando para escribir en la caché por niveles de lecturas optimizadas.

IO:AuroraStorageLogAllocate

Una sesión está asignando metadatos y se prepara para escribir un registro de transacciones.

IO:BufFileRead

Cuando las operaciones requieren más memoria que la cantidad definida por los parámetros de memoria de trabajo, el motor crea archivos temporales en el disco. Este evento de espera se produce cuando las operaciones leen desde los archivos temporales. Para obtener más información, consulte IO:BufFileRead y IO:BufFileWrite.

IO:BufFileWrite

Cuando las operaciones requieren más memoria que la cantidad definida por los parámetros de memoria de trabajo, el motor crea archivos temporales en el disco. Este evento de espera se produce cuando las operaciones escriben desde los archivos temporales. Para obtener más información, consulte IO:BufFileRead y IO:BufFileWrite.

IO:ControlFileRead

Un proceso está esperando una lectura desde el archivo `pg_control`.

IO:ControlFileSync

Un proceso está esperando a que el archivo `pg_control` se almacene de forma permanente.

IO:ControlFileSyncUpdate

Un proceso está esperando a que una actualización del archivo `pg_control` se almacene de forma permanente.

IO:ControlFileWrite

Un proceso está esperando una escritura en el archivo `pg_control`.

IO:ControlFileWriteUpdate

Un proceso está esperando una escritura para actualizar el archivo `pg_control`.

IO:CopyFileRead

Un proceso está esperando una lectura durante una operación de copia de archivos.

IO:CopyFileWrite

Un proceso está esperando una escritura durante una operación de copia de archivos.

IO:DataFileExtend

Un proceso está esperando a que se amplíe un archivo de datos de relación.

IO:DataFileFlush

Un proceso está esperando a que un archivo de datos de relación se almacene de forma permanente.

IO:DataFileImmediateSync

Un proceso está esperando una sincronización inmediata de un archivo de datos de relación a un almacenamiento permanente.

IO:DataFilePrefetch

Un proceso está esperando una captura previa asíncrona desde un archivo de datos de relación.

IO:DataFileSync

Un proceso está esperando a que los cambios en un archivo de datos de relación se almacenen de forma permanente.

IO:DataFileRead

Un proceso de backend intentó encontrar una página en los búferes compartidos, no la encontró y, por lo tanto, la leyó desde el almacenamiento. Para obtener más información, consulte IO:DataFileRead.

IO:DataFileTruncate

Un proceso está esperando a que se trunque un archivo de datos de relación.

IO:DataFileWrite

Un proceso está esperando una escritura en un archivo de datos de relación.

IO:DSMFillZeroWrite

Un proceso está esperando para escribir cero bytes en un archivo de copia de seguridad de memoria compartida dinámica.

IO:LockFileAddToDataDirRead

Un proceso está esperando una lectura mientras se agrega una línea al archivo de bloqueo del directorio de datos.

IO:LockFileAddToDataDirSync

Un proceso está esperando a que los datos se almacenen de forma permanente mientras se agrega una línea al archivo de bloqueo del directorio de datos.

IO:LockFileAddToDataDirWrite

Un proceso está esperando una escritura mientras se agrega una línea al archivo de bloqueo del directorio de datos.

IO:LockFileCreateRead

Un proceso está esperando para escribir mientras se crea el archivo de bloqueo del directorio de datos.

IO:LockFileCreateSync

Un proceso está esperando a que los datos se almacenen de forma permanente mientras se crea el archivo de bloqueo del directorio de datos.

IO:LockFileCreateWrite

Un proceso está esperando una escritura mientras se crea el archivo de bloqueo del directorio de datos.

IO:LockFileReCheckDataDirRead

Un proceso está esperando una lectura durante la nueva verificación del archivo de bloqueo del directorio de datos.

IO:LogicalRewriteCheckpointSync

Un proceso está esperando a que las asignaciones de reescritura lógica se almacenen de forma permanente durante un punto de verificación.

IO:LogicalRewriteMappingSync

Un proceso está esperando a que los datos de asignación se almacenen de forma permanente durante una rescritura lógica.

IO:LogicalRewriteMappingWrite

Un proceso está esperando una escritura de los datos de asignación durante una reescritura lógica.

IO:LogicalRewriteSync

Un proceso está esperando a que las asignaciones de rescritura lógica se almacenen de forma permanente.

IO:LogicalRewriteTruncate

Un proceso está esperando el truncamiento de los datos de asignación durante una reescritura lógica.

IO:LogicalRewriteWrite

Un proceso está esperando una escritura de las asignaciones de rescritura lógica.

IO:RelationMapRead

Un proceso está esperando una lectura del archivo de asignación de relación.

IO:RelationMapSync

Un proceso está esperando a que el archivo de asignación de relación se almacene de forma permanente.

IO:RelationMapWrite

Un proceso está esperando una escritura en el archivo de asignación de relación.

IO:ReorderBufferRead

Un proceso está esperando una lectura durante la administración del búfer de reordenamiento.

IO:ReorderBufferWrite

Un proceso está esperando una escritura durante la administración del búfer de reordenamiento.

IO:ReorderLogicalMappingRead

Un proceso está a la espera de una lectura de una asignación lógica durante la administración del búfer de reordenamiento.

IO:ReplicationSlotRead

Un proceso está esperando una lectura desde un archivo de control de ranuras de replicación.

IO:ReplicationSlotRestoreSync

Un proceso está esperando a que un archivo de control de ranuras de replicación se almacene de forma permanente mientras lo restaura en la memoria.

IO:ReplicationSlotSync

Un proceso está esperando a que un archivo de control de ranuras de replicación se almacene de forma permanente.

IO:ReplicationSlotWrite

Un proceso está esperando una escritura en un archivo de control de ranuras de replicación.

IO:SLRUFlushSync

Un proceso está esperando a que los datos simples de uso menos reciente (SLRU) se almacenen de forma permanente durante un punto de verificación o cierre de la base de datos.

IO:SLRURead

Un proceso está esperando una lectura de una página simple de uso menos reciente (SLRU).

IO:SLRUSync

Un proceso está esperando a que los datos simples de uso menos reciente (SLRU) se almacenen de forma permanente después de una escritura de página.

IO:SLRUWrite

Un proceso está esperando una escritura de una página simple de uso menos reciente (SLRU).

IO:SnapbuildRead

Un proceso está esperando una lectura de una instantánea del catálogo histórico en serie.

IO:SnapbuildSync

Un proceso está esperando a que una instantánea del catálogo histórico en serie se almacene de forma permanente.

IO:SnapbuildWrite

Un proceso está esperando una escritura de una instantánea del catálogo histórico en serie.

IO:TimelineHistoryFileSync

Un proceso está esperando a que un archivo de historial de cronología recibido a través de la replicación del streaming se almacene de forma permanente.

IO:TimelineHistoryFileWrite

Un proceso está esperando una escritura de un archivo de historial de cronología a través de la replicación del streaming.

IO:TimelineHistoryRead

Un proceso está esperando una lectura de un archivo de historial de cronología.

IO:TimelineHistorySync

Un proceso está esperando a que un archivo de historial de cronología recién creado se almacene de forma permanente.

IO:TimelineHistoryWrite

Un proceso está esperando una escritura de un archivo de historial de cronología recién creado.

IO:TwophaseFileRead

Un proceso está esperando una lectura de un archivo de estado de dos fases.

IO:TwophaseFileSync

Un proceso está esperando a que un archivo de estado de dos fases se almacene de forma permanente.

IO:TwophaseFileWrite

Un proceso está esperando una escritura de un archivo de estado de dos fases.

IO:WALBootstrapSync

Un proceso está esperando a que el registro de escritura anticipada (WAL) se almacene de forma permanente durante el proceso de arranque.

IO:WALBootstrapWrite

Un proceso está esperando una escritura de una página del registro de escritura anticipada (WAL) durante el proceso de arranque.

IO:WALCopyRead

Un proceso está esperando una lectura al crear un nuevo segmento del registro de escritura anticipada (WAL) mediante la copia de uno existente.

IO:WALCopySync

Un proceso está esperando a que un nuevo segmento del registro de escritura anticipada (WAL) creado mediante la copia de uno existente se almacene de forma permanente.

IO:WALCopyWrite

Un proceso está esperando una escritura al crear un nuevo segmento del registro de escritura anticipada (WAL) mediante la copia de uno existente.

IO:WALInitSync

Un proceso está esperando a que un archivo de registro de escritura anticipada (WAL) recién inicializado se almacene de forma permanente.

IO:WALInitWrite

Un proceso está esperando una escritura mientras se inicializa un nuevo archivo de registro de escritura anticipada (WAL).

IO:WALRead

Un proceso está esperando una lectura desde un archivo de registro de escritura anticipada (WAL).

IO:WALSenderTimelineHistoryRead

Un proceso está esperando una lectura desde un archivo de historial de cronología durante un comando de cronología de remitente de WAL.

IO:WALSync

Un proceso está esperando a que un archivo de registro de escritura anticipada (WAL) se almacene de forma permanente.

IO:WALSyncMethodAssign

Un proceso está esperando a que los datos se almacenen de forma permanente mientras se asigna un nuevo método de sincronización del registro de escritura anticipada (WAL).

IO:WALWrite

Un proceso está esperando una escritura en un archivo de registro de escritura anticipada (WAL).

IO:XactSync

Un evento de backend está esperando a que el subsistema de almacenamiento de Aurora reconozca la confirmación de una transacción regular, o bien la confirmación o restauración de una transacción preparada. Para obtener más información, consulte IO:XactSync.

IPC:AuroraLogicalSchemaUpdate

Dos procesos de backend están intentando insertar la misma entrada en la caché del esquema. Un proceso continuará mientras el otro espera a que se complete.

IPC:AuroraOptimizedReadsCacheWriteStop

Un proceso está esperando a que el escritor en segundo plano deje de escribir en la caché por niveles con lecturas optimizadas.

IPC:BackupWaitWalArchive

Un proceso está esperando los archivos de registro de escritura anticipada (WAL) necesarios para archivar correctamente una copia de seguridad.

IPC:BgWorkerShutdown

Un proceso está esperando a que un proceso de trabajo en segundo plano se cierre.

IPC:BgWorkerStartup

Un proceso está esperando a que un proceso de trabajo en segundo plano se inicie.

IPC:BtreePage

Un proceso está esperando a que esté disponible el número de páginas necesario para continuar un análisis paralelo de árbol B.

IPC:CheckpointDone

Un proceso está esperando a que se complete un punto de verificación.

IPC:CheckpointStart

Un proceso está esperando a que se inicie un punto de verificación.

IPC:ClogGroupUpdate

Un proceso está esperando a que el líder del grupo actualice el estado de la transacción al final de una transacción.

IPC:DamRecordTxAck

Un proceso de backend generó un evento de transmisiones de actividad de la base de datos y está esperando a que el evento se vuelva permanente. Para obtener más información, consulte IPC:DamRecordTxAck.

IPC:ExecuteGather

Un proceso está esperando la actividad desde un proceso secundario mientras se ejecuta un nodo del plan de Gather.

IPC:Hash/Batch/Allocating

Un proceso está esperando a que un participante hash paralelo elegido asigne una tabla hash.

IPC:Hash/Batch/Electing

Un proceso elige un participante hash paralelo para asignar una tabla hash.

IPC:Hash/Batch/Loading

Un proceso está esperando a que otros participantes hash paralelos terminen de cargar una tabla hash.

IPC:Hash/Build/Allocating

Un proceso está esperando a que un participante hash paralelo elegido asigne la tabla hash inicial.

IPC:Hash/Build/Electing

Un proceso elige un participante hash paralelo para asignar la tabla hash inicial.

IPC:Hash/Build/HashingInner

Un proceso está esperando a que otros participantes hash paralelos terminen las operaciones hash de la relación interna.

IPC:Hash/Build/HashingOuter

Un proceso está esperando a que otros participantes hash paralelos terminen la partición de la relación externa.

IPC:Hash/GrowBatches/Allocating

Un proceso está esperando a que un participante hash paralelo elegido asigne más lotes.

IPC:Hash/GrowBatches/Deciding

Un proceso elige un participante hash paralelo para decidir sobre el crecimiento futuro de los lotes.

IPC:Hash/GrowBatches/Electing

Un proceso elige un participante hash paralelo para asignar más lotes.

IPC:Hash/GrowBatches/Finishing

Un proceso está esperando a que un participante hash paralelo elegido decida sobre el crecimiento futuro de los lotes.

IPC:Hash/GrowBatches/Repartitioning

Un proceso está esperando a que otros participantes hash paralelos terminen la creación de nuevas particiones.

IPC:Hash/GrowBuckets/Allocating

Un proceso está esperando a que un participante hash paralelo elegido termine de asignar más buckets.

IPC:Hash/GrowBuckets/Electing

Un proceso elige un participante hash paralelo para asignar más buckets.

IPC:Hash/GrowBuckets/Reinserting

Un proceso está esperando a que otros participantes hash paralelos terminen de insertar tuplas en los buckets nuevos.

IPC:HashBatchAllocate

Un proceso está esperando a que un participante hash paralelo elegido asigne una tabla hash.

IPC:HashBatchElect

Un proceso está esperando para elegir un participante hash paralelo para asignar una tabla hash.

IPC:HashBatchLoad

Un proceso está esperando a que otros participantes hash paralelos terminen de cargar una tabla hash.

IPC:HashBuildAllocate

Un proceso está esperando a que un participante hash paralelo elegido asigne la tabla hash inicial.

IPC:HashBuildElect

Un proceso está esperando para elegir un participante hash paralelo para asignar la tabla hash inicial.

IPC:HashBuildHashInner

Un proceso está esperando a que otros participantes hash paralelos terminen las operaciones hash de la relación interna.

IPC:'HashBuildHashOuter

Un proceso está esperando a que otros participantes hash paralelos terminen la partición de la relación externa.

IPC:HashGrowBatchesAllocate

Un proceso está esperando a que un participante hash paralelo elegido asigne más lotes.

IPC:'HashGrowBatchesDecide

Un proceso está esperando para elegir un participante hash paralelo para decidir sobre el crecimiento futuro de los lotes.

IPC:HashGrowBatchesElect

Un proceso está esperando para elegir un participante hash paralelo para asignar más lotes.

IPC:HashGrowBatchesFinish

Un proceso está esperando a que un participante hash paralelo elegido decida sobre el crecimiento futuro de los lotes.

IPC:HashGrowBatchesRepartition

Un proceso está esperando a que otros participantes hash paralelos terminen la creación de nuevas particiones.

IPC:HashGrowBucketsAllocate

Un proceso está esperando a que un participante hash paralelo elegido termine de asignar más buckets.

IPC:HashGrowBucketsElect

Un proceso está esperando para elegir un participante hash paralelo para asignar más buckets.

IPC:HashGrowBucketsReinsert

Un proceso está esperando a que otros participantes hash paralelos terminen de insertar tuplas en los buckets nuevos.

IPC:LogicalSyncData

Un proceso está esperando a que un servidor remoto de replicación lógica envíe datos para la sincronización de la tabla inicial.

IPC:LogicalSyncStateChange

Un proceso está esperando a que un servidor remoto de replicación lógica cambie de estado.

IPC:MessageQueueInternal

Un proceso está esperando a que se adjunte otro proceso a una cola de mensajes compartida.

IPC:MessageQueuePutMessage

Un proceso está esperando para escribir un mensaje de protocolo en una cola de mensajes compartida.

IPC:MessageQueueReceive

Un proceso está esperando para recibir bytes de una cola de mensajes compartida.

IPC:MessageQueueSend

Un proceso está esperando para enviar bytes a una cola de mensajes compartida.

IPC:ParallelBitmapScan

Un proceso está esperando a que se inicialice un análisis de mapa de bits paralelo.

IPC:ParallelCreateIndexScan

Un proceso está esperando a que los procesos de trabajo de CREATE INDEX paralelos terminen un análisis de montón.

IPC:ParallelFinish

Un proceso está esperando a que los procesos de trabajo paralelos terminen de calcular.

IPC:ProcArrayGroupUpdate

Un proceso está esperando a que el líder del grupo elimine el ID de la transacción al final de una transacción.

IPC:ProcSignalBarrier

Un proceso está esperando a que todos los backend procesen un evento de barrera.

IPC:Promote

Un proceso está esperando la promoción en espera.

IPC:RecoveryConflictSnapshot

Un proceso está esperando la resolución de conflictos de recuperación para una limpieza vacuum.

IPC:RecoveryConflictTablespace

Un proceso está esperando a la resolución de conflictos de recuperación para eliminar un espacio de tabla.

IPC:RecoveryPause

Un proceso está esperando a que se reanude la recuperación.

IPC:ReplicationOriginDrop

Un proceso está esperando a que un origen de replicación quede inactivo para que se pueda eliminar.

IPC:ReplicationSlotDrop

Un proceso está esperando a que una ranura de replicación quede inactiva para que pueda eliminarse.

IPC:SafeSnapshot

Un proceso está esperando a obtener una instantánea válida para una transacción READ ONLY DEFERRABLE.

IPC:SyncRep

Un proceso está esperando la confirmación de un servidor remoto durante la replicación síncrona.

IPC:XactGroupUpdate

Un proceso está esperando a que el líder del grupo actualice el estado de la transacción al final de una transacción.

Lock:advisory

Un proceso de backend solicitó un bloqueo de asesoría y lo está esperando. Para obtener más información, consulte Lock:advisory.

Lock:extend

Un proceso de backend está esperando a que se libere un bloqueo para que pueda extender una relación. Este bloqueo es necesario porque solo un proceso de backend puede extender una relación a la vez. Para obtener más información, consulte Lock:extend.

Lock:frozenid

Un proceso está esperando para actualizar `pg_database.datfrozenxid` y `pg_database.datminmxid`.

Lock:object

Un proceso está esperando para obtener un bloqueo en un objeto de base de datos sin relación.

Lock:page

Un proceso está esperando para obtener un bloqueo en una página de una relación.

Lock:Relation

Un proceso de backend está esperando para adquirir un bloqueo de una relación bloqueada por otra transacción. Para obtener más información, consulte Lock:Relation.

Lock:spectoken

Un proceso está esperando para obtener un bloqueo de inserción especulativa.

Lock:speculative token

Un proceso está esperando para adquirir un bloqueo de inserción especulativa.

Lock:transactionid

Una transacción está esperando un bloqueo a nivel de fila. Para obtener más información, consulte Lock:transactionid.

Lock:tuple

Un proceso de backend está esperando para adquirir un bloqueo en una tupla, mientras que otro proceso de backend mantiene un bloqueo conflictivo en la misma tupla. Para obtener más información, consulte Lock:tuple.

Lock:userlock

Un proceso está esperando para obtener un bloqueo de usuario.

Lock:virtualxid

Un proceso está esperando para obtener un bloqueo de ID de transacción virtual.

LWLock:AddinShmemInit

Un proceso está esperando para administrar la asignación de espacio de una extensión en la memoria compartida.

LWLock:AddinShmemInitLock

Un proceso está esperando para administrar la asignación de espacio de la memoria compartida.

LWLock:async

Un proceso está esperando la E/S de un búfer asíncrono (notificación).

LWLock:AsyncCtlLock

Un proceso está esperando para leer o actualizar un estado de notificación compartida.

LWLock:AsyncQueueLock

Un proceso está esperando para leer o actualizar los mensajes de notificación.

LWLock:AuroraOptimizedReadsCacheMapping

Un proceso está esperando para asociar un bloque de datos a una página en la caché por niveles de lecturas optimizadas.

LWLock:AutoFile

Un proceso está esperando para actualizar el archivo `postgresql.auto.conf`.

LWLock:AutoFileLock

Un proceso está esperando para actualizar el archivo `postgresql.auto.conf`.

LWLock:Autovacuum

Un proceso está esperando para leer o actualizar el estado actual de los procesos de trabajo de autovacuum.

LWLock:AutovacuumLock

Un proceso de trabajo o iniciador de autovacuum está esperando para actualizar o leer el estado actual de los procesos de trabajo de autovacuum.

LWLock:AutovacuumSchedule

Un proceso está esperando para garantizar que una tabla seleccionada para autovacuum aún necesita la operación vacuum.

LWLock:AutovacuumScheduleLock

Un proceso está esperando para garantizar que la tabla que seleccionó para una operación vacuum aún la necesita.

LWLock:BackendRandomLock

Un proceso está esperando para generar un número aleatorio.

LWLock:BackgroundWorker

Un proceso está esperando para leer o actualizar el estado del proceso de trabajo en segundo plano.

LWLock:BackgroundWorkerLock

Un proceso está esperando para leer o actualizar el estado del proceso de trabajo en segundo plano.

LWLock:BtreeVacuum

Un proceso está esperando para leer o actualizar la información relacionada con vacuum de un índice del árbol B.

LWLock:BtreeVacuumLock

Un proceso está esperando para leer o actualizar la información relacionada con vacuum de un índice del árbol B.

LWLock:buffer_content

Un proceso de backend está esperando para adquirir un bloqueo ligero sobre el contenido de un búfer de memoria compartida. Para obtener más información, consulte LWLock:buffer_content (BufferContent).

LWLock:buffer_mapping

Un proceso de backend está esperando para asociar un bloque de datos a un búfer del grupo de búferes compartido. Para obtener más información, consulte LWLock:buffer_mapping.

LWLock:BufferIO

Un proceso de backend quiere leer una página en la memoria compartida. El proceso está esperando a que otros procesos finalicen su E/S de la página. Para obtener más información, consulte LWLock:BufferIO (IPC:BufferIO).

LWLock:Checkpoint

Un proceso está esperando para iniciar un punto de verificación.

LWLock:CheckpointLock

Un proceso está esperando para hacer un punto de verificación.

LWLock:CheckpointerComm

Un proceso está esperando para administrar las solicitudes de `fsync`.

LWLock:CheckpointerCommLock

Un proceso está esperando para administrar las solicitudes de `fsync`.

LWLock:clog

Un proceso está esperando la E/S en un búfer de obstrucción (estado de transacción).

LWLock:CLogControlLock

Un proceso está esperando para leer o actualizar el estado de la transacción.

LWLock:CLogTruncationLock

Un proceso está esperando para ejecutar `txid_status` o actualizar el ID de transacción más antiguo que tenga disponible.

LWLock:commit_timestamp

Un proceso está esperando la E/S en un búfer de marca de tiempo de confirmación.

LWLock:CommitTs

Un proceso está esperando para leer o actualizar el último valor establecido para una marca de tiempo de confirmación de transacción.

LWLock:CommitTsBuffer

Un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para una marca de tiempo de confirmación.

LWLock:CommitTsControlLock

Un proceso está esperando para leer o actualizar las marcas de tiempo de confirmación de transacciones.

LWLock:CommitTsLock

Un proceso está esperando para leer o actualizar el último valor establecido para la marca de tiempo de la transacción.

LWLock:CommitTsSLRU

Un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) para una marca de tiempo de confirmación.

LWLock:ControlFile

Un proceso está esperando para leer o actualizar el archivo `pg_control` o crear un nuevo archivo de registro de escritura anticipada (WAL).

LWLock:ControlFileLock

Un proceso está esperando para leer o actualizar el archivo de control o la creación de un nuevo archivo de registro de escritura anticipada (WAL).

LWLock:DynamicSharedMemoryControl

Un proceso está esperando para leer o actualizar la información de asignación dinámica de memoria compartida.

LWLock:DynamicSharedMemoryControlLock

Un proceso está esperando para leer o actualizar el estado de la memoria compartida dinámica.

LWLock:lock_manager

Un proceso de backend está esperando para agregar o examinar los bloqueos de los procesos de backend. O bien está esperando para unirse o salir de un grupo de bloqueo utilizado por una consulta paralela. Para obtener más información, consulte LWLock:lock_manager.

LWLock:LockFastPath

Un proceso está esperando para leer o actualizar la información de bloqueo de método rápido de un proceso.

LWLock:LogicalRepWorker

Un proceso está esperando para leer o actualizar el estado de los procesos de trabajo de replicación lógica.

LWLock:LogicalRepWorkerLock

Un proceso está esperando a que finalice una acción en un proceso de trabajo de replicación lógica.

LWLock:LogicalSchemaCache

Un proceso ha modificado la caché del esquema.

LWLock:multixact_member

Un proceso está esperando la E/S en un búfer de multixact_member.

LWLock:multixact_offset

Un proceso está esperando la E/S en un búfer de desplazamiento de multixact.

LWLock:MultiXactGen

Un proceso está esperando para leer o actualizar el estado de multixact compartido.

LWLock:MultiXactGenLock

Un proceso está esperando para leer o actualizar un estado de multixact compartido.

LWLock:MultiXactMemberBuffer

Un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para un miembro de multixact. Para obtener más información, consulte LWLock:MultiXact.

LWLock:MultiXactMemberControlLock

Un proceso está esperando para leer o actualizar las asignaciones de miembros de multixact.

LWLock:MultiXactMemberSLRU

Un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) de un miembro de multixact. Para obtener más información, consulte LWLock:MultiXact.

LWLock:MultiXactOffsetBuffer

Un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para un desplazamiento de multixact. Para obtener más información, consulte LWLock:MultiXact.

LWLock:MultiXactOffsetControlLock

Un proceso está esperando para leer o actualizar las asignaciones de desplazamiento de multixact.

LWLock:MultiXactOffsetSLRU

Un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) de un desplazamiento de multixact. Para obtener más información, consulte LWLock:MultiXact.

LWLock:MultiXactTruncation

Un proceso está esperando para leer o truncar información de multixact.

LWLock:MultiXactTruncationLock

Un proceso está esperando para leer o truncar información de multixact.

LWLock:NotifyBuffer

Un proceso está esperando la E/S en el búfer simple de uso menos reciente (SLRU) para un mensaje de NOTIFY.

LWLock:NotifyQueue

Un proceso está esperando para leer o actualizar los mensajes de NOTIFY.

LWLock:NotifyQueueTail

Un proceso está esperando para actualizar un límite en el almacenamiento de mensajes de NOTIFY.

LWLock:NotifyQueueTailLock

Un proceso está esperando para actualizar el límite del almacenamiento de mensajes de notificación.

LWLock:NotifySLRU

Un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) para un mensaje de NOTIFY.

LWLock:OidGen

Un proceso está esperando para asignar un ID de objeto (OID) nuevo.

LWLock:OidGenLock

Un proceso está esperando para asignar un ID de objeto (OID).

LWLock:oldserxid

Un proceso está esperando la E/S en un búfer de oldserxid.

LWLock:OldSerXidLock

Un proceso está esperando para leer o registrar transacciones serializables en conflicto.

LWLock:OldSnapshotTimeMap

Un proceso está esperando para leer o actualizar la información antigua del control de instantáneas.

LWLock:OldSnapshotTimeMapLock

Un proceso está esperando para leer o actualizar la información antigua del control de instantáneas.

LWLock:parallel_append

Un proceso está esperando para elegir el siguiente subplan durante la ejecución del plan anexado paralelo.

LWLock:parallel_hash_join

Un proceso está esperando para asignar o intercambiar un fragmento de memoria o actualizar los contadores durante la ejecución de un plan hash paralelo.

LWLock:parallel_query_dsa

Un proceso está esperando a que se bloquee la asignación dinámica de memoria compartida para una consulta paralela.

LWLock:ParallelAppend

Un proceso está esperando para elegir el siguiente subplan durante la ejecución del plan anexado paralelo.

LWLock:ParallelHashJoin

Un proceso está esperando para sincronizar los procesos de trabajo durante la ejecución del plan para una combinación hash paralela.

Lwlock:ParallelQueryDSA

Un proceso está esperando la asignación dinámica de memoria compartida para una consulta paralela.

Lwlock:PerSessionDSA

Un proceso está esperando la asignación dinámica de memoria compartida para una consulta paralela.

Lwlock:PerSessionRecordType

Un proceso está esperando para acceder a la información de una consulta paralela acerca de los tipos compuestos.

Lwlock:PerSessionRecordTypmod

Un proceso está esperando para acceder a la información de una consulta paralela acerca de los modificadores de tipo que identifican tipos de registros anónimos.

Lwlock:PerXactPredicateList

Un proceso está esperando para acceder a la lista de bloqueos de predicados que la transacción serializable actual mantiene durante una consulta paralela.

Lwlock:predicate_lock_manager

Un proceso está esperando para agregar o examinar la información de bloqueo de predicados.

Lwlock:PredicateLockManager

Un proceso está esperando para acceder a la información de bloqueo de predicado utilizada por las transacciones serializables.

Lwlock:proc

Un proceso está esperando para leer o actualizar la información de bloqueo de método rápido.

LWLock:ProcArray

Un proceso está esperando para acceder a las estructuras de datos compartidas por proceso (normalmente, para obtener una instantánea o informar del ID de transacción de una sesión).

LWLock:ProcArrayLock

Un proceso está esperando para obtener una instantánea o borrar un ID de transacción al final de una transacción.

LWLock:RelationMapping

Un proceso está esperando para leer o actualizar un archivo `pg_filenode.map` (utilizado para hacer un seguimiento de las asignaciones de nodos de archivo de determinados catálogos del sistema).

LWLock:RelationMappingLock

Un proceso está esperando para actualizar el archivo de asignación de relaciones utilizado para almacenar la asignación de catálogo a nodo de archivo.

LWLock:RelCacheInit

Un proceso está esperando para leer o actualizar un archivo `pg_internal.init` (un archivo de inicialización de la caché de relaciones).

LWLock:RelCacheInitLock

Un proceso está esperando para leer o escribir un archivo de inicialización de la caché de relaciones.

LWLock:replication_origin

Un proceso está esperando para leer o actualizar el progreso de la replicación.

LWLock:replication_slot_io

Un proceso está esperando la E/S en una ranura de replicación.

LWLock:ReplicationOrigin

Un proceso está esperando para crear, eliminar o utilizar un origen de replicación.

LWLock:ReplicationOriginLock

Un proceso está esperando para configurar, eliminar o utilizar un origen de replicación.

LWLock:ReplicationOriginState

Un proceso está esperando para leer o actualizar el progreso de un origen de replicación.

LWLock:ReplicationSlotAllocation

Un proceso está esperando para asignar o liberar una ranura de replicación.

LWLock:ReplicationSlotAllocationLock

Un proceso está esperando para asignar o liberar una ranura de replicación.

LWLock:ReplicationSlotControl

Un proceso está esperando para leer o actualizar un estado de ranura de replicación.

LWLock:ReplicationSlotControlLock

Un proceso está esperando para leer o actualizar el estado de la ranura de replicación.

LWLock:ReplicationSlotIO

Un proceso está esperando la E/S en una ranura de replicación.

LWLock:SerialBuffer

Un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para un conflicto de transacciones serializable.

LWLock:SerializableFinishedList

Un proceso está esperando para acceder a la lista de transacciones serializables terminadas.

LWLock:SerializableFinishedListLock

Un proceso está esperando para acceder a la lista de transacciones serializables terminadas.

LWLock:SerializablePredicateList

Un proceso está esperando para acceder a la lista de bloqueos de predicados almacenados por las transacciones serializables.

LWLock:SerializablePredicateLockListLock

Un proceso está esperando para hacer una operación en una lista de bloqueos que las transacciones serializables mantienen.

LWLock:SerializableXactHash

Un proceso está esperando para leer o actualizar la información acerca de las transacciones serializables.

LWLock:SerializableXactHashLock

Un proceso está esperando para recuperar o almacenar información acerca de las transacciones serializables.

LWLock:SerialSLRU

Un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) para un conflicto de transacciones serializable.

LWLock:SharedTidBitmap

Un proceso está esperando para acceder a un mapa de bits de identificador de tupla compartido (TID) durante un análisis de índice de mapa de bits paralelo.

LWLock:SharedTupleStore

Un proceso está esperando para acceder a un almacén de tuplas compartido durante una consulta paralela.

LWLock:ShmemIndex

Un proceso está esperando para buscar o asignar espacio en la memoria compartida.

LWLock:ShmemIndexLock

Un proceso está esperando para buscar o asignar espacio en la memoria compartida.

LWLock:SInvalRead

Un proceso está esperando para recuperar mensajes de la cola de invalidación de catálogos compartida.

LWLock:SInvalReadLock

Un proceso está esperando para recuperar o eliminar mensajes de una cola de invalidación compartida.

LWLock:SInvalWrite

Un proceso está esperando para agregar un mensaje de la cola de invalidación de catálogos compartida.

LWLock:SInvalWriteLock

Un proceso está esperando para agregar un mensaje en una cola de invalidación compartida.

LWLock:SyncRep

Un proceso está esperando para leer o actualizar información acerca del estado de la replicación síncrona.

LWLock:SyncRepLock

Un proceso está esperando para leer o actualizar información acerca de las réplicas síncronas.

LWLock:SyncScan

Un proceso está esperando para seleccionar la ubicación inicial de un análisis de tabla sincronizado.

LWLock:SyncScanLock

Un proceso está esperando para obtener la ubicación inicial de una tabla para los análisis sincronizados.

LWLock:TablespaceCreate

Un proceso está esperando para crear o eliminar un espacio de tabla.

LWLock:TablespaceCreateLock

Un proceso está esperando para crear o eliminar el espacio de tabla.

LWLock:tbm

Un proceso está esperando un bloqueo de iterador compartido en un mapa de bits de árbol (TBM).

LWLock:TwoPhaseState

Un proceso está esperando para leer o actualizar el estado de las transacciones preparadas.

LWLock:TwoPhaseStateLock

Un proceso está esperando para leer o actualizar el estado de las transacciones preparadas.

LWLock:wal_insert

Un proceso está esperando para insertar el registro de escritura anticipada (WAL) en un búfer de memoria.

LWLock:WALBufMapping

Un proceso está esperando para reemplazar una página en los búferes del registro de escritura anticipada (WAL).

LWLock:WALBufMappingLock

Un proceso está esperando para reemplazar una página en los búferes del registro de escritura anticipada (WAL).

LWLock:WALInsert

Un proceso está esperando para insertar los datos del registro de escritura anticipada (WAL) en un búfer de memoria.

LWLock:WALWrite

Un proceso está esperando a que se escriban los búferes del registro de escritura anticipada (WAL) en el disco.

LWLock:WALWriteLock

Un proceso está esperando a que se escriban los búferes del registro de escritura anticipada (WAL) en el disco.

LWLock:WrapLimitsVacuum

Un proceso está esperando para actualizar los límites del ID de transacción y el consumo de multixact.

LWLock:WrapLimitsVacuumLock

Un proceso está esperando para actualizar los límites del ID de transacción y el consumo de multixact.

LWLock:XactBuffer

Un proceso está esperando la E/S en un búfer simple de uso menos reciente (SLRU) para un estado de transacción.

LWLock:XactSLRU

Un proceso está esperando para acceder a la caché simple de uso menos reciente (SLRU) para un estado de transacción.

LWLock:XactTruncation

Un proceso está esperando para ejecutar pg_xact_status o actualizar el ID de transacción más antiguo que tiene disponible.

LWLock:XidGen

Un proceso está esperando para asignar un nuevo ID de transacción.

LWLock:XidGenLock

Un proceso está esperando para asignar un ID de transacción.

Timeout:BaseBackupThrottle

Un proceso está esperando durante la copia de seguridad base cuando hicieron limitaciones controladas de la actividad.

Timeout:PgSleep

Un proceso de backend llamó a la función pg_sleep y espera a que el tiempo de espera venza. Para obtener más información, consulte Timeout:PgSleep.

Timeout:RecoveryApplyDelay

Un proceso está esperando para aplicar el registro de escritura anticipada (WAL) durante la recuperación debido a una configuración de retraso.

Timeout:RecoveryRetrieveRetryInterval

Un proceso está esperando durante la recuperación cuando los datos del registro de escritura anticipada (WAL) no están disponibles desde ningún origen (pg_wal, archivo o transmisión).

Timeout:VacuumDelay

Un proceso está esperando en un punto de retraso de vacuum basado en costos.

Para una lista completa de eventos de espera de PostgreSQL, consulte la tabla de eventos de espera de PostgreSQL en la documentación de PostgreSQL.

Actualizaciones del motor de base de datos de Amazon Aurora PostgreSQL

A continuación, puede encontrar información sobre las versiones y actualizaciones del motor de Amazon Aurora PostgreSQL. También puede encontrar información sobre cómo actualizar su motor de Aurora PostgreSQL. Para obtener más información sobre las versiones de Aurora en general, consulte Versiones de Amazon Aurora.

> **Tip**
>
> Puede minimizar el tiempo de inactividad necesario para la actualización de un clúster de base de datos mediante una implementación azul/verde. Para obtener más información, consulte Uso de las implementaciones azul/verde para actualizar las bases de datos.

Temas

- Identificación de las versiones de Amazon Aurora PostgreSQL
- Versiones de Amazon Aurora PostgreSQL y versiones del motor
- Versiones de extensión para Amazon Aurora PostgreSQL
- Actualización de clústeres de base de datos PostgreSQL de Amazon Aurora
- Uso de una versión de compatibilidad a largo plazo (LTS) de Aurora PostgreSQL

Identificación de las versiones de Amazon Aurora PostgreSQL

Amazon Aurora incluye algunas características que son generales para Aurora y están disponibles para todos los clústeres de base de datos Aurora. Aurora incluye otras características específicas de un motor de base de datos en particular compatible con Aurora. Estas características están disponibles solo para aquellos clústeres de base de datos Aurora que usan ese motor de base de datos, como Aurora PostgreSQL.

Una versión de base de datos de Aurora suele tener dos números de versión, el número de versión del motor de base de datos y el número de versión de Aurora. Si una versión de Aurora PostgreSQL tiene un número de versión de Aurora, se incluye después del número de versión del motor en el listado de versiones de Versiones de Amazon Aurora PostgreSQL y versiones del motor.

Temas

- Número de versión de Aurora
- Números de versión del motor de PostgreSQL

Número de versión de Aurora

Los números de versión de Aurora usan el esquema de nomenclatura *major*.*minor*.*patch*. Una versión de revisión de Aurora incluye correcciones de errores importantes que se agregan a una versión menor después de su lanzamiento. Para obtener más información sobre las versiones mayores, menores y de revisión de Amazon Aurora, consulte Versiones principales de Amazon Aurora, Versiones secundarias de Amazon Aurora, y Versiones de parches de Amazon Aurora.

Puede averiguar el número de versión de Aurora de su instancia de base de datos de Aurora PostgreSQL con la siguiente consulta SQL:

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

A partir del lanzamiento de las versiones 13.3, 12.8, 11.13 y 10.18 de PostgreSQL, y para todas las demás versiones posteriores, los números de versión de Aurora se ajustan más a la versión del motor de PostgreSQL. Por ejemplo, la consulta de un clúster de bases de datos de Aurora PostgreSQL 13.3 muestra lo siguiente:

```
aurora_version
----------------
 13.3.1
(1 row)
```

Las versiones anteriores, como el clúster de bases de datos de Aurora PostgreSQL 10.14, muestran números de versión similares a los siguientes:

```
aurora_version
----------------
 2.7.3
(1 row)
```

Números de versión del motor de PostgreSQL

A partir de PostgreSQL 10, las versiones del motor de base de datos PostgreSQL utilizan un esquema de numeración *major*.*minor* para todas las versiones. Algunos ejemplos incluyen PostgreSQL 10.18, PostgreSQL 12.7 y PostgreSQL 13.3.

Las versiones anteriores a PostgreSQL 10 usaban un esquema de numeración *major*.*major*.*minor* en el que los dos primeros dígitos conforman el número de la versión major (principal) y un tercer dígito denota una versión minor (secundaria). Por ejemplo, PostgreSQL 9.6 es una versión principal. Las versiones secundarias 9.6.21 o 9.6.22 incluyen el tercer dígito.

> **Note**
>
> Ya no se admite la versión 9.6 del motor PostgreSQL. Para actualizar, consulta Actualización de clústeres de base de datos PostgreSQL de Amazon Aurora. Para conocer las políticas de versión y los plazos de lanzamiento, consulte Cuánto tiempo permanecen disponibles las versiones principales de Amazon Aurora.

Puede averiguar el número de versión del motor de base de datos PostgreSQL con la siguiente consulta SQL:

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

Para un clúster de bases de datos de Aurora PostgreSQL 13.3, los resultados son los siguientes:

```
version
------------------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
(1 row)
```

Versiones de Amazon Aurora PostgreSQL y versiones del motor

Las versiones de Amazon Aurora PostgreSQL-Compatible Edition se actualizan regularmente. Las actualizaciones se aplican a clústeres de bases de datos Aurora PostgreSQL durante los períodos de mantenimiento del sistema. El momento en el que se aplican las actualizaciones depende del tipo de actualización, la Región de AWS, y de la configuración del periodo de mantenimiento para el clúster de bases de datos. Muchas de las versiones enumeradas incluyen tanto un número de versión de PostgreSQL como un número de versión de Amazon Aurora. Sin embargo, a partir del lanzamiento de las versiones 13.3, 12.8, 11.13 y 10.18 de PostgreSQL, y para todas las demás versiones posteriores, no se usan los números de versión de Aurora. Para determinar los números de versión de su base de datos de Aurora PostgreSQL, consulte Identificación de las versiones de Amazon Aurora PostgreSQL.

Para obtener información sobre extensiones y módulos, consulte Versiones de extensión para Amazon Aurora PostgreSQL.

> **Note**
>
> Para obtener información sobre las versiones, las políticas y los plazos disponibles de Amazon Aurora, consulte Cuánto tiempo permanecen disponibles las versiones principales de Amazon Aurora.

Para obtener información sobre la compatibilidad de Amazon Aurora, consulte las preguntas frecuentes de Amazon RDS.

Para determinar qué versiones del motor de base de datos de Aurora PostgreSQL están disponibles en una Región de AWS, utilice el comando describe-db-engine-versions de la AWS CLI. Por ejemplo:

```
aws rds describe-db-engine-versions --engine aurora-postgresql --query '*[].
[EngineVersion]' --output text --region aws-region
```

Para obtener una lista de Regiones de AWS, consulte Disponibilidad por región de Aurora PostgreSQL.

Para obtener más información sobre las versiones de PostgreSQL disponibles en Aurora PostgreSQL, consulte las Notas de la versión de Aurora PostgreSQL.

Versiones de extensión para Amazon Aurora PostgreSQL

Puede instalar y configurar varias extensiones PostgreSQL para usarlas con clústeres de bases de datos Aurora PostgreSQL. Puede utilizar la extensión `pg_partman` de PostgreSQL para automatizar la creación y el mantenimiento de las particiones de tablas. Para obtener más información sobre esta y otras extensiones disponibles para Aurora PostgreSQL, consulte Uso de extensiones y contenedores de datos externos.

Para obtener más información sobre las versiones de PostgreSQL compatibles con Aurora PostgreSQL, consulte las versiones de extensiones de Amazon Aurora PostgreSQL en las notas de la versión de Aurora PostgreSQL.

Actualización de clústeres de base de datos PostgreSQL de Amazon Aurora

Amazon Aurora hace que estén disponibles en Regiones de AWS las nuevas versiones del motor de base de datos PostgreSQL solo después de realizar pruebas exhaustivas. Puede actualizar los clústeres de base de datos de Aurora PostgreSQL a la nueva versión cuando esté disponible en su región.

Según la versión de Aurora PostgreSQL que el clúster de base de datos esté ejecutando actualmente, una actualización a la nueva versión es una actualización secundaria o principal. Por ejemplo, actualizar un clúster de base de datos de Aurora PostgreSQL 11.15 a Aurora PostgreSQL 13.6 es una actualización de versión principal. La actualización de un clúster de base de datos de Aurora PostgreSQL 13.3 a Aurora PostgreSQL 13.7 es una actualización de versión secundaria. En los temas siguientes, encontrará información sobre cómo realizar ambos tipos de actualizaciones.

Contenido

- Información general de los procesos de actualización de Aurora PostgreSQL
- Obtención de una lista de versiones disponibles en su Región de AWS
- Actualización a una versión principal
  - Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal
  - Recomendaciones posteriores a la actualización
  - Actualización del motor de Aurora PostgreSQL a una nueva versión principal
    - Actualizaciones importantes para bases de datos globales
- Actualización a una versión secundaria
  - Antes de realizar una actualización de versión secundaria
  - Cómo realizar actualizaciones de versión secundarias y aplicar revisiones
  - Actualizaciones de versión secundarias y aplicación de revisiones sin tiempo de inactividad
  - Limitaciones de parches sin tiempo de inactividad
  - Actualización del motor de Aurora PostgreSQL a una nueva versión secundaria
- Actualización de las extensiones de PostgreSQL
- Técnica alternativa de actualización azul/verde

Información general de los procesos de actualización de Aurora PostgreSQL

Las diferencias entre las actualizaciones de versión principales y secundarias son las siguientes:

Actualizaciones y revisiones de versión secundarias

Las actualizaciones y revisiones de versiones secundarias incluyen solo los cambios que son compatibles con las aplicaciones existentes. Las actualizaciones y revisiones de versión secundarias estarán disponibles solo después de que Aurora PostgreSQL las pruebe y apruebe.

Aurora puede aplicar actualizaciones de versiones secundarias automáticamente. Cuando crea un nuevo clúster de bases de datos Aurora PostgreSQL, la opción Habilitar la actualización de la versión menor está habilitada de forma predeterminada. A menos que desactive manualmente esta opción, Aurora aplica periódicamente las actualizaciones de versión secundaria durante el periodo de mantenimiento programado. Para obtener más información sobre la opción de actualización automática de versiones secundarias (AmVU) y cómo modificar el clúster de base de datos de Aurora para utilizarla, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora.

Si la opción de actualización automática de versión secundaria no está habilitada para el clúster de bases de datos de Aurora PostgreSQL, Aurora PostgreSQL no se actualiza automáticamente a la nueva versión secundaria. En su lugar, cuando se publica una nueva versión secundaria en su Región de AWS y el clúster de base de datos de Aurora PostgreSQL ejecuta una versión secundaria anterior, Aurora le pide que efectúe la actualización. Para ello, agrega una recomendación a las tareas de mantenimiento del clúster.

Las revisiones no se consideran una actualización y no se aplican automáticamente. Aurora PostgreSQL le pide que aplique revisiones mediante la incorporación de una recomendación a las tareas de mantenimiento de su clúster de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Cómo realizar actualizaciones de versión secundarias y aplicar revisiones.

> **Note**
>
> Las revisiones que resuelven problemas de seguridad u otros problemas críticos también se agregan como tareas de mantenimiento. No obstante, estas revisiones son necesarias. Asegúrese de aplicar revisiones de seguridad a su clúster de base de datos de Aurora PostgreSQL cuando estén disponibles en sus tareas de mantenimiento pendientes. Las actualizaciones de versiones secundarias automáticas se realizan a la versión secundaria predeterminada.

El proceso de actualización implica la posibilidad de que se produzcan breves interrupciones mientras se actualiza cada instancia del clúster a la nueva versión. No obstante, después de las versiones 14.3.3, 13.7.3, 12.11.3, 11.16.3, 10.21.3 de Aurora PostgreSQL y otras versiones posteriores de estas versiones secundarias y las versiones principales recientes, el proceso de actualización utiliza la característica de aplicación de revisiones sin tiempo de inactividad (ZDP). Esta característica minimiza las interrupciones y, en la mayoría de los casos, las elimina por completo. Para obtener más información, consulte Actualizaciones de versión secundarias y aplicación de revisiones sin tiempo de inactividad. Para obtener más información sobre las características y limitaciones compatibles de ZDP, consulte Limitaciones de parches sin tiempo de inactividad.

Actualizaciones de la versión principal

A diferencia de lo que ocurre con las actualizaciones de versión secundarias y las revisiones, Aurora PostgreSQL no dispone de una opción de actualización automática de la versión principal. Las nuevas versiones principales de PostgreSQL pueden contener cambios en la base de datos que no sean compatibles con las aplicaciones existentes. La nueva funcionalidad puede provocar que sus aplicaciones existentes dejen de funcionar correctamente.

Para evitar problemas, le recomendamos que siga el proceso descrito en Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal antes de actualizar las instancias de base de datos en sus clústeres de Aurora PostgreSQL. En primer lugar, asegúrese de que las aplicaciones pueden ejecutarse en la nueva versión con ese procedimiento. Después, puede actualizar manualmente el clúster de base de datos de Aurora PostgreSQL a la nueva versión.

El proceso de actualización implica la posibilidad de una breve interrupción cuando todas las instancias del clúster se actualicen a la nueva versión. El proceso de planificación preliminar también lleva tiempo. Le recomendamos que realice siempre las tareas de actualización durante el período de mantenimiento del clúster o cuando las operaciones sean mínimas. Para obtener más información, consulte Actualización a una versión principal.

> **Note**
>
> Tanto las actualizaciones de versión secundarias como las actualizaciones de versión principales podrían conllevar interrupciones breves. Por este motivo, le recomendamos que realice o programe actualizaciones durante el período de mantenimiento o durante otros períodos de poca utilización.

En ocasiones, los clústeres de base de datos de Aurora PostgreSQL requieren actualizaciones del sistema operativo. Estas actualizaciones a veces pueden incluir una versión más reciente de la biblioteca glibc. Durante estas actualizaciones, le recomendamos que siga las directrices que se describen en Intercalaciones admitidas en Aurora PostgreSQL.

Obtención de una lista de versiones disponibles en su Región de AWS

Puede obtener una lista de todas las versiones de motor disponibles como destinos de actualización para su clúster de base de datos de Aurora PostgreSQL mediante una consulta de su Región de AWS con el comando describe-db-engine-versions de la AWS CLI, como se indica a continuación.

Para Linux, macOS o Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Para Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Por ejemplo, para identificar los destinos de actualización válidos para un clúster de bases de datos de Aurora PostgreSQL versión 12.10, ejecute el siguiente comando de la:AWS CLI

Para Linux, macOS o Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version 12.10 \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Para Windows:

```
aws rds describe-db-engine-versions ^
```

```
  --engine aurora-postgresql ^
  --engine-version 12.10 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

En la siguiente tabla, encontrará los destinos de actualización de versiones principales y secundarias para distintas versiones de base de datos de Aurora PostgreSQL. Para mantener la compatibilidad, no todas las versiones se ofrecen como destinos de actualización. Aurora PostgreSQL ofrece nuevas características y correcciones de errores en todas las versiones secundarias que aparecen cada tres meses. Para obtener información sobre las actualizaciones secundarias de Aurora PostgreSQL, consulte Release Notes for Aurora PostgreSQL.

Versión de origen actual	Destinos de actualización
17.6	Ninguno
17.5	17.6
17.4	17.6, 17.5, 17.4
16.10	Ninguno
16.9	17.6, 17.5 16.10
16.8	17.6, 17.5, 17.4 16.10, 16.9
16.6	17.6, 17.5, 17.4 16.10, 16.9, 16.8
16.4	17.6, 17.5, 17.4 16.10, 16.9, 16.8, 16.6
16.3	17.6, 17.5, 17.4 16.10, 16.9, 16.8, 16.6, 16.4

Versión de origen actual	Destinos de actualización
16.2	17.6, 17.5, 17.4 16.10, 16.9, 16.8 16.6, 16.4, 16.3
16.1	17.6, 17.5, 17.4 16.10, 16.9, 16.8, 16.6, 16.4, 16.3, 16.2
15.13	17.6 16.10 15.14
15.12	17.6 16.10 15.14
15.1	17.6 16.10 15.14
15.8	17.6 16.10 15.14
15.7	17.6 16.10 15.14

Versión de origen actual	Destinos de actualización
15.6	17.6 16.10 15.14
15.5	17.6 16.10 15.14
15.4	17.6 16.10 15.14
15.3	17.6 16.10 15.14
15.2	17.6 16.10 15.14
14.19	Ninguno
14.18	17.6 16.10 15.14 14.19

Versión de origen actual	Destinos de actualización
14.17	17.6 16.10 15.14 14.19
14.15	17.6 16.10 15.14 14.19
14.13	17.6 16.10 15.14 14.19
14.12	17.6 16.10 15.14 14.19
14.11	17.6 16.10 15.14 14.19

Versión de origen actual	Destinos de actualización
14.10	17.6 16.10 15.14 14.19
14.9	17.6 16.10 15.14 14.19
14.8	17.6 16.10 15.14 14.19
14.7	17.6 16.10 15.14 14.19
14.6	17.6 16.10 15.14 14.19

Versión de origen actual	Destinos de actualización
14.5	17.6 16.10 15.14 14.19
14.4	17.6 16.10 15.14 14.19
14.3	17.6 16.10 15.14 14.19
13.22	Ninguno
13.21	17.6 16.10 15.14 14.19 13.22

Versión de origen actual	Destinos de actualización
13.20	17.6 16.10 15.14 14.19 13.22
13.18	17.6 16.10 15.14 14.19 13.22
13.16	17.6 16.10 15.14 14.19 13.22
13.15	17.6 16.10 15.14 14.19 13.22

Versión de origen actual	Destinos de actualización
13.14	17.6 16.10 15.14 14.19 13.22
13.13	17.6 16.10 15.14 14.19 13.22
1312	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13
13.11	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12

Versión de origen actual	Destinos de actualización
13.10	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11
13.9	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.6 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13,13.12, 13.11, 13.10
13.8	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.6, 14.5 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9

Versión de origen actual	Destinos de actualización
13.7	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.6, 14.5, 14.4, 14.3 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9, 13.8
12.22	17.5, 17.4 16.9, 16.8, 16.6 15.13, 15.12, 15.10 14.18, 14.17, 14.15 13.21, 13.20, 13.18
12.20	17.5, 17.4 16.9, 16.8, 16.6, 16.4 15.13, 15.12, 15.10, 15.8 14.15, 14.13 13.21, 13.20, 13.18, 13.16 12.22

Versión de origen actual	Destinos de actualización
12.19	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3 15.13, 15.12, 15.10, 15.8, 15.7 14.18, 14.17, 14.15, 14.13, 14.12 13.21, 13.20, 13.18, 13.16, 13.15 12.22, 12.20
12.18	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2 15.13, 15.12, 15.10, 15.8, 15.7, 15.6 14.18, 14.17, 14.15, 14.13, 14.12, 14.11 13.21, 13.20, 13.18, 13.16, 13.15, 13.14 12.22, 12.20, 12.19
12.17	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13 12.22, 12.20, 12.19, 12.18

Versión de origen actual	Destinos de actualización
12.16	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12 12.22, 12.20, 12.19, 12.18, 12.17
12.15	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11 12.22, 12.20, 12.19, 12.18, 12.17, 12.16
12.14	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10 12.22, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15

Versión de origen actual	Destinos de actualización
12.13	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.6 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9 12.22, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15, 12.14
12.12	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.6, 14.5 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9, 13.8 12.22, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15, 12.14, 12.13
12.11	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.5, 14.4, 14.3 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9, 13.8, 13.7 12.22, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15, 12.14, 12.13, 12.12

Versión de origen actual	Destinos de actualización
12.9	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7 13.21, 13.20, 13.18, 13.16, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9, 13.8, 13.7 12.22, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15, 12.14, 12.13, 12.12, 12.11
11.21	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12 12.22, 12.20, 12.19, 12.18, 12.17, 12.16

Versión de origen actual	Destinos de actualización
11.9	17.5, 17.4 16.9, 16.8, 16.6, 16.4, 16.3, 16.2, 16.1 15.13, 15.12, 15.10, 15.8, 15.7, 15.6, 15.5, 15.4, 15.3, 15.2 14.18, 14.17, 14.15, 14.13, 14.12, 14.11, 14.10, 14.9, 14.8, 14.7, 14.6 13.21, 13.20, 13.18, 13.16, 13.15, 13.14, 13.13, 13.12, 13.11, 13.10, 13.9 12.22, 12.20, 12.19, 12.18, 12.17, 12.16, 12.15, 12.14, 12.13, 12.12, 12.11, 12.09 11.21

Para cualquier versión que esté considerando, compruebe siempre la disponibilidad de la clase de instancia de base de datos del clúster. Por ejemplo, `db.r4` no admite Aurora PostgreSQL 13. Si su clúster de base de datos de Aurora PostgreSQL utiliza actualmente una clase de instancia db.r4, debe pasar a db.r5 antes de intentar la actualización. Para obtener más información sobre las clases de instancias de bases de datos, incluidas las basadas en Graviton2 y las basadas en Intel, consulte Clases de instancia de base de datos de Amazon Aurora.

Actualización a una versión principal

Las actualizaciones de versión principales podrían contener cambios realizados en la base de datos que no son compatibles con las versiones anteriores de la base de datos. La nueva funcionalidad de una nueva versión puede provocar que sus aplicaciones existentes dejen de funcionar correctamente. Para evitar problemas, Amazon Aurora no aplica automáticamente actualizaciones de versión principales. En su lugar, le recomendamos que planifique cuidadosamente la actualización de una versión principal mediante los siguientes pasos:

1. Elija la versión principal que desee de la lista de objetivos disponibles de los mostrados para su versión en la tabla. Puede obtener una lista precisa de las versiones disponibles en su Región de AWS correspondientes a la versión actual mediante la AWS CLI. Para obtener más información, consulte Obtención de una lista de versiones disponibles en su Región de AWS.

2. Compruebe que las aplicaciones funcionan según lo esperado en una implementación de prueba de la nueva versión. Para obtener información sobre el proceso completo, consulte Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal.
3. Después de comprobar que las aplicaciones funcionan según lo previsto en la implementación de prueba, puede actualizar el clúster. Para obtener más información, consulte Actualización del motor de Aurora PostgreSQL a una nueva versión principal.

> **Note**
>
> Puede realizar una actualización de la versión principal de las versiones basadas en Babelfish para Aurora PostgreSQL 13 a partir de la versión 13.6 a las versiones basadas en Aurora PostgreSQL 14 a partir de la versión 14.6. Babelfish para Aurora PostgreSQL 13.4 y 13.5 no admite la actualización de versiones principales.

Puede obtener una lista de las versiones de motor disponibles como destinos de actualización de versiones principales para su clúster de base de datos de Aurora PostgreSQL mediante una consulta de su Región de AWS con el comando describe-db-engine-versions de la AWS CLI, como se indica a continuación.

Para Linux, macOS o Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}' \
  --output text
```

Para Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}" ^
  --output text
```

En algunos casos, la versión a la que desea actualizar no es un destino para la versión actual. En estos casos, utilice la información de la versions table para realizar actualizaciones de versión secundarias hasta que el clúster esté en una versión que tenga el destino elegido en su fila de destinos.

Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal

Cada nueva versión principal incluye mejoras en el optimizador de consultas que se han diseñado para mejorar el rendimiento. Sin embargo, la carga de trabajo puede incluir consultas que den lugar a un plan que tiene un peor rendimiento en la nueva versión. Por eso le recomendamos que pruebe y revise el rendimiento antes de actualizar en producción. Puede administrar la estabilidad del plan de consultas en todas las versiones mediante la extensión Query Plan Management (QPM), como se detalla en Garantizar la estabilidad del plan después de una actualización a una versión principal.

Antes de actualizar los clústeres de base de datos de Aurora PostgreSQL de producción a una nueva versión principal, le recomendamos que pruebe la actualización para verificar que todas sus aplicaciones funcionan correctamente:

1. Tenga preparado un grupo de parámetros compatible con la versión.

   Si utiliza una instancia de base de datos personalizada o un grupo de parámetros de clúster de base de datos, puede elegir una de estas dos opciones:

   a. Especifique la instancia de base de datos predeterminada, el grupo de parámetros del clúster de base de datos o ambos para la nueva versión del motor de base de datos.
   b. Cree su propio grupo de parámetros personalizado para la nueva versión del motor de base de datos.

   Si asocia un nuevo grupo de parámetros de instancia de base de datos o clúster de base de datos como parte de la solicitud de actualización, asegúrese de reiniciar la base de datos una vez finalizada la actualización para aplicar los parámetros. Si una instancia de base de datos tiene que reiniciarse para aplicar los cambios del grupo de parámetros, el estado del grupo de parámetros de la instancia mostrará `pending-reboot`. Puede ver el estado del grupo de parámetros de una instancia en la consola o mediante un comando de la CLI como describe-db-instances o describe-db-clústers.
2. Compruebe si hay bases de datos no válidas y elimine las que existan.

   La columna `datconnlimit` del catálogo de `pg_database` incluye un valor de `-2` para marcar como no válidas las bases de datos que se interrumpieron durante una operación `DROP DATABASE`. Utilice la siguiente consulta para comprobar si existen bases de datos no válidas.

```
SELECT
    datname
FROM
    pg_database
WHERE
    datconnlimit = - 2;
```

Si la consulta devuelve nombres de bases de datos, estas bases de datos no son válidas. Utilice la instrucción `DROP DATABASE invalid_db_name` para eliminar las bases de datos no válidas. Puede utilizar la siguiente instrucción dinámica para eliminar todas las bases de datos no válidas.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

3. Compruebe si hay algún uso no admitido:

   - Confirme o revierta todas las transacciones preparadas abiertas antes de intentar una actualización. Puede usar la siguiente consulta para comprobar que no haya transacciones preparadas abiertas en la instancia.

     ```
     SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
     ```

   - Elimine todos los usos de los tipos de datos reg* antes de intentar realizar una actualización. Salvo en el caso de `regtype` y `regclass`, no se puede actualizar los tipos de datos reg*. La utilidad pg_upgrade (que Amazon Aurora usa para realizar la actualización.) no puede hacer persistir este tipo de datos. Para obtener más información sobre esta utilidad, consulte pg_upgrade en la documentación de PostgreSQL.

     Para comprobar que no se usan tipos de datos reg* incompatibles, utilice la consulta siguiente en cada base de datos.

     ```
     SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
      pg_catalog.pg_attribute a
       WHERE c.oid = a.attrelid
           AND NOT a.attisdropped
           AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
     ```

```
                        'pg_catalog.regprocedure'::pg_catalog.regtype,
                        'pg_catalog.regoper'::pg_catalog.regtype,
                        'pg_catalog.regoperator'::pg_catalog.regtype,
                        'pg_catalog.regconfig'::pg_catalog.regtype,
                        'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

- Si va a actualizar un clúster de base de datos de Aurora PostgreSQL 10.18 o una versión posterior y tiene instalada la extensión `pgRouting`, elimínela antes de actualizar a la versión 12.4 o posterior.

  Si va a actualizar una versión de Aurora PostgreSQL 10.x que tiene instalada la extensión `pg_repack` versión 1.4.3, elimine la extensión antes de actualizar a una versión superior.

4. Compruebe las bases de datos template1 y template0.

   Para que la actualización se realice correctamente, las bases de datos de plantilla 1 y plantilla 0 deben existir y figurar como plantilla. Para comprobarlo, utilice el siguiente comando:

```
SELECT datname, datistemplate FROM pg_database;

datname    | datistemplate
-----------+---------------
template0  | t
rdsadmin   | f
template1  | t
postgres   | f
```

   En el resultado del comando, el valor `datistemplate` de las bases de datos template1 y template0 debe ser `t`.

5. Elimine las ranuras de replicación lógica.

   El proceso de actualización no puede continuar si el clúster de base de datos de Aurora PostgreSQL utiliza ranuras de replicación lógica. Las ranuras de replicación lógica se utilizan habitualmente para tareas de migración de datos a corto plazo, como migrar datos con AWS DMS o replicar tablas de la base de datos en lagos de datos, las herramientas de inteligencia empresarial (BI) y otros destinos. Antes de actualizar, asegúrese de conocer el propósito de las ranuras de replicación lógica existentes y confirme que es correcto eliminarlos. Puede comprobar las ranuras de replicación lógica mediante la siguiente consulta:

```
SELECT * FROM pg_replication_slots;
```

Si las ranuras de replicación lógica se siguen utilizando, no debe eliminarlos y no puede continuar con la actualización. Sin embargo, si no se necesitan las ranuras de replicación lógica, puede eliminarlos con la siguiente SQL:

```
SELECT pg_drop_replication_slot(slot_name);
```

Los escenarios de replicación lógica que utilizan la extensión `pglogical` también deben tener espacios eliminados del nodo de publicación para que la actualización de la versión principal se realice correctamente en dicho nodo. Sin embargo, puede reiniciar el proceso de replicación desde el nodo de suscriptor después de la actualización. Para obtener más información, consulte Restablecimiento de la replicación lógica después de una actualización principal.

6. Realice una copia de seguridad.

   El proceso de actualización crea una instantánea del clúster de base de datos durante la actualización. Si también desea realizar una copia de seguridad manual antes del proceso de actualización, consulte Creación de una instantánea de clúster de base de datos para obtener más información.

7. Actualice ciertas extensiones a la última versión disponible antes de realizar la actualización de la versión principal. Las extensiones que se van a actualizar incluyen las siguientes:
   - `pgRouting`
   - `postgis_raster`
   - `postgis_tiger_geocoder`
   - `postgis_topology`
   - `address_standardizer`
   - `address_standardizer_data_us`

   Ejecute el comando siguiente para cada extensión instalada actualmente.

   ```
   ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
   ```

   Para obtener más información, consulte Actualización de las extensiones de PostgreSQL. Para obtener más información acerca de la actualización de PostGIS, consulte Paso 6: Actualice la extensión de PostGIS.

8. Si va a actualizar a la versión 11.x, elimine las extensiones que no admita antes de realizar la actualización de la versión principal. Las extensiones que se van a eliminar incluyen:
   - `chkpass`
   - `tsearch2`
9. Elimine los tipos de datos `unknown`, en función de la versión de destino.

   La versión 10 de PostgreSQL no admite el tipo de datos `unknown`. Si una base de datos de la versión 9.6 usa el tipo de datos `unknown`, una actualización a la versión 10 muestra un mensaje de error como el siguiente.

   ```
   Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
   The instance could not be upgraded because the 'unknown' data type is used in user
    tables.
   Please remove all usages of the 'unknown' data type and try again."
   ```

   Para encontrar el tipo de datos `unknown` en la base de datos y así poder eliminar dichas columnas o cambiarlas por tipos de datos compatibles, utilice el siguiente código SQL para cada base de datos.

   ```
   SELECT n.nspname, c.relname, a.attname
       FROM pg_catalog.pg_class c,
       pg_catalog.pg_namespace n,
       pg_catalog.pg_attribute a
       WHERE c.oid = a.attrelid AND NOT a.attisdropped AND
       a.atttypid = 'pg_catalog.unknown'::pg_catalog.regtype AND
       c.relkind IN ('r','m','c') AND
       c.relnamespace = n.oid AND
       n.nspname !~ '^pg_temp_' AND
       n.nspname !~ '^pg_toast_temp_' AND n.nspname NOT IN ('pg_catalog',
    'information_schema');
   ```

10. Realice una actualización de prueba.

    Es muy recomendable probar una actualización de versión principal en un duplicado de la base de datos de producción antes de intentar llevarla a cabo en la base de datos de producción. Puede supervisar los planes de ejecución de la instancia de prueba duplicada para detectar posibles regresiones del plan de ejecución y evaluar su rendimiento. Para crear una instancia de prueba duplicada, puede restaurar su base de datos a partir de una instantánea reciente o clonar la base

de datos. Para obtener más información, consulte Restauración a partir de una instantánea o Clonación de un volumen de clúster de base de datos de Amazon Aurora.

Para obtener más información, consulte Actualización del motor de Aurora PostgreSQL a una nueva versión principal.

11 Actualice su instancia de producción.

Si la actualización de la versión principal de prueba se ha completado correctamente, debería poder actualizar su base de datos de producción con confianza. Para obtener más información, consulte Actualización del motor de Aurora PostgreSQL a una nueva versión principal.

> **Note**
>
> Durante el proceso de actualización, Aurora PostgreSQL toma una instantánea del clúster de base de datos si el periodo de retención de copia de seguridad es mayor que 0. Durante este proceso no puede realizar una restauración del clúster en un momento determinado. Más adelante, podrá realizar una restauración en un momento determinado anterior al inicio de la actualización y posterior a la finalización de la instantánea automática de la instancia. Sin embargo, no se puede restaurar a una versión secundaria anterior.

Para obtener información acerca de una actualización en curso, puede usar Amazon RDS para ver dos registros que la utilidad pg_upgrade produce. Estos son `pg_upgrade_internal.log` y `pg_upgrade_server.log`. Amazon Aurora agrega una marca temporal al nombre de archivo de estos registros. Puede ver estos registros como cualquier otro registro. Para obtener más información, consulte Supervisión de archivos de registro de Amazon Aurora.

12 Actualice las extensiones de PostgreSQL. El proceso de actualización de PostgreSQL no actualiza ninguna extensión de PostgreSQL. Para obtener más información, consulte Actualización de las extensiones de PostgreSQL.

Recomendaciones posteriores a la actualización

Después de completar una actualización de versión principal, se recomienda lo siguiente:

- Ejecute la operación `ANALYZE` para actualizar la tabla `pg_statistic`. Debe hacerlo para cada base de datos en todas las instancias de base de datos de PostgreSQL. Las estadísticas del

optimizador no se transfieren durante una actualización de la versión principal, por lo que debe regenerar todas las estadísticas para evitar problemas de rendimiento. Ejecute el comando sin parámetros para generar estadísticas para todas las tablas normales de la base de datos actual, de la siguiente manera:

```
ANALYZE VERBOSE;
```

La marca `VERBOSE` es opcional, pero su uso muestra el progreso. Para obtener más información, consulte ANALYZE en la documentación de PostgreSQL.

> **Note**
>
> Ejecute ANALYZE en el sistema después de la actualización para evitar problemas de rendimiento.

- Si actualizó a la versión 10 de PostgreSQL, ejecute `REINDEX` en los índices hash que tenga. Los índices hash se cambiaron en la versión 10 y se deben reconstruir. Para localizar índices hash no válidos, ejecute el siguiente SQL para cada base de datos que contenga índices hash.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

- Le recomendamos que pruebe su aplicación en la base de datos actualizada con una carga de trabajo similar para comprobar que todo funciona del modo previsto. Cuando haya comprobado la actualización, podrá eliminar esta instancia de prueba.

Actualización del motor de Aurora PostgreSQL a una nueva versión principal

Cuando inicia el proceso de actualización a una nueva versión principal, Aurora PostgreSQL toma una instantánea del clúster de base de datos de Aurora antes de realizar cualquier cambio en el clúster. Esta instantánea se crea solo para las actualizaciones de versión principales, no para las actualizaciones de versión secundarias. Cuando el proceso de actualización se complete, podrá encontrar esta instantánea entre las instantáneas manuales que aparecen en Snapshots (Instantáneas) en la consola de RDS. El nombre de la instantánea incluye `preupgrade` como prefijo

el nombre de su clúster de base de datos de Aurora PostgreSQL, la versión de origen, la versión de destino y la fecha y la marca de tiempo, como se muestra en el siguiente ejemplo.

```
preupgrade-docs-lab-apg-global-db-12-8-to-13-6-2022-05-19-00-19
```

Una vez completada la actualización, puede utilizar la instantánea que Aurora creó y almacenó en su lista de instantáneas manuales para restaurar el clúster de base de datos a su versión anterior, si es necesario.

> **Tip**
>
> En general, las instantáneas proporcionan muchas maneras de restaurar el clúster de base de datos de Aurora a varios momentos. Para obtener más información, consulte Restauración de una instantánea de clúster de base de datos y Restauración de un clúster de base de dato a un momento indicado. Sin embargo, Aurora PostgreSQL no admite el uso de instantáneas para restaurar a una versión secundaria anterior.

Durante el proceso de actualización de la versión principal, Aurora asigna un volumen y clona el clúster de base de datos de Aurora PostgreSQL de origen. Si se produce un error en la actualización por cualquier motivo, Aurora PostgreSQL utiliza el clon para revertir la actualización. Después de asignar más de 15 clones de un volumen de origen, los clones posteriores se convierten en copias completas y tardan más tiempo. Esto puede provocar que el proceso de actualización también sea más largo. Si Aurora PostgreSQL revierte la actualización, tenga en cuenta lo siguiente:

- Puede ver las entradas y métricas de facturación tanto para el volumen original como para el volumen clonado asignado durante la actualización. Aurora PostgreSQL limpia el volumen adicional después de que el periodo de retención de copia de seguridad del clúster supere el momento de la actualización.
- La siguiente copia instantánea entre regiones de este clúster será una copia completa en lugar de una copia incremental.

Para actualizar de forma segura las instancias de base de datos que componen su clúster, Aurora PostgreSQL usa la utilidad pg_upgrade. Una vez completada la actualización del escritor, cada instancia del lector experimenta una breve interrupción mientras se actualiza a la nueva versión principal. Para obtener más información sobre esta utilidad de PostgreSQL, consulte pg_upgrade en la documentación de PostgreSQL.

Puede actualizar el clúster de base de datos de Aurora PostgreSQL a una versión nueva mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para actualizar la versión del motor de un clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija el clúster de base de datos que desea actualizar.
3. Elija Modify (Modificar). Aparece la página Modify DB clúster (Modificar clúster de base de datos).
4. En Engine version (Versión del motor), elija la nueva versión.
5. Elija Continue (Continuar) y consulte el resumen de las modificaciones.
6. Para aplicar los cambios inmediatamente, elija Apply immediately. Si se selecciona esta opción, puede producirse una interrupción en algunos casos. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify clúster (Modificar clúster) para guardarlos.

   O bien, elija Back (Atrás) para editar los cambios o Cancel (Cancelar) para cancelarlos.

AWS CLI

Para actualizar la versión del motor de un clúster de base de datos, utilice el comando modify-db-clúster de la AWS CLI. Especifique los siguientes parámetros:

- `--db-cluster-identifier`: el nombre del clúster de bases de datos.
- `--engine-version`: número de versión del motor de base de datos al que se va a actualizar. Para obtener información sobre versiones de motores válidas, utilice el comando describe-db-engine-versions de la AWS CLI.
- `--allow-major-version-upgrade`: un indicador requerido cuando el parámetro `--engine-version` es una versión principal diferente de la versión principal actual del clúster de base de datos.
- `--no-apply-immediately`: aplicar los cambios en el siguiente periodo de mantenimiento. Para aplicar los cambios inmediatamente, use `--apply-immediately`.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API de RDS

Para actualizar la versión del motor de un clúster de base de datos, utilice la operación ModifyDBclúster. Especifique los siguientes parámetros:

- `DBClusterIdentifier`: nombre del clúster de base de datos, por ejemplo, *`mydbcluster`*.
- `EngineVersion`: número de versión del motor de base de datos al que se va a actualizar. Para obtener información sobre versiones de motores válidas, utilice la operación DescribeDBEngineVersions.
- `AllowMajorVersionUpgrade`: un indicador requerido cuando el parámetro `EngineVersion` es una versión principal diferente de la versión principal actual del clúster de base de datos.
- `ApplyImmediately`: indica si se deben aplicar los cambios inmediatamente o en el siguiente periodo de mantenimiento. Para aplicar los cambios inmediatamente, establezca el valor en `true`. Para aplicar los cambios en el siguiente periodo de mantenimiento, establezca el valor en `false`.

Actualizaciones importantes para bases de datos globales

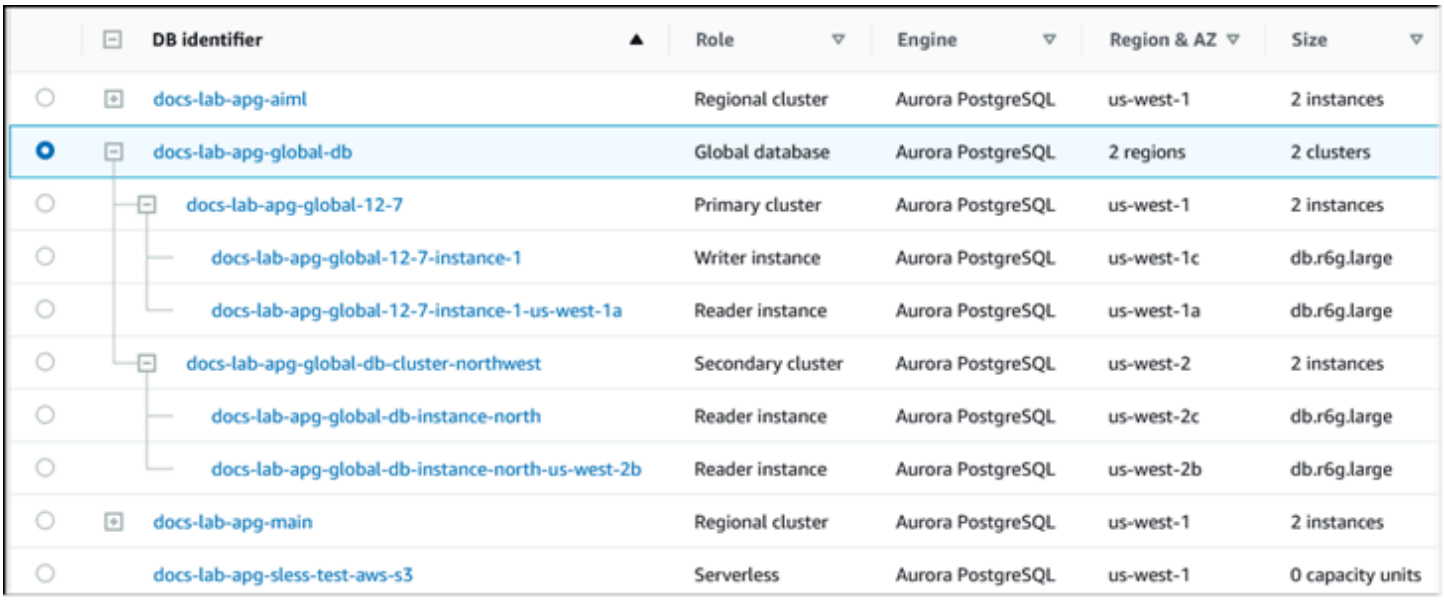
En el caso de un clúster de base de datos global de Aurora, el proceso de actualización se aplica a todos los clústeres de base de datos que componen su base de datos global de Aurora al mismo tiempo. Lo hace para asegurarse de que cada uno ejecuta la misma versión de Aurora PostgreSQL.

También garantiza que cualquier cambio en las tablas del sistema, formatos de archivo de datos, etc., se replican automáticamente en todos los clústeres secundarios.

Para actualizar un clúster de base de datos global a una nueva versión principal de Aurora PostgreSQL, le recomendamos que pruebe las aplicaciones en la versión actualizada, como se detalla en Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal. Asegúrese de preparar el grupo de parámetros de su clúster de base de datos y la configuración del grupo de parámetros de base de datos para cada Región de AWS en su base de datos global de Aurora antes de la actualización como se detalla en step 1. de Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal.

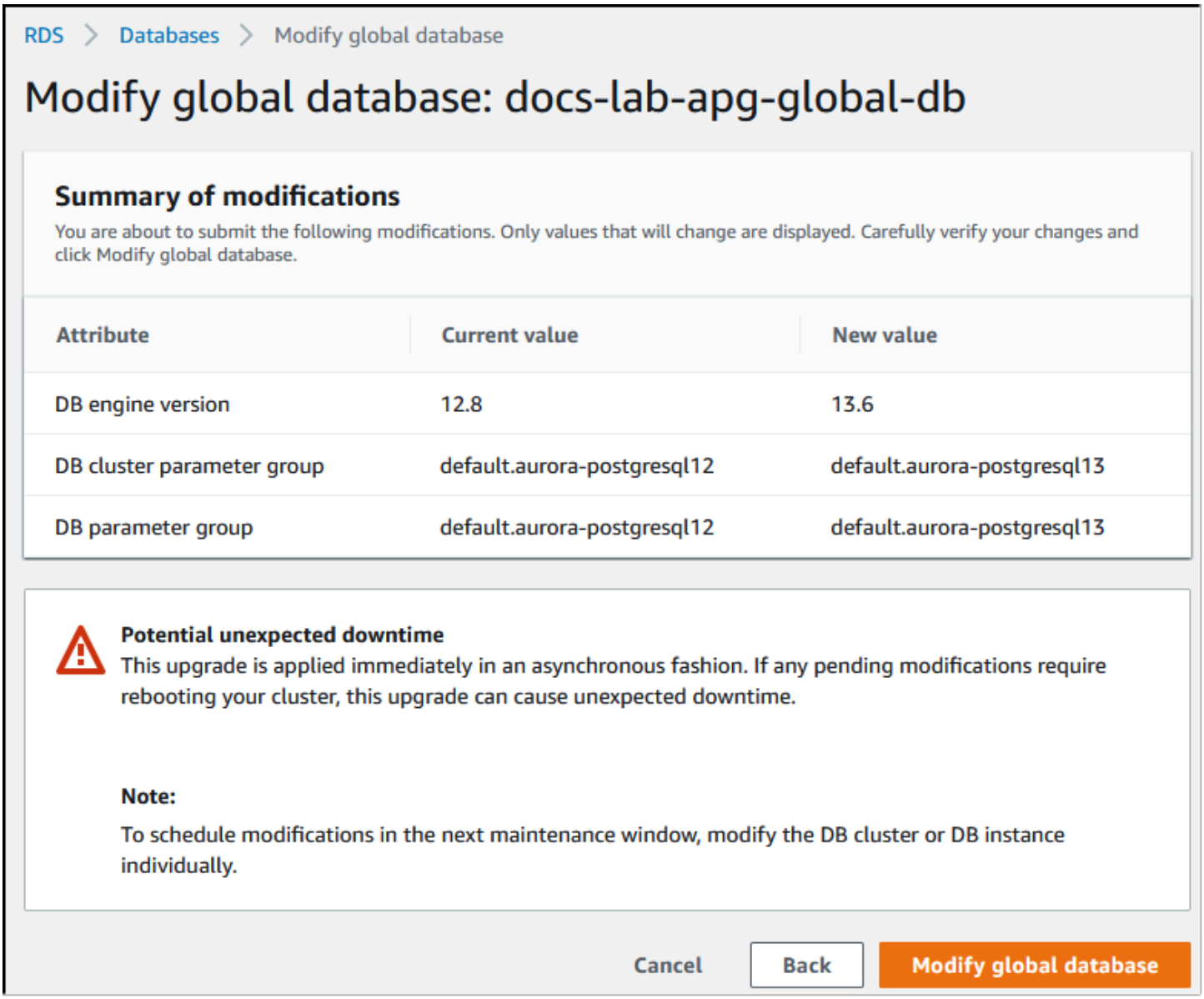
Si el clúster de base de datos global de Aurora PostgreSQL tiene un objetivo de punto de recuperación (RPO) establecido para su parámetro `rds.global_db_rpo`, asegúrese de restablecer el parámetro antes de actualizar. El proceso de actualización de la versión principal no funciona si el RPO está activado. De forma predeterminada, este parámetro está desactivado. Para obtener más información sobre las bases de datos globales de Aurora PostgreSQL y RPO, consulte Administración de RPO para bases de datos globales basadas en Aurora PostgreSQL–.

Si verifica que sus aplicaciones pueden ejecutarse como se espera en la implementación de prueba de la nueva versión, puede iniciar el proceso de actualización. Para ello, consulte Actualización del motor de Aurora PostgreSQL a una nueva versión principal. Asegúrese de elegir el elemento de nivel superior de la lista Databases (Bases de datos) en la consola de RDS, Global database (Base de datos global), como se muestra en la siguiente imagen.



DB identifier	Role	Engine	Region & AZ	Size
docs-lab-apg-aiml	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-db	Global database	Aurora PostgreSQL	2 regions	2 clusters
docs-lab-apg-global-12-7	Primary cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-12-7-instance-1	Writer instance	Aurora PostgreSQL	us-west-1c	db.r6g.large
docs-lab-apg-global-12-7-instance-1-us-west-1a	Reader instance	Aurora PostgreSQL	us-west-1a	db.r6g.large
docs-lab-apg-global-db-cluster-northwest	Secondary cluster	Aurora PostgreSQL	us-west-2	2 instances
docs-lab-apg-global-db-instance-north	Reader instance	Aurora PostgreSQL	us-west-2c	db.r6g.large
docs-lab-apg-global-db-instance-north-us-west-2b	Reader instance	Aurora PostgreSQL	us-west-2b	db.r6g.large
docs-lab-apg-main	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-sless-test-aws-s3	Serverless	Aurora PostgreSQL	us-west-1	0 capacity units

Como con cualquier modificación, puede confirmar que desea que el proceso continúe cuando se le solicite.



En lugar de utilizar la consola, puede iniciar el proceso de actualización mediante la AWS CLI o la API de RDS. Al igual que con la consola, se opera en el clúster de base de datos global de Aurora en lugar de hacerlo en cualquiera de sus componentes, como se indica a continuación:

- Use el comando modify-global-clúster de la AWS CLI para iniciar la actualización de la base de datos global de Aurora mediante la AWS CLI.
- Use la API ModifyGlobalclúster para iniciar la actualización.

Actualización a una versión secundaria

Puede utilizar los métodos siguientes para actualizar la versión secundaria de un clúster de bases de datos o para aplicar parches a un clúster de bases de datos:

Temas

- Antes de realizar una actualización de versión secundaria
- Cómo realizar actualizaciones de versión secundarias y aplicar revisiones
- Actualizaciones de versión secundarias y aplicación de revisiones sin tiempo de inactividad
- Limitaciones de parches sin tiempo de inactividad
- Actualización del motor de Aurora PostgreSQL a una nueva versión secundaria

Antes de realizar una actualización de versión secundaria

Le recomendamos que lleve a cabo las siguientes acciones para reducir el tiempo de inactividad durante una actualización de versión secundaria:

- El mantenimiento del clúster de base de datos Aurora debe realizarse durante un periodo de poco tráfico. Utilice Performance Insights para identificar estos periodos de tiempo y configurar correctamente los plazos de mantenimiento. Para obtener más información sobre Performance Insights, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon RDS. Para obtener más información sobre los periodos de mantenimiento de clústeres de base de datos, consulte Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos.
- Utilice los SDK de AWS que admitan fluctuaciones y retrocesos exponenciales como procedimiento recomendado. Para obtener más información, consulte Exponential Backoff And Jitter.

Cómo realizar actualizaciones de versión secundarias y aplicar revisiones

Las actualizaciones y revisiones de versión secundarias estarán disponibles en Regiones de AWS solo después de rigurosas pruebas. Antes de lanzar actualizaciones y revisiones, Aurora PostgreSQL realiza pruebas para asegurarse de que los problemas de seguridad conocidos, los errores y otros problemas que surgen después del lanzamiento de la versión secundaria de la comunidad no perturben la estabilidad de la flota de Aurora PostgreSQL.

Si activa Habilitar actualización automática de versiones secundarias, Aurora PostgreSQL actualiza periódicamente el clúster de bases de datos durante el periodo de mantenimiento especificado.

Asegúrese de que la opción Habilitar actualización automática de versiones secundarias está activada para todas las instancias del clúster de bases de datos de Aurora PostgreSQL. Para obtener información sobre cómo configurar la actualización automática de versiones secundarias y cómo funciona la configuración cuando se aplica a los niveles de clúster e instancia, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora.

Consulte el valor de la opción Habilitar actualización automática de versiones secundarias para todos los clústeres de bases de datos de Aurora PostgreSQL mediante el comando describe-db-instances de la AWS CLI del siguiente modo.

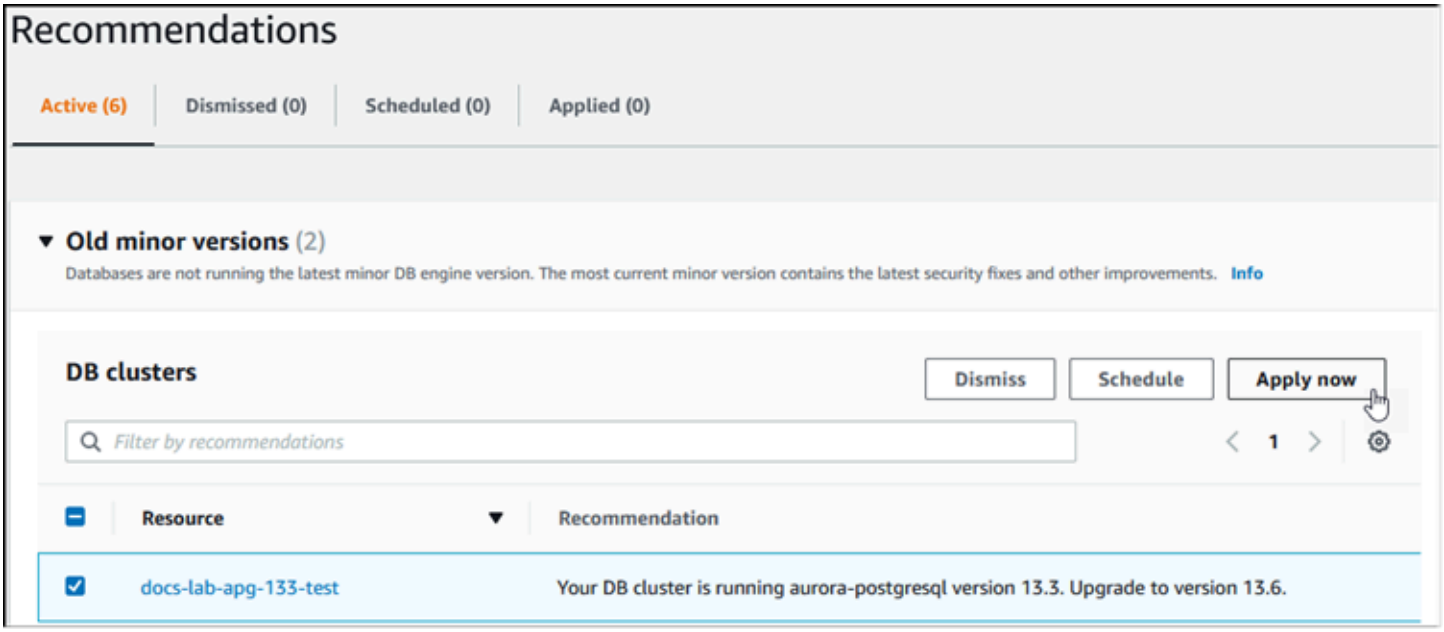
```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Esta consulta devuelve una lista de todos los clústeres de base de datos de Aurora y sus instancias con un valor `true` o `false` para el estado de la configuración `AutoMinorVersionUpgrade`. En el comando se supone que tiene configurada la AWS CLI para dirigirse a Región de AWS predeterminada.

Para obtener más información sobre la opción AmVU y cómo modificar el clúster de base de datos de Aurora para utilizarla, consulte Actualizaciones de versiones secundarias automáticas para clústeres de base de datos de Aurora.

Puede actualizar sus clústeres de base de datos de Aurora PostgreSQL a nuevas versiones secundarias, ya sea mediante la respuesta a las tareas de mantenimiento o la modificación del clúster para utilizar la nueva versión.

Puede identificar cualquier actualización o revisión disponible para sus clústeres de base de datos de Aurora PostgreSQL si utiliza la consola de RDS y abre el menú Recommendations (Recomendaciones). Encontrará una lista de varios problemas de mantenimiento, como Old minor versions (Versiones secundarias anteriores). Según su entorno de producción, puede elegir entre programar la actualización o tomar medidas inmediatas, mediante de la elección de Apply now (Aplicar ahora), como se muestra a continuación.



Para obtener más información sobre cómo mantener un clúster de base de datos de Aurora, incluida la forma de aplicar revisiones y actualizaciones de versión secundarias de forma manual, consulte Mantenimiento de un clúster de base de datos de Amazon Aurora.

Actualizaciones de versión secundarias y aplicación de revisiones sin tiempo de inactividad

La actualización de un clúster de base de datos de Aurora PostgreSQL conlleva la posibilidad de una interrupción. Durante el proceso de actualización, la base de datos se cierra mientras se actualiza. Si comienza la actualización cuando la base de datos está ocupada, perderá todas las conexiones y transacciones que el clúster de base de datos tiene en proceso. Si espera hasta que la base de datos esté inactiva para realizar la actualización, es posible que tenga que esperar mucho tiempo.

La característica de aplicación de revisiones sin tiempo de inactividad (ZDP) mejora el proceso de actualización. Con ZDP, tanto las actualizaciones de versión secundarias como las revisiones pueden aplicarse con un impacto mínimo en el clúster de base de datos de Aurora PostgreSQL. Se utiliza la ZDP al aplicar parches o actualizaciones de versiones secundarias más recientes a las versiones de Aurora PostgreSQL y otras versiones posteriores de estas versiones secundarias y versiones principales más recientes. Es decir, la actualización a nuevas versiones secundarias desde cualquiera de estas versiones en adelante utiliza ZDP.

La siguiente tabla muestra las versiones de Aurora PostgreSQL y las clases de instancia de base de datos donde está disponible ZDP:

Versión	Clases de instancia db.r*	Clases de instancia db.t*	Clases de instancia db.r*	Clases de instancia db.serverless
Versión 10.21 y posteriores	Sí	Sí	Sí	N/A
Versión 11.16 y posteriores	Sí	Sí	Sí	N/A
Versión 11.17 y posteriores	Sí	Sí	Sí	N/A
Versión 12.11 y posteriores	Sí	Sí	Sí	N/A
Versión 12.12 y posteriores	Sí	Sí	Sí	N/A
Versión 13.7 y posteriores	Sí	Sí	Sí	N/A
Versión 13.8 y posteriores	Sí	Sí	Sí	Sí
Versión 14.3 y posteriores	Sí	Sí	Sí	N/A
Versión 14.4 y posteriores	Sí	Sí	Sí	N/A
Versión 14.5 y posteriores	Sí	Sí	Sí	Sí
Versión 15.3 y posteriores	Sí	Sí	Sí	Sí
Versión 16.1 y posteriores	Sí	Sí	Sí	Sí

Durante el proceso de actualización con ZDP, el motor de base de datos busca un punto silencioso para pausar todas las transacciones nuevas. Esta acción protege la base de datos durante las revisiones y las actualizaciones. Para garantizar que las aplicaciones se ejecuten sin problemas con las transacciones pausadas, recomendamos integrar la lógica de reintento en el código. Este enfoque garantiza que el sistema pueda gestionar cualquier breve tiempo de inactividad sin errores y pueda volver a intentar las nuevas transacciones después de la actualización.

Cuando la ZDP se completa correctamente, las sesiones de la aplicación se conservan, excepto las sesiones de conexiones interrumpidas, y el motor de base de datos se reinicia mientras la actualización está en curso. Aunque el reinicio del motor de base de datos puede provocar una bajada temporal del rendimiento, esta suele durar solo unos segundos o, como mucho, un minuto aproximadamente.

En algunos casos, la aplicación de revisiones sin tiempo de inactividad (ZDP) podría no realizarse correctamente. Por ejemplo, los cambios de parámetros que tienen un estado `pending` en su clúster de base de datos de Aurora PostgreSQL o en sus instancias también interfieren con la ZDP.

Puede encontrar métricas y eventos para las operaciones de la ZDP en la página Events (Eventos) de la consola. Los eventos incluyen el inicio de la actualización de la ZDP y la finalización de dicha actualización. En este evento puede encontrar el tiempo que duró el proceso y el número de conexiones conservadas e interrumpidas que se produjeron durante el reinicio. Puede encontrar detalles en el registro de errores de la base de datos.

Limitaciones de parches sin tiempo de inactividad

Las limitaciones siguientes se aplican a la aplicación de parches sin tiempo de inactividad:

- ZDP intenta preservar las conexiones de cliente actuales a la instancia de escritor de Aurora PostgreSQL durante todo el proceso de actualización de Aurora PostgreSQL. Sin embargo, en los siguientes casos, las conexiones se interrumpen para que la ZDP se complete:
  - Hay en curso consultas o transacciones de ejecución prolongada.
  - Hay instrucciones en lenguaje de definición de datos (DDL) en ejecución.
  - Se están utilizando tablas temporales o bloqueos de tabla.
  - Todas las sesiones escuchan en los canales de notificación.
  - Se está utilizando un cursor en estado “WITH HOLD”.
  - Las conexiones TLSv1.1 están en uso. A partir de las versiones de Aurora PostgreSQL posteriores a 16.1, 15.3, 14.8, 13.11, 12.15 y 11.20, ZDP es compatible con las conexiones TLSv1.3.

- La característica ZDP no es compatible con los siguientes casos:
  - Cuando los clústeres de base de datos de Aurora PostgreSQL se configuran como Aurora Serverless v1.
  - Durante la actualización de las instancias del lector de Aurora.
  - Durante la actualización de cualquier instancia del lector de Aurora que forme parte de un clúster de bases de datos globales de Aurora en una región secundaria.
  - Durante los parches y actualizaciones del sistema operativo.

Actualización del motor de Aurora PostgreSQL a una nueva versión secundaria

Puede actualizar el clúster de base de datos de Aurora PostgreSQL a una versión secundaria nueva mediante la consola, la AWS CLI o la API de RDS. Antes de realizar la actualización, recomendamos que siga las mismas prácticas recomendadas que para las actualizaciones de versión. Igual que con las nuevas versiones principales, las nuevas versiones secundarias también pueden incluir mejoras del optimizador, como correcciones, que pueden provocar regresiones en el plan de consultas. Para garantizar la estabilidad del plan, recomendamos que utilice la extensión Query Plan Management (QPM), tal como se detalla en Garantizar la estabilidad del plan después de una actualización a una versión principal.

Consola

Para actualizar la versión del motor del clúster de base de datos de Aurora PostgreSQL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija el clúster de base de datos que desea actualizar.
3. Elija Modify (Modificar). Aparece la página Modify DB clúster (Modificar clúster de base de datos).
4. En Engine version (Versión del motor), elija la nueva versión.
5. Elija Continue (Continuar) y consulte el resumen de las modificaciones.
6. Para aplicar los cambios inmediatamente, elija Apply immediately. Si se selecciona esta opción, puede producirse una interrupción en algunos casos. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modify clúster (Modificar clúster) para guardarlos.

O bien, elija Back (Atrás) para editar los cambios o Cancel (Cancelar) para cancelarlos.

AWS CLI

Para actualizar la versión del motor de un clúster de base de datos, utilice el comando modify-db-clúster de la AWS CLI con los siguientes parámetros:

- `--db-cluster-identifier`: nombre del clúster de base de datos de Aurora PostgreSQL.
- `--engine-version`: número de versión del motor de base de datos al que se va a actualizar. Para obtener información sobre versiones de motores válidas, utilice el comando describe-db-engine-versions de la AWS CLI.
- `--no-apply-immediately`: aplicar los cambios en el siguiente periodo de mantenimiento. Para aplicar los cambios inmediatamente, use `--apply-immediately` en su lugar.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --no-apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --no-apply-immediately
```

API de RDS

Para actualizar la versión del motor de un clúster de base de datos, utilice la operación ModifyDBclúster. Especifique los siguientes parámetros:

- `DBClusterIdentifier`: nombre del clúster de base de datos, por ejemplo, *mydbcluster*.
- `EngineVersion`: número de versión del motor de base de datos al que se va a actualizar. Para obtener información sobre versiones de motores válidas, utilice la operación DescribeDBEngineVersions.

- `ApplyImmediately`: indica si se deben aplicar los cambios inmediatamente o en el siguiente periodo de mantenimiento. Para aplicar los cambios inmediatamente, establezca el valor en `true`. Para aplicar los cambios en el siguiente periodo de mantenimiento, establezca el valor en `false`.

Actualización de las extensiones de PostgreSQL

Al actualizar el clúster de base de datos de Aurora PostgreSQL a una nueva versión principal o secundaria, no se actualizan al mismo tiempo las extensiones de PostgreSQL. En la mayoría de los casos, debe actualizarla la extensión después de que se complete la actualización de la versión principal o secundaria. No obstante, en algunos casos, se actualiza la extensión antes de actualizar el motor de base de datos de Aurora PostgreSQL. Para obtener más información, consulte list of extensions to update en Prueba de la actualización del clúster de base de datos de producción a una nueva versión principal.

La instalación de las extensiones de PostgreSQL requiere privilegios de `rds_superuser`. Por lo general, un `rds_superuser` delega permisos sobre extensiones específicas a usuarios (roles) relevantes, para facilitar la administración de una extensión determinada. Esto significa que la tarea de actualizar todas las extensiones del clúster de base de datos de Aurora PostgreSQL puede implicar a muchos usuarios diferentes (roles). Tenga esto en cuenta sobre todo si desea automatizar el proceso de actualización mediante el uso de scripts. Para obtener más información sobre los privilegios y roles de PostgreSQL, consulte Seguridad con Amazon Aurora PostgreSQL.

> **Note**
>
> Para obtener información sobre la actualización de la extensión de PostGIS, consulte Administración de datos espaciales con la extensión PostGIS(Paso 6: Actualice la extensión de PostGIS).
> Para actualizar la extensión `pg_repack`, elimínela y, a continuación, cree la nueva versión en la instancia de base de datos actualizada. Para obtener más información, consulte pg_repack installation (Instalación de pg_repack) en la documentación de `pg_repack`.

Para actualizar una extensión después de una actualización del motor, utilice el comando `ALTER EXTENSION UPDATE`.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

Para enumerar las extensiones instaladas actualmente, utilice el catálogo pg_extension de PostgreSQL en el siguiente comando.

```
SELECT * FROM pg_extension;
```

Para ver una lista de las versiones específicas de la extensión que están disponibles para su instalación, utilice la visualización pg_available_extension_versions de PostgreSQL en el siguiente comando.

```
SELECT * FROM pg_available_extension_versions;
```

Técnica alternativa de actualización azul/verde

En algunas situaciones, su prioridad principal es realizar un cambio inmediato del clúster antiguo a uno actualizado. En tales situaciones, puede utilizar un proceso de varios pasos que ejecuta los clústeres antiguo y nuevo en paralelo. Aquí, replicará los datos del clúster anterior al nuevo hasta que esté listo para que el nuevo clúster asuma el control. Para obtener información, consulte Uso de las implementaciones azul/verde para actualizar las bases de datos.

Uso de una versión de compatibilidad a largo plazo (LTS) de Aurora PostgreSQL

Cada versión nueva de Aurora PostgreSQL sigue estando disponible durante cierta cantidad de tiempo para que la utilice al crear o actualizar un clúster de bases de datos. Después de este período, debe actualizar los clústeres que utilicen dicha versión. Puede actualizar manualmente el clúster antes de que finalice el periodo de soporte, o Aurora puede actualizarlo automáticamente cuando su versión de Aurora MySQL deje de admitirse.

Aurora designa determinadas versiones de Aurora PostgreSQL como versiones con "soporte a largo plazo" (LTS). Los clústeres de base de datos que utilizan versiones LTS pueden mantenerse en la misma versión durante más tiempo y se someten a menos ciclos de actualización que los clústeres que utilizan versiones que no son LTS. Las versiones menores LTS solo incluyen correcciones de errores (a través de versiones de parches) para problemas de estabilidad y seguridad; una versión LTS no incluye nuevas características publicadas después de su presentación.

Una vez al año, los clústeres de base de datos que se ejecutan en una versión secundaria LTS, se parchean a la última versión del parche de la versión LTS. Aplicamos este parche para ayudar a garantizar que se beneficie de las correcciones acumulativas de seguridad y estabilidad. Es posible

que se realice el parche de una versión secundaria LTS con más frecuencia si hay correcciones críticas, por ejemplo, para la seguridad, que deben aplicarse.

> **Note**
>
> Si desea permanecer en una versión secundaria de LTS mientras dure su ciclo de vida, desactive la actualización automática de versiones secundarias para sus instancias de base de datos. Para evitar actualizar automáticamente el clúster de base de datos desde la versión secundaria de LTS, desmarque la casilla Habilitar actualización automática de versiones secundarias en cualquier instancia de base de datos de su clúster de Aurora.

Le recomendamos que actualice a la versión más reciente, en lugar de utilizar la versión LTS, para la mayoría de los clústeres de Aurora PostgreSQL. Al hacerlo se aprovecha Aurora como servicio administrado y le ofrece acceso a las características y correcciones de errores más recientes. Las versiones LTS están destinadas a clústeres con las siguientes características:

- No puede permitirse periodos de inactividad en la aplicación Aurora PostgreSQL para actualizaciones fuera de los raros casos para parches críticos.
- El ciclo de prueba del clúster y las aplicaciones asociadas tardan mucho tiempo para cada actualización al motor de base de datos de Aurora PostgreSQL.
- La versión de base de datos para su clúster de Aurora PostgreSQL tiene todas las características de motor de base de datos y correcciones de errores que necesita su aplicación.

Las versiones LTS actuales para Aurora PostgreSQL son las siguientes:

- PostgreSQL 16.8. Se lanzó el 7 de abril de 2025. Para obtener más información, consulte PostgreSQL 16.8 en las notas de la versión de Aurora PostgreSQL.
- PostgreSQL 15.10. Se lanzó el 27 de diciembre de 2024. Para obtener más información, consulte PostgreSQL 15.10 en las Notas de la versión de Aurora PostgreSQL.
- PostgreSQL 14.6 Se publicó el 20 de enero de 2023. Para obtener más información, consulte PostgreSQL 14.6 en las Notas de la versión de Aurora PostgreSQL.
- PostgreSQL 13.9 Se publicó el 20 de enero de 2023. Para obtener más información, consulte PostgreSQL 13.9 en las Notas de la versión de Aurora PostgreSQL.
- PostgreSQL 12.9 Esta versión se lanzó el 25 de febrero de 2022. Para obtener más información, consulte PostgreSQL 12.9 en las notas de la versión de Aurora PostgreSQL.

- PostgreSQL 11.9 (versión 3.4 de Aurora PostgreSQL) Esta versión se lanzó el 11 de diciembre de 2020. Para obtener más información acerca de esta versión, consulte el tema sobre PostgreSQL 11.9, Aurora PostgreSQL versión 3.4 en las notas de la versión de Aurora PostgreSQL.

Para obtener más información sobre los plazos de soporte y los ciclos de lanzamiento de las versiones LTS, consulte Calendarios de versiones para Aurora PostgreSQL.

Para obtener información sobre cómo identificar las versiones de Aurora y del motor de base de datos, consulte Identificación de las versiones de Amazon Aurora PostgreSQL.

Uso de Base de datos ilimitada de Amazon Aurora PostgreSQL

Base de datos ilimitada de Amazon Aurora PostgreSQL ofrece un escalado horizontal automatizado para procesar millones de transacciones de escritura por segundo y administra petabytes de datos a la vez que mantiene la simplicidad de operar dentro de una única base de datos. Con Base de datos ilimitada de Aurora PostgreSQL, puede centrarse en crear aplicaciones a gran escala sin tener que crear y mantener soluciones complejas para escalar sus datos en varias instancias de bases de datos para respaldar sus cargas de trabajo.

Temas

- Arquitectura de Base de datos ilimitada de Aurora PostgreSQL
- Introducción a Base de datos ilimitada de Aurora PostgreSQL
- Requisitos y consideraciones sobre Base de datos ilimitada de Aurora PostgreSQL
- Requisitos previos para usar Base de datos ilimitada de Aurora PostgreSQL
- Creación de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL
- Uso de los grupos de particiones de base de datos
- Creación de tablas de Base de datos ilimitada de Aurora PostgreSQL
- Carga de datos en Base de datos ilimitada de Aurora PostgreSQL
- Consulta de Base de datos ilimitada de Aurora PostgreSQL
- Administración de Base de datos ilimitada de Aurora PostgreSQL
- Supervisión de Base de datos ilimitada de Aurora PostgreSQL
- Copia de seguridad y restauración de Base de datos ilimitada de Aurora PostgreSQL
- Actualización de Base de datos ilimitada de Amazon Aurora PostgreSQL
- Referencia sobre Base de datos ilimitada de Aurora PostgreSQL

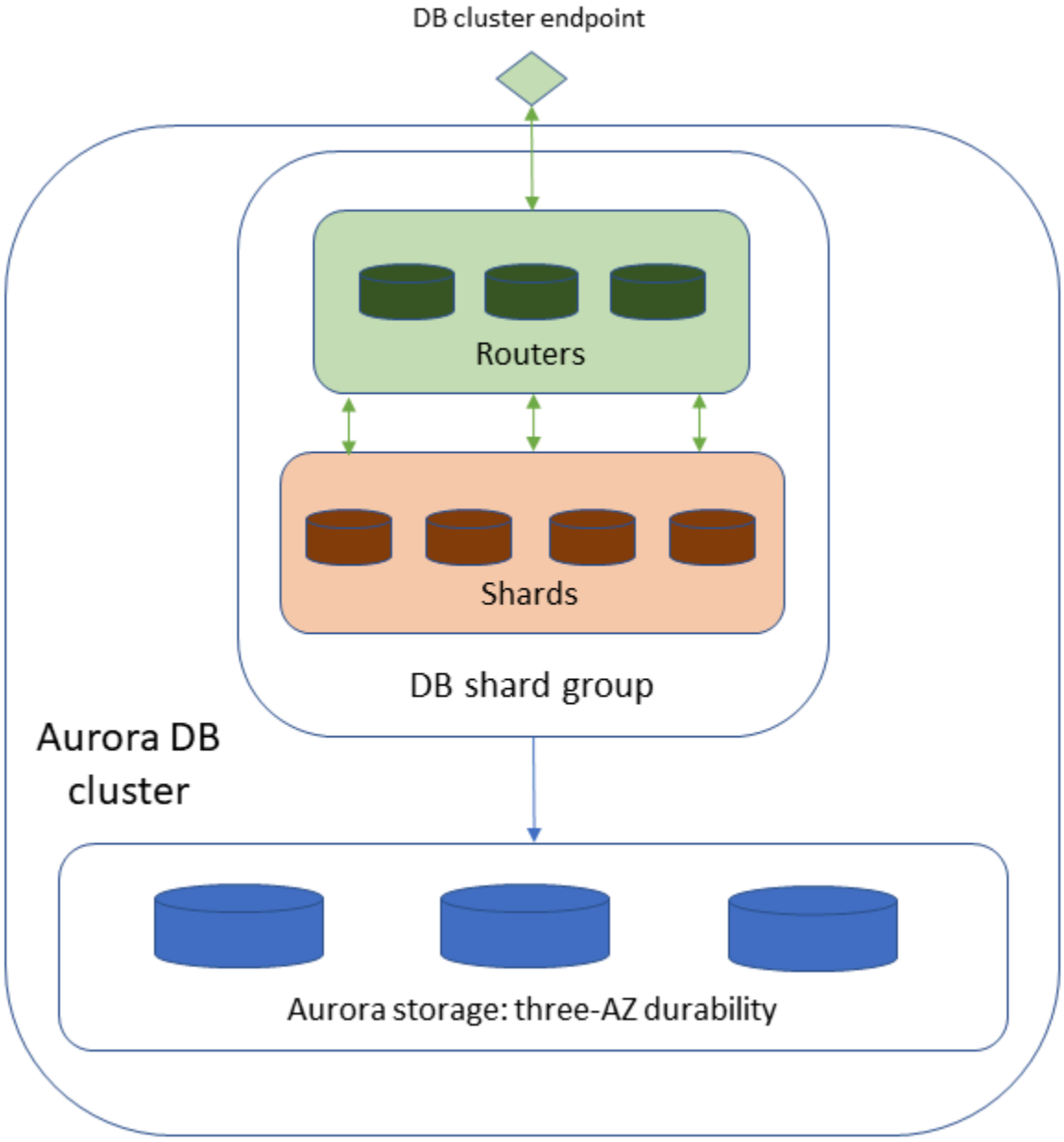
Arquitectura de Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada logra escalar con una arquitectura de dos capas que consta de varios nodos de base de datos. Los nodos son enrutadores o particiones.

- Las particiones son instancias de Base de datos ilimitada de Aurora PostgreSQL, cada una de las cuales almacena un subconjunto de los datos de la base de datos, lo que permite el procesamiento simultáneo para lograr un mayor rendimiento de escritura.
- Los enrutadores administran la naturaleza distribuida de la base de datos y presentan una única imagen de la base de datos a los clientes de la base de datos. Los enrutadores mantienen los metadatos sobre el lugar donde se almacenan los datos, analizan los comandos SQL entrantes y los envían a las particiones. Luego, agregan los datos de las particiones para devolver un único resultado al cliente y administran las transacciones distribuidas para mantener la coherencia en toda la base de datos distribuida.

Base de datos ilimitada de Aurora PostgreSQL se diferencia de los Clústeres de base de datos de Aurora estándar en que tiene un grupo de particiones de base de datos en lugar de una instancia de base de datos de escritura e instancias de base de datos de lectura. Todos los nodos que componen la arquitectura de Base de datos ilimitada están incluidos en el grupo de particiones de base de datos. Las particiones y enrutadores individuales del grupo de particiones de base de datos no están visibles en su Cuenta de AWS. Utilice el punto de conexión del clúster de base de datos para acceder a Base de datos ilimitada.

La siguiente figura muestra la arquitectura de alto nivel de Base de datos ilimitada de Aurora PostgreSQL.



Para obtener más información sobre la arquitectura de Base de datos ilimitada de Aurora PostgreSQL y cómo utilizarla, consulte el vídeo Achieving scale with Amazon Aurora PostgreSQL Limitless Database en el canal AWS Events de YouTube.

Para obtener más información acerca de la arquitectura de un clúster de base de datos de Aurora estándar, consulte Clústeres de base de datos de Amazon Aurora.

Términos clave de Base de datos ilimitada de Aurora PostgreSQL

Grupo de particiones de bases de datos

Es un contenedor para los nodos de Base de datos ilimitada (particiones y enrutadores).

Enrutador

Es un nodo que acepta conexiones SQL de los clientes, envía comandos SQL a las particiones, mantiene la coherencia en todo el sistema y devuelve los resultados a los clientes.

Partición

Es un nodo que almacena un subconjunto de tablas particionadas, copias completas de tablas de referencia y tablas estándar. Acepta consultas de enrutadores, pero los clientes no pueden conectarse directamente a ellos.

Tabla particionada

Es una tabla cuyos datos están divididos en particiones.

Clave de partición

Es una columna o conjunto de columnas de una tabla particionada que se utiliza para determinar la división entre las particiones.

Tablas colocadas

Son dos tablas particionadas que comparten la misma clave de partición y que se declaran como colocadas de forma explícita. Todos los datos del mismo valor de clave de partición se envían a la misma partición.

Tabla de referencia

Es una tabla cuyos datos se copian en su totalidad en cada partición.

Tabla estándar

Es el tipo de tabla predeterminado en Base de datos ilimitada. Puede convertir tablas estándar en tablas particionadas o de referencia.

Todas las tablas estándar se almacenan en la misma partición seleccionada por el sistema, lo que permite realizar uniones entre tablas estándar dentro de una única partición. Sin embargo, las tablas estándar están limitadas por la capacidad máxima de la partición (128 TiB). Esta partición también almacena datos de tablas particionadas y de referencia, por lo que el límite efectivo para las tablas estándar es inferior a 128 TiB.

Tipos de tabla de Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada de Aurora PostgreSQL admite tres tipos de tabla: particionada, de referencia y estándar.

Los datos de las tablas particionadas se distribuyen entre todas las particiones del grupo de particiones de base de datos. Base de datos ilimitada lo hace automáticamente con una clave de partición, que es una columna o un conjunto de columnas que se especifican al dividir la tabla. Todos los datos con el mismo valor de clave de partición se envían a la misma partición. La partición se basa en el hash, no en rangos o listas.

Los siguientes son buenos ejemplos de casos de uso de tablas particionadas:

- La aplicación funciona con un subconjunto de datos distinto.
- El archivo de tabla es muy grande.
- Es posible que la tabla crezca más rápido que otras tablas.

Las tablas particionadas se pueden colocar, lo que significa que comparten la misma clave de partición y que todos los datos de ambas tablas con el mismo valor de clave de partición se envían a la misma partición. Si coloca las tablas y las une con la clave de partición, la unión se puede realizar en una sola partición, ya que todos los datos necesarios están presentes en esa partición.

Las tablas de referencia tienen una copia completa de todos los datos de cada partición del grupo de particiones de base de datos. Las tablas de referencia se suelen utilizar para tablas más pequeñas con un volumen de escritura más bajo, pero aun así es necesario unirlas con frecuencia y no se prestan a la partición. Entre los ejemplos de tablas de referencia se incluyen las tablas de fechas y las tablas de datos geográficos, como estados, ciudades y códigos postales.

Las tablas estándar son el tipo de tabla predeterminado en Base de datos ilimitada de Aurora PostgreSQL. No son tablas distribuidas. Base de datos ilimitada de Aurora PostgreSQL admite uniones entre tablas estándar y tablas estándar, particionadas y de referencia.

Facturación de Base de datos ilimitada de Aurora PostgreSQL

Para obtener más información sobre cómo se cobra por Base de datos ilimitada de Aurora PostgreSQL, consulte Facturación de instancia de base de datos para Aurora.

Para obtener información acerca de los precios de Aurora, consulte la página de precios de Aurora.

Introducción a Base de datos ilimitada de Aurora PostgreSQL

Realice las siguientes acciones para empezar a utilizar Base de datos ilimitada de Aurora PostgreSQL:

1. Cree un clúster de base de datos de Aurora PostgreSQL y un grupo de particiones de base de datos para Base de datos ilimitada. Para obtener más información, consulte Creación de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL.
2. Cree tablas particionadas y de referencia en el grupo de particiones de base de datos. Para obtener más información, consulte Creación de tablas de Base de datos ilimitada de Aurora PostgreSQL.
3. Cambie la capacidad de su grupo de particiones de base de datos, divida las particiones y añada enrutadores. Para obtener más información, consulte Uso de los grupos de particiones de base de datos.
4. Cargue los datos en el grupo de particiones de base de datos. Para obtener más información, consulte Carga de datos en Base de datos ilimitada de Aurora PostgreSQL.
5. Ejecute consultas y otras instrucciones SQL en el grupo de particiones de base de datos. Para obtener más información, consulte Consulta de Base de datos ilimitada de Aurora PostgreSQL.
6. Supervise el rendimiento de Base de datos ilimitada. Para obtener más información, consulte Supervisión de Base de datos ilimitada de Aurora PostgreSQL.

Requisitos y consideraciones sobre Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada de Aurora PostgreSQL tiene los siguientes requisitos y consideraciones.

Temas

- Requisitos de Base de datos ilimitada de Aurora PostgreSQL
- Consideraciones sobre Base de datos ilimitada de Aurora PostgreSQL
- Características no admitidas por Base de datos ilimitada de Aurora PostgreSQL

Requisitos de Base de datos ilimitada de Aurora PostgreSQL

Asegúrese de cumplir estos requisitos para Base de datos ilimitada de Aurora PostgreSQL.

- La base de datos ilimitada de Aurora PostgreSQL está disponible en todas las Regiones de AWS excepto en Asia-Pacífico (Taipéi).

  > **Note**
  >
  > Si crea el clúster de base de datos (DB) ilimitada de Aurora PostgreSQL en Este de EE. UU. (Norte de Virginia), no incluya la zona de disponibilidad (AZ) `us-east-1e` en el grupo de subredes de la base de datos. Debido a las limitaciones de recursos, Aurora Serverless v2 (y, por lo tanto, base de datos ilimitada de Aurora PostgreSQL) no es compatible con la zona de disponibilidad `us-east-1e`.

- Base de datos ilimitada de Aurora PostgreSQL solo admite la configuración de almacenamiento de clúster de base de datos Aurora I/O-Optimized. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.
- La base de datos ilimitada de Aurora PostgreSQL utiliza versiones especiales del motor de base de datos de Aurora PostgreSQL para la base de datos ilimitada de Aurora PostgreSQL:
  - `16.6.8-limitless`
  - `16.6-limitless`
  - `16.4-limitless`
- Su clúster de base de datos no puede tener instancias de base de datos de escritura o lectura.

- Debe usar Monitorización mejorada e Información de rendimiento. El período de retención de Información de rendimiento debe ser de al menos un mes (31 días).
- Debe exportar el registro de PostgreSQL a Registros de Amazon CloudWatch.

> **Note**
>
> Algunas características obligatorias, como Monitorización mejorada, Información de rendimiento y Registros de CloudWatch, conllevan cargos adicionales. Para obtener información acerca de los precios de Aurora, consulte la página de precios de Aurora.

Consideraciones sobre Base de datos ilimitada de Aurora PostgreSQL

Las siguientes consideraciones se aplican a los grupos de particiones de base de datos en Base de datos ilimitada de Aurora PostgreSQL:

- Solo puede tener un grupo de particiones de base de datos por clúster de base de datos.
- Puede disponer de hasta cinco grupos de particiones de base de datos por Región de AWS.

  Por lo tanto, puede tener hasta cinco clústeres de bases de datos de Base de datos ilimitada de Aurora PostgreSQL por Región de AWS. Para obtener más información, consulte Cuotas en Amazon Aurora.
- Puede establecer la capacidad máxima del grupo de particiones de base de datos entre 16 y 6144 ACU. Para conocer los límites de capacidad superiores a 6144 ACU, póngase en contacto con AWS.

  El número inicial de enrutadores y particiones viene determinado por la capacidad máxima que se establece al crear un grupo de particiones de base de datos. Para obtener más información, consulte Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada.
- La cantidad de enrutadores y particiones no cambia cuando se modifica la capacidad máxima de un grupo de particiones de base de datos.
- Asegúrese de que la subred de base de datos en la que ha creado el grupo de particiones de base de datos tenga suficientes direcciones IP libres para conectarse al grupo de particiones de base de datos. Necesita una dirección IP para cada enrutador y hasta tres direcciones IP para cada partición del grupo de particiones de base de datos.

Para obtener más información sobre el número de enrutadores que se genera al crear un grupo de particiones de base de datos, consulte Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada.

- Si hace que su grupo de particiones de base de datos sea de acceso público, asegúrese de configurar una puerta de enlace de Internet en su VPC.
- Las funciones SQL se utilizan para dividir las particiones y añadir enrutadores.
- No se admite la combinación de particiones.
- No puede eliminar particiones y enrutadores individuales.
- No puede modificar (realizar operaciones `UPDATE` en) las claves de partición, ni siquiera cambiar sus valores en las filas de la tabla.

  Para cambiar una clave de partición, elimínela y vuelva a crearla.
- Se admiten los niveles de aislamiento de lectura repetible, lectura confirmada y lectura no comprometida. No puede establecer el nivel de aislamiento como serializable.
- No se admiten algunos comandos SQL. Para obtener más información, consulte Referencia sobre Base de datos ilimitada de Aurora PostgreSQL.
- No se admiten todas las extensiones de PostgreSQL. Para obtener más información, consulte Extensiones.
- Al crear un grupo de particiones o al agregar nuevos nodos de grupos de particiones (particiones o enrutadores), esos nodos se crean en una de las zonas de disponibilidad (AZ) disponibles para el clúster de base de datos. No puede elegir una AZ específica para los nodos individuales.
- Si utiliza una redundancia de procesamiento de dos (dos esperas de computación para el grupo de particiones de base de datos), asegúrese de que su grupo de subredes de base de datos tenga al menos tres AZ.
- La base de datos ilimitada de Aurora PostgreSQL admite hasta 54 caracteres para los nombres de las tablas particionadas.

Las siguientes consideraciones se aplican al clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL:

- Le recomendamos que utilice políticas administradas por AWS para limitar los permisos de la base de datos y las aplicaciones a los que necesitan los clientes para sus casos de uso. Para obtener más información, consulte Prácticas recomendadas relativas a políticas.

- Al crear su clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL, solo establece los parámetros de escalado para el grupo de particiones de base de datos.
- Si necesita eliminar el clúster de base de datos, primero debe eliminar el grupo de particiones de base de datos.
- Base de datos ilimitada de Aurora PostgreSQL no puede ser un origen de replicación.

Características no admitidas por Base de datos ilimitada de Aurora PostgreSQL

Las siguientes características de Aurora PostgreSQL no son compatibles con Base de datos ilimitada de Aurora PostgreSQL:

- Autenticación de Active Directory (Kerberos)
- Amazon DevOps Guru
- Amazon ElastiCache
- Implementaciones azules/verdes de Amazon RDS
- Amazon RDS Proxy
- Aurora Auto Scaling (añadir automáticamente instancias de lectura al clúster de base de datos)
- Base de datos global de Aurora
- Machine Learning de Aurora
- Recomendaciones de Aurora
- Aurora Serverless v1
- Integraciones sin ETL de Aurora
- AWS Backup
- AWS LambdaIntegración de
- AWS Secrets Manager
- Babelfish para Aurora PostgreSQL
- Clonación de clústeres de base de datos
- Puntos de conexión personalizados
- Secuencias de actividades de la base de datos
- Réplicas de lectura
- API de datos de RDS

Requisitos previos para usar Base de datos ilimitada de Aurora PostgreSQL

Para utilizar Base de datos ilimitada de Aurora PostgreSQL, primero debe realizar las siguientes tareas.

Temas

- Activación de las operaciones de grupos de particiones de base de datos

Activación de las operaciones de grupos de particiones de base de datos

Antes de crear un grupo de particiones de base de datos, debe activar las operaciones del grupo de particiones de base de datos.

- Añada la siguiente sección a la política de IAM del rol de IAM del usuario que accede a Base de datos ilimitada de Aurora PostgreSQL:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDBShardGroup",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBShardGroup",
                "rds:DescribeDBShardGroups",
                "rds:DeleteDBShardGroup",
                "rds:ModifyDBShardGroup",
                "rds:RebootDBShardGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:*:shard-group:*",
                "arn:aws:rds:*:*:cluster:*"
            ]
        }
    ]
}
```

Creación de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL

Cree un nuevo clúster de base de datos de Aurora con la versión Base de datos ilimitada de Aurora PostgreSQL y añada un grupo de particiones de base de datos al clúster. Al agregar un grupo de particiones de base de datos, se especifica la capacidad de computación máxima para todo el grupo de particiones de base de datos (suma de las capacidades de todos los enrutadores y particiones) en unidades de capacidad (ACU) de Aurora. Cada ACU es una combinación de aproximadamente 2 gigabytes (GiB) de memoria, la CPU correspondiente y las redes. El escalado aumenta o reduce la capacidad del grupo de particiones de base de datos en función de la carga de trabajo de la aplicación, de forma similar al funcionamiento de Aurora Serverless v2.

Temas

- Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada
- Creación de un clúster de base de datos

Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada

El número inicial de enrutadores y particiones en un grupo de particiones de base de datos viene determinado por la capacidad máxima que se establece al crear el grupo de particiones de base de datos. Cuanto mayor sea la capacidad máxima, mayor será el número de enrutadores y particiones que se crea en el grupo de particiones de base de datos.

Cada nodo (partición o enrutador) tiene su propio valor de capacidad actual, que también se mide en ACU.

- Base de datos ilimitada escala un nodo hasta una capacidad superior cuando su capacidad actual es demasiado baja para gestionar la carga. Sin embargo, los nodos no se escalan verticalmente cuando la capacidad total está en su máximo.
- Base de datos ilimitada escala el nodo a una capacidad inferior cuando su capacidad actual es superior a la necesaria. Sin embargo, los nodos no se reducen verticalmente cuando la capacidad total está en su máximo.

La siguiente tabla muestra la correlación entre la capacidad máxima del grupo de particiones de base de datos en las unidades de capacidad (ACU) de Aurora y el número de nodos (enrutadores y particiones) creados.

> **Note**
>
> Estos valores están sujetos a cambios.
> Si establece la redundancia de computación en un valor distinto de cero, el número total de particiones se duplicará o triplicará. Esto generará un costo adicional.
> Los nodos en espera de computación se escalan y reducen verticalmente hasta alcanzar la misma capacidad que la del escritor. El rango de capacidad no se establece por separado para los modos de espera.

Total de nodos	Enrutadores	Particiones	Capacidad mínima (ACU) predeterminada	Rango de capacidad máxima (ACU)
4	2	2.	16	entre 16 y 400
5	2	3	20	entre 401 y 500
6	2	4	24	entre 501 y 600
7	3	4	28	entre 601 y 700
8	3	5	32	entre 701 y 800
9	3	6	36	entre 801 y 900
10	4	6	40	entre 901 y 1000
11	4	7	44	entre 1001 y 1100
12	4	8	48	entre 1101 y 1200

Total de nodos	Enrutadores	Particiones	Capacidad mínima (ACU) predeterminada	Rango de capacidad máxima (ACU)
13	5	8	52	entre 1201 y 1300
14	5	9	56	entre 1301 y 1400
15	5	10	60	entre 1401 y 1500
16	6	10	64	entre 1501 y 1600
17	6	11	68	entre 1601 y 1700
18	6	12	72	entre 1701 y 1800
19	7	12	76	entre 1801 y 1900
20	7	13	80	entre 1901 y 2000
21	7	14	84	entre 2001 y 2100
22	8	14	88	entre 2101 y 2200
23	8	15	92	entre 2201 y 2300
24	8	16	96	entre 2301 y 6144

La configuración dinámica basada en la capacidad máxima para el grupo de particiones de base de datos solo está disponible durante la creación. El número de enrutadores y particiones permanece igual cuando se modifica la capacidad máxima. Para obtener más información, consulte Modificación de la capacidad de un grupo de particiones de base de datos.

Puede usar comandos SQL para agregar particiones y enrutadores a un grupo de particiones de base de datos. Para obtener más información, consulte los siguientes temas:

- División de una partición en un grupo de particiones de base de datos
- Adición de un enrutador a un grupo de partición de base de datos

> **Note**
>
> No puede eliminar ni las particiones ni los enrutadores.

Creación de un clúster de base de datos

Aplique los siguientes procedimientos para crear un clúster de Base de datos ilimitada de Aurora PostgreSQL que utilice Base de datos ilimitada de Aurora PostgreSQL.

Puede utilizar la Consola de administración de AWS o la AWS CLI para crear su clúster de base de datos que utilice Base de datos ilimitada de Aurora PostgreSQL. Cree el clúster de base de datos principal y el grupo de particiones de base de datos.

Consola

Cuando se utiliza la Consola de administración de AWS para crear el clúster de base de datos principal, el grupo de particiones de base de datos también se crea siguiendo el mismo procedimiento.

Creación de un clúster de base de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Create database (Crear base de datos).

   Aparece la página Crear base de datos.
3. En Tipo de motor, elija Aurora (compatible con PostgreSQL).
4. Para Versión, elija una de las siguientes opciones:

   - Aurora PostgreSQL con base de datos ilimitada (compatible con PostgreSQL 16.4)
   - Aurora PostgreSQL con base de datos ilimitada (compatible con PostgreSQL 16.6)
5. En Base de datos ilimitada de Aurora PostgreSQL:

a. Introduzca un Identificador de grupo de particiones de base de datos.

> **Important**
>
> Tras crear el grupo de particiones de base de datos, ya no podrá cambiar el identificador del clúster de base de datos ni el identificador del grupo de particiones de base de datos.

b. Para Rango de capacidad del grupo de particiones de base de datos:

   i. Introduzca la Capacidad mínima (ACU). Utilice un valor de al menos 16 ACU.

      Para un entorno de desarrollo, el valor predeterminado es 16 ACU. Para un entorno de producción, el valor predeterminado es 24 ACU.

   ii. Introduzca la Capacidad máxima (ACU). Utilice un valor de al menos 16 ACU o como máximo 6144 ACU.

      Para un entorno de desarrollo, el valor predeterminado es 64 ACU. Para un entorno de producción, el valor predeterminado es 384 ACU.

   Para obtener más información, consulte Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada.

c. Para Implementación del grupo de partición de base de datos, elija si desea crear esperas para el grupo de particiones de base de datos:

- Sin redundancia de computación: crea un grupo de particiones de base de datos sin esperas para cada partición. Este es el valor predeterminado.
- Redundancia de computación con un solo destino de conmutación por error: crea un grupo de particiones de base de datos con una espera de computación en una zona de disponibilidad (AZ) diferente.
- Redundancia de computación con dos destinos de conmutación por error: crea un grupo de particiones de base de datos con dos esperas de computación en dos zonas de disponibilidad diferentes.

> **Note**
>
> Si establece la redundancia de computación en un valor distinto de cero, el número total de instancias de base de datos de particiones se duplicará o triplicará. Estas instancias de base de datos adicionales son instancias en espera de computación, que se escalan y reducen verticalmente hasta alcanzar la misma capacidad que la del escritor. El rango de capacidad no se establece por separado para los modos de espera. Por lo tanto, el uso y la facturación de ACU se duplican y triplican en consecuencia. Para conocer el uso exacto de ACU derivado de la redundancia informática, consulte la métrica `DBShardGroupComputeRedundancyCapacity` en Métricas de DBShardGroup.

d. Elija si desea que el grupo de particiones de base de datos sea de acceso público.

> **Note**
>
> No puede modificar esta configuración después de crear el grupo de particiones de base de datos.

6. Para Conectividad:

a. (Opcional) Seleccione Conectarse a un recurso de computación de EC2 y, a continuación, elija una instancia de EC2 existente o cree una nueva.

> **Note**
>
> Si se conecta a una instancia de EC2, no podrá hacer que el grupo de particiones de base de datos sea de acceso público.

b. Para Tipo de red, elija IPv4 o Modo de pila doble.

c. Elija la nube privada virtual (VPC) y el grupo de subredes de la base de datos, o utilice la configuración predeterminada.

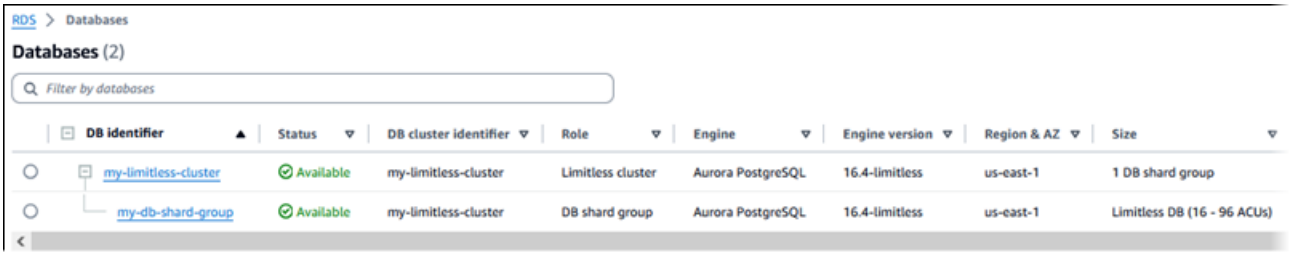
> **Note**
>
> Si crea su clúster de base de datos de Base de datos ilimitada en la región de Este de EE. UU. (Norte de Virginia), no incluya la zona de disponibilidad (AZ) `us-east-1e` en su grupo de subredes de base de datos. Debido a las limitaciones de recursos, Aurora Serverless v2 (y, por lo tanto, Base de datos ilimitada) no es compatible con la AZ `us-east-1e`.

d. Elija Grupo de seguridad de VPC (firewall) o utilice la configuración predeterminada.

7. Para Autenticación de bases de datos, elija Autenticación con contraseña o Autenticación de bases de datos con contraseña e IAM.
8. Para Supervisión, asegúrese de que las casillas de verificación Activar Información de rendimiento y Activar la monitorización mejorada estén seleccionadas.

   Para Información de rendimiento, elija un tiempo de retención de al menos un mes.
9. Amplíe la última Configuración adicional de la página.
10. En Exportaciones de registros, asegúrese de que la casilla de verificación de Registro de PostgreSQL esté seleccionada.
11. Especifique otros valores según sea necesario. Para obtener más información, consulte Configuración de clústeres de bases de datos de Aurora.
12. Elija Creación de base de datos.

Una vez creados el clúster de base de datos principal y el grupo de particiones de base de datos, se muestran en la página Bases de datos.


RDS > Databases
Databases (2)
Filter by databases
DB identifier
Status
DB cluster identifier
Role
Engine
Engine version
Region & AZ
Size
my-limitless-cluster
Available
my-limitless-cluster
Limitless cluster
Aurora PostgreSQL
16.4-limitless
us-east-1
1 DB shard group
my-db-shard-group
Available
my-limitless-cluster
DB shard group
Aurora PostgreSQL
16.4-limitless
us-east-1
Limitless DB (16 - 96 ACUs)

CLI

Cuando utilice la AWS CLI para crear un clúster de base de datos que utilice Base de datos ilimitada de Aurora PostgreSQL, debe realizar las siguientes tareas:

1. Creación de un clúster de base de datos principal.
2. Creación del grupo de particiones de base de datos.

Creación de un clúster de base de datos principal

Para crear el clúster de base de datos, se necesitan los siguientes parámetros:

- `--db-cluster-identifier`: es el nombre del clúster de base de datos.
- `--engine`: el clúster de base de datos debe utilizar el motor de base de datos `aurora-postgresql`.
- `--engine-version`: el clúster de base de datos debe utilizar una de las versiones del motor de base de datos:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: el clúster de base de datos debe utilizar la configuración de almacenamiento del clúster de base de datos `aurora-iopt1`.
- `--cluster-scalability-type`: especifica el modo de escalabilidad del clúster de base de datos de Aurora. Cuando se establece en `limitless`, el clúster funciona como una Base de datos ilimitada de Aurora PostgreSQL. Si se establece en `standard` (valor predeterminado), el clúster utiliza la creación normal de instancias de base de datos.

  > **Note**
  >
  > No puede modificar esta configuración después de crear el clúster de base de datos.

- `--master-username`: es el nombre del usuario maestro del clúster de base de datos.
- `--master-user-password`: es la contraseña para el usuario maestro.
- `--enable-performance-insights`: debe habilitar Información de rendimiento.
- `--performance-insights-retention-period`: el período de retención de Información de rendimiento debe ser de al menos 31 días.

- `--monitoring-interval`: es el intervalo, en segundos, entre puntos cuando se recopilan las métricas de Monitorización mejorada para el clúster de base de datos. Este valor no puede ser `0`.
- `--monitoring-role-arn`: es el nombre de recurso de Amazon (ARN) del rol de IAM que permite a RDS enviar métricas de Monitorización mejorada a Registros de Amazon CloudWatch.
- `--enable-cloudwatch-logs-exports`: debe exportar los registros de `postgresql` a Registros de CloudWatch.

Los siguientes parámetros son opcionales:

- `--db-subnet-group-name`: es el grupo de subredes de la base de datos que desea asociar con el clúster de base de datos. Esto también determina la VPC asociada al clúster de base de datos.

  > **Note**
  >
  > Si crea su clúster de base de datos de Base de datos ilimitada en la región de Este de EE. UU. (Norte de Virginia), no incluya la zona de disponibilidad (AZ) `us-east-1e` en su grupo de subredes de base de datos. Debido a las limitaciones de recursos, Aurora Serverless v2 (y, por lo tanto, Base de datos ilimitada) no es compatible con la AZ `us-east-1e`.

- `--vpc-security-group-ids`: es una lista de grupos de seguridad de VPC para asociar con el clúster de base de datos.
- `--performance-insights-kms-key-id`: es el identificador AWS KMS key para el cifrado de datos de Información de rendimiento. Si no especifica una clave de KMS, se utiliza la clave predeterminada para su Cuenta de AWS.
- `--region`: es la Región de AWS donde se crea el clúster de base de datos. Debe ser compatible con Base de datos ilimitada de Aurora PostgreSQL.

Para usar la VPC y el grupo de seguridad de VPC predeterminados, omita las opciones `--db-subnet-group-name` y `--vpc-security-group-ids`.

Creación de un clúster de base de datos principal

- ```
  aws rds create-db-cluster \
      --db-cluster-identifier my-limitless-cluster \
      --engine aurora-postgresql \
      --engine-version 16.6-limitless \
      --storage-type aurora-iopt1 \
  ```
  ```

```
    --cluster-scalability-type limitless \
    --master-username myuser \
    --master-user-password mypassword \
    --db-subnet-group-name mysubnetgroup \
    --vpc-security-group-ids sg-c7e5b0d2 \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
    --enable-cloudwatch-logs-exports postgresql
```

Para obtener más información, consulte create-db-cluster.

Creación del grupo de particiones de base de datos

A continuación, cree el grupo de particiones de base de datos en el clúster de base de datos que acaba de crear. Se requieren los siguientes parámetros:

- `--db-shard-group-identifier`: es el nombre de su grupo de particiones de base de datos.

  El identificador del grupo de particiones de base de datos tiene las siguientes restricciones:

  - Debe ser único en la Cuenta de AWS y en la Región de AWS donde se ha creado.
  - Debe contener entre 1 y 63 letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.
  -

    > **Important**
    >
    > Tras crear el grupo de particiones de base de datos, ya no podrá cambiar el identificador del clúster de base de datos ni el identificador del grupo de particiones de base de datos.

- `--db-cluster-identifier`: es el nombre del clúster de base de datos en el que está creando el grupo de particiones de base de datos.
- `--max-acu`: es la capacidad máxima de su grupo de particiones de base de datos. Debe ser de entre 16 y 6144 ACU. Para conocer los límites de capacidad superiores a 6144 ACU, póngase en contacto con AWS.

  El número inicial de enrutadores y particiones viene determinado por la capacidad máxima que se establece al crear el grupo de particiones de base de datos. Cuanto mayor sea la capacidad máxima, mayor será el número de enrutadores y particiones que se crea en el grupo de particiones de base de datos. Para obtener más información, consulte Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada.

Los siguientes parámetros son opcionales:

- `--compute-redundancy`: indica si se deben crear esperas para el grupo de particiones de base de datos. Este parámetro puede tener uno de los siguientes valores:
  - `0`: crea un grupo de particiones de base de datos sin esperas para cada partición. Este es el valor predeterminado.

- 1: crea un grupo de particiones de base de datos con una espera de computación en una zona de disponibilidad (AZ) diferente.
- 2: crea un grupo de particiones de base de datos con dos esperas de computación en dos AZ diferentes.

> **Note**
>
> Si establece la redundancia de computación en un valor distinto de cero, el número total de particiones se duplicará o triplicará. Esto generará un costo adicional.
> Los nodos en espera de computación se escalan y reducen verticalmente hasta alcanzar la misma capacidad que la del escritor. El rango de capacidad no se establece por separado para los modos de espera.

- `--min-acu`: es la capacidad mínima de su grupo de particiones de base de datos. Debe tener al menos 16 ACU, que es el valor predeterminado.
- `--publicly-accessible|--no-publicly-accessible`: indica si se deben asignar direcciones IP de acceso público al grupo de particiones de base de datos. El acceso al grupo de particiones de base de datos está controlado por los grupos de seguridad que utiliza el clúster.

  El valor predeterminado es `--no-publicly-accessible`.

> **Note**
>
> No puede modificar esta configuración después de crear el grupo de particiones de base de datos.

Creación del grupo de particiones de base de datos

-
```
aws rds create-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --db-cluster-identifier my-limitless-cluster \
    --max-acu 1000
```

Uso de los grupos de particiones de base de datos

Realice las siguientes tareas para agregar y administrar un grupo de particiones de base de datos en Base de datos ilimitada de Aurora PostgreSQL.

Temas

- Conexión a su clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL
- Búsqueda del número de enrutadores y particiones en un grupo de particiones de base de datos
- Descripción de los grupos de particiones de base de datos
- Reinicio de un grupo de particiones de base de datos
- Modificación de la capacidad de un grupo de particiones de base de datos
- División de una partición en un grupo de particiones de base de datos
- Adición de un enrutador a un grupo de partición de base de datos
- Eliminación de un grupo de particiones de base de datos
- Adición de un grupo de particiones de base de datos a un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL existente

Conexión a su clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL

Para trabajar con la base de datos ilimitada de Aurora PostgreSQL, debe conectarse al escritor del clúster o punto de conexión del lector. Puede utilizar `psql` o cualquier otra utilidad de conexión que funcione con PostgreSQL:

```
$ psql -h DB_cluster_endpoint -p port_number -U database_username -d postgres_limitless
```

En el siguiente ejemplo se utiliza el punto de conexión para el clúster de base de datos que ha creado en CLI.

```
$ psql -h my-limitless-cluster.cluster-ckifpdyyyxxx.us-east-1.rds.amazonaws.com -p 5432
 -U postgres -d postgres_limitless
```

> **Note**
>
> La base de datos predeterminada para el grupo de particiones de base de datos de Base de datos ilimitada de Aurora PostgreSQL es `postgres_limitless`.

Uso del complemento Limitless Connection

Al conectarse a Base de datos ilimitada de Aurora PostgreSQL, los clientes se conectan con el punto de conexión del clúster y Amazon Route 53 los redirige a un enrutador de transacciones. Sin embargo, Route 53 tiene una capacidad limitada para equilibrar la carga y puede permitir cargas de trabajo desiguales en los enrutadores de transacciones. El complemento Limitless Connection para el controlador AWS JDBC soluciona este problema al equilibrar la carga del cliente teniendo en cuenta la carga. Para obtener más información sobre el controlador de AWS JDBC, consulte Conexión a Aurora PostgreSQL con el controlador JDBC de Amazon Web Services (AWS).

Búsqueda del número de enrutadores y particiones en un grupo de particiones de base de datos

Puede utilizar la siguiente consulta para buscar el número de enrutadores y particiones:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Descripción de los grupos de particiones de base de datos

Utilice el comando `describe-db-shard-groups` de la AWS CLI para describir los grupos de particiones de base de datos. El siguiente parámetro es opcional:

- `--db-shard-group-identifier`: es el nombre de un grupo de particiones de base de datos.

En el siguiente ejemplo se describe un grupo de particiones de base de datos específico.

```
aws rds describe-db-shard-groups --db-shard-group-identifier my-db-shard-group
```

La salida es similar al siguiente ejemplo.

```
{
    "DBShardGroups": [
        {
            "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
            "DBShardGroupIdentifier": "my-shard-group",
            "DBClusterIdentifier": "my-limitless-cluster",
            "MaxACU": 1000.0,
            "ComputeRedundancy": 0,
            "Status": "available",
            "PubliclyAccessible": false,
```

```
            "Endpoint": "my-limitless-cluster.limitless-ccetp2abcdef.us-
east-1.rds.amazonaws.com"
        }
    ]
}
```

Reinicio de un grupo de particiones de base de datos

A veces hay que reiniciar el grupo de particiones de base de datos, por ejemplo, cuando el parámetro `max_connections` cambia debido a una modificación de la capacidad máxima.

Puede usar la Consola de administración de AWS o la AWS CLI para cambiar la capacidad de un grupo de particiones de base de datos.

Consola

Utilice el siguiente procedimiento:

Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

1. Acceda a la página Databases (Bases de datos).
2. Seleccione el grupo de particiones de base de datos que desee reiniciar.
3. Para Actions (Acciones), elija Reboot (Reiniciar).
4. Elija Confirmar.

CLI

Para reiniciar un grupo de particiones de base de datos, utilice el comando `reboot-db-shard-group` de la AWS CLI con el siguiente parámetro:

- `--db-shard-group-identifier`: es el nombre de un grupo de particiones de base de datos.

En el siguiente ejemplo de , se reinicia un grupo de partición de base de datos.

```
aws rds reboot-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Modificación de la capacidad de un grupo de particiones de base de datos

Puede usar la Consola de administración de AWS o la AWS CLI para cambiar la capacidad de un grupo de particiones de base de datos.

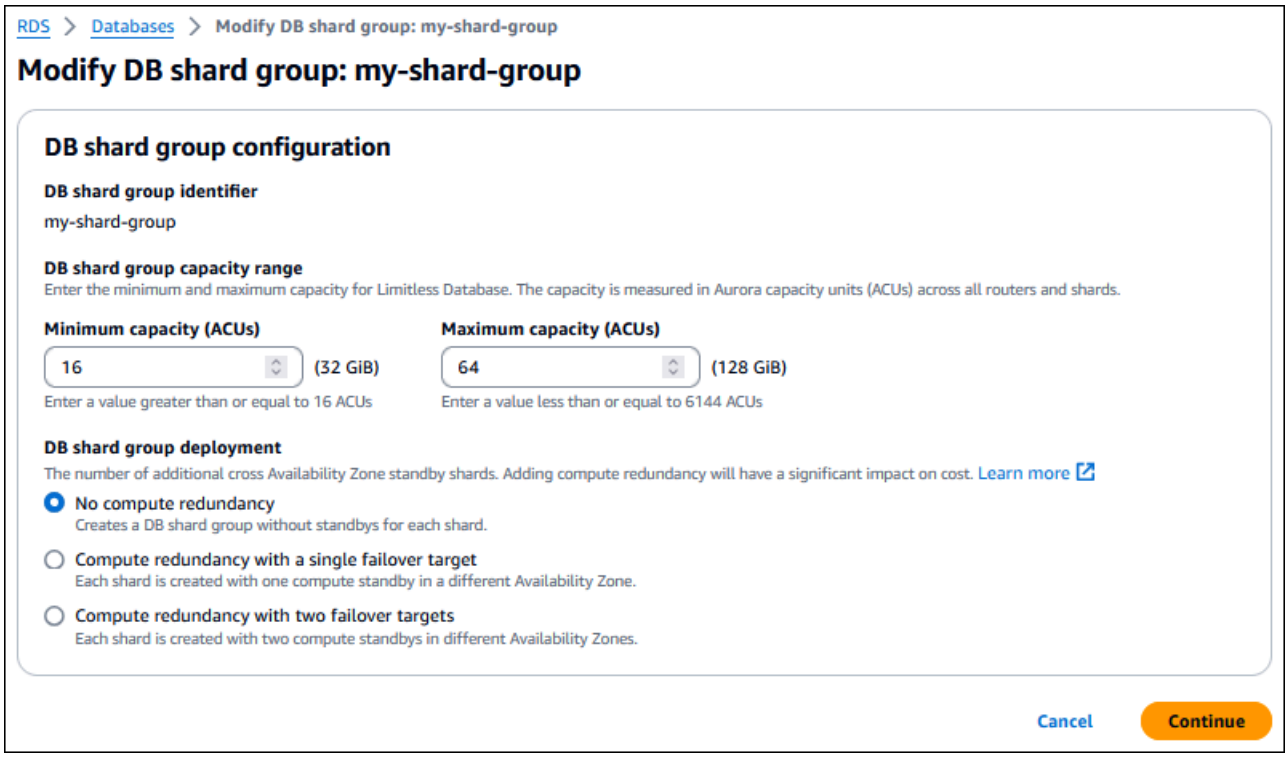
Consola

Utilice el siguiente procedimiento:

Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

1. Acceda a la página Databases (Bases de datos).
2. Seleccione el grupo de particiones de base de datos que desee modificar.
3. Para Actions (Acciones), elija Modify (Modificar).

   Aparece la página Modificar grupo de particiones de base de datos.



4. Introduzca un nuevo valor en Capacidad mínima (ACU), por ejemplo, **`100`**.
5. Introduzca un nuevo valor en Capacidad máxima (ACU), por ejemplo, **`1000`**.
6. Elija Continuar.

   Aparece la página de confirmación con un resumen de los cambios.

7. Revise los cambios y, a continuación, seleccione Modificar grupo de particiones de base de datos.

CLI

Use el comando `modify-db-shard-group` de la AWS CLI con los siguientes parámetros:

- `--db-shard-group-identifier`: es el nombre del grupo de particiones de base de datos.
- `--max-acu`: es la nueva capacidad máxima del grupo de particiones de base de datos. Puede establecer la capacidad máxima del grupo de particiones de base de datos entre 16 y 6144 ACU. Para conocer los límites de capacidad superiores a 6144 ACU, póngase en contacto con AWS.

  La cantidad de enrutadores y particiones no cambia.
- `--min-acu`: es la nueva capacidad mínima de su grupo de particiones de base de datos. Debe tener al menos 16 ACU, que es el valor predeterminado.

El siguiente ejemplo de la CLI cambia el rango de capacidad de un grupo de particiones de base de datos a entre 100 y 1000 ACU.

```
aws rds modify-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --min-acu 100 \
    --max-acu 1000
```

División de una partición en un grupo de particiones de base de datos

Puede dividir manualmente una partición de un grupo de particiones de base de datos en dos particiones más pequeñas. Esto se denomina división de particiones iniciada por el usuario.

Base de datos ilimitada de Aurora PostgreSQL también puede dividir las particiones si contienen grandes cantidades de datos o se usan con mucha frecuencia. Esto se denomina división de particiones iniciada por el sistema.

Temas

- Requisitos previos
- División de una partición
- Seguimiento de divisiones de particiones
- Finalización de las divisiones de particiones
- Cancelación de la división de particiones

Requisitos previos

Las divisiones de particiones iniciadas por el usuario tienen los siguientes requisitos previos:

- Debe tener un grupo de particiones de base de datos.
- El grupo de particiones de base de datos no puede estar vacío: debe contener al menos una tabla particionada.
- Un usuario debe tener el privilegio `rds_aurora_limitless_cluster_admin`. El `rds_superuser` tiene este privilegio; por lo tanto, el usuario maestro también lo tiene. El `rds_superuser` puede conceder el privilegio a otros usuarios:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

- Debe conocer el ID del subclúster (nodo) de la partición que desea dividir. Puede obtener el ID realizando la siguiente consulta:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
```

```
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Para habilitar la división de particiones iniciada por el sistema, debe establecer los siguientes parámetros de clúster de base de datos en un grupo de parámetros de clúster de base de datos personalizado asociado a su clúster de base de datos:

Parámetro	Valor
`rds_aurora.limitless_enable_auto_scale`	on
`rds_aurora.limitless_auto_scale_options`	Indique `split_shard` o `add_router,split_shard`
`rds_aurora.limitless_finalize_split_shard_mode`	Esta parámetro determina cómo finalizan las divisiones de particiones iniciadas por el sistema. El valor puede ser uno de los siguientes: • `user_initiated` : usted decide cuándo debe finalizar la división de particiones. Este es el valor predeterminado. • `immediate` : las divisiones de particiones finalizan de inmediato. Para obtener más información, consulte Finalización de las divisiones de particiones.

Parámetro	Valor
	Note Este parámetro solo se aplica a divisiones de particiones iniciadas por el sistema.

Para obtener más información, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

División de una partición

Para dividir una partición en un grupo de particiones de base de datos, utilice la función `rds_aurora.limitless_split_shard`. Esta función inicia un trabajo de división de particiones que se ejecuta de forma asíncrona.

```
SELECT rds_aurora.limitless_split_shard('subcluster_id');
```

Espere a que se devuelva el ID del trabajo cuando el trabajo se haya enviado correctamente, por ejemplo:

```
SELECT rds_aurora.limitless_split_shard('3');

    job_id
---------------
 1691300000000
(1 row)
```

Note

No se admiten las operaciones concurrentes de división de particiones. Ejecute cada operación secuencialmente y complete cada operación antes de iniciar otra operación de agregación.

Seguimiento de divisiones de particiones

Puede usar el ID del trabajo para realizar el seguimiento de un trabajo de división de particiones. Para describir un trabajo en particular y obtener más detalles sobre él, ejecute la siguiente consulta:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(job_id);
```

Por ejemplo:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(1691300000000);

    job_id     |    action   |      job_details      | status  |    submission_time
 |                 message
---------------+-------------+-----------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | SPLIT_SHARD | Split Shard 3 by User | SUCCESS | 2023-08-06 05:33:20+00
 | Scaling job succeeded.                 +
               |             |                       |         |
 | New shard instance with ID 7 was created.
(1 row)
```

La consulta devuelve un error cuando se pasa un trabajo inexistente como entrada.

```
SELECT * from rds_aurora.limitless_list_shard_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

Puede realizar un seguimiento del estado de todos los trabajos de división de particiones con la misma consulta sin un ID de trabajo, por ejemplo:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs();

    job_id     |   action    |  job_details          |   status    |    submission_time
     |                 message
---------------+-------------+-----------------------+-------------
+------------------------
+-------------------------------------------------------------
 1691200000000 | SPLIT_SHARD | Split Shard 3 by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | SPLIT_SHARD | Split Shard 4 by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded. +
```

```
               |            |                        |            |
    | New shard instance with ID 7 was created.
 1691400000000 | SPLIT_SHARD | Split Shard 5 by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add shard job 1691400000000.
               |            |                        |            |
    | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | SPLIT_SHARD | Split Shard 5 by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

El estado del trabajo puede ser uno de los siguientes:

- `IN_PROGRESS`: el trabajo de división de particiones se ha enviado y está en curso. Solo puede tener un trabajo en curso a la vez.
- `PENDING`: el trabajo de división de particiones está esperando a que lo finalice. Para obtener más información, consulte Finalización de las divisiones de particiones.
- `CANCELLATION_IN_PROGRESS`: el usuario está cancelando el trabajo de división de particiones.
- `CANCELED`: el usuario o el sistema han cancelado correctamente el trabajo de división de particiones.
- `SUCCESS`: el trabajo de división de particiones se ha completado correctamente. El campo `message` contiene el ID de instancia de la nueva partición.
- `FAILED`: el trabajo de división de particiones ha fallado. El campo `message` contiene los detalles del error y cualquier acción que se pueda tomar como seguimiento del trabajo fallido.

Finalización de las divisiones de particiones

La finalización es el último paso del proceso de división de particiones. Esta tarea provoca algunos tiempos de inactividad. Si inicia un trabajo de división de particiones, la finalización se produce inmediatamente después de que el trabajo se complete correctamente.

A veces, el sistema divide las particiones en función de la carga de trabajo, cuando activa las divisiones de particiones iniciadas por el sistema con el parámetro `rds_aurora.limitless_enable_auto_scale`.

En este caso, puede elegir si quiere que la finalización se produzca inmediatamente o en el momento que desee. Utilice el parámetro de clúster de bases de datos `rds_aurora.limitless_finalize_split_shard_mode` para elegir cuándo ocurrirá:

- Si establece el valor en `immediate`, ocurrirá inmediatamente.
- Si establece el valor en `user_initiated`, tendrá que finalizar el trabajo de división de particiones manualmente.

  Se le envía un evento de RDS y el estado del trabajo de división de particiones se establece en `PENDING`.

Cuando se establece en `user_initiated`, utiliza la función `rds_aurora.limitless_finalize_split_shard` para finalizar el trabajo de división de particiones:

```
SELECT * FROM rds_aurora.limitless_finalize_split_shard(job_id);
```

> **Note**
>
> Esta función solo se aplica a las divisiones de particiones iniciadas por el sistema, no por usted.

Cancelación de la división de particiones

Puede cancelar una división de particiones iniciada por el usuario o por el sistema, es decir, `IN_PROGRESS` o `PENDING`. Para cancelarlo, necesita el ID del trabajo.

```
SELECT * from rds_aurora.limitless_cancel_shard_scale_jobs(job_id);
```

No se devuelve ningún resultado a menos que haya un error. Puede hacer un seguimiento de la cancelación mediante una consulta de seguimiento del trabajo.

Adición de un enrutador a un grupo de partición de base de datos

Puede añadir un enrutador a un grupo de partición de base de datos.

Temas

- Requisitos previos
- Adición de un enrutador
- Seguimiento de la adición de enrutadores
- Cancelación de la adición del enrutador

Requisitos previos

La adición de un enrutador tiene los siguientes requisitos previos:

- Debe tener un grupo de particiones de base de datos.
- Un usuario debe tener el privilegio `rds_aurora_limitless_cluster_admin`. El `rds_superuser` tiene este privilegio; por lo tanto, el usuario maestro también lo tiene. El `rds_superuser` puede conceder el privilegio a otros usuarios:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

> **Note**
>
> Si cambia el certificado de CA predeterminado de su Cuenta de AWS después de crear el grupo de partición de base de datos, el nuevo enrutador utilizará el nuevo certificado de CA, que es diferente del certificado de CA del enrutador existente. En función del almacén de confianza, es posible que algunas conexiones fallen.

- Para habilitar la adición de enrutadores iniciada por el sistema, establezca los siguientes parámetros de clúster de base de datos en un grupo de parámetros de clúster de base de datos personalizado asociado a su clúster de base de datos:

Parámetro	Valor
`rds_aurora.limitless_enable_auto_scale`	`on`
`rds_aurora.limitless_auto_scale_options`	`add_router` o `add_router,split_shard`

Para obtener más información, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

Adición de un enrutador

Para añadir un enrutador, utilice la función `rds_aurora.limitless_add_router`. Esta función inicia un trabajo de adición de enrutadores que se ejecuta de forma asíncrona.

```
SELECT rds_aurora.limitless_add_router();
```

Espere a que se devuelva el ID del trabajo cuando el trabajo se haya enviado correctamente, por ejemplo:

```
    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> No se admiten las operaciones de adición concurrente de enrutadores. Ejecute las operaciones secuencialmente y complete cada operación antes de iniciar otra operación de agregación.

Seguimiento de la adición de enrutadores

Puede usar el ID del trabajo para realizar un seguimiento de un trabajo de adición de un enrutador. Para describir un trabajo en particular y obtener más detalles sobre él, ejecute la siguiente consulta:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(job_id);
```

Por ejemplo:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(1691300000000);

    job_id     |   action   |        job_details       | status  |    submission_time
   |                  message
---------------+------------+--------------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                  +
               |            |                          |         |
   | New router instance with ID 7 was created.
(1 row)
```

La consulta devuelve un error cuando se pasa un trabajo inexistente como entrada.

```
SELECT * from rds_aurora.limitless_list_router_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

Puede realizar un seguimiento del estado de todos los trabajos de adición de enrutadores con la misma consulta sin un ID de trabajo, por ejemplo:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs();

    job_id     |   action   |        job_details       |   status    |
 submission_time     |                 message
---------------+------------+--------------------------+-------------
+------------------------+-------------------------------------------
 1691200000000 | ADD_ROUTER | Add 1 new Router by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                 +
               |            |                          |             |
       | New router instance with ID 7 was created.
```

```
 1691400000000 | ADD_ROUTER | Add 1 new Router by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add router job 1691400000000.
               |            |                          |             |
       | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | ADD_ROUTER | Add 1 new Router by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

El estado del trabajo puede ser uno de los siguientes:

- `IN_PROGRESS`: el trabajo de adición del enrutador se ha enviado y está en curso. Solo puede tener un trabajo en curso a la vez.
- `CANCELLATION_IN_PROGRESS`: el usuario está cancelando el trabajo de adición del enrutador.
- `CANCELED`: el usuario o el sistema han cancelado correctamente el trabajo de adición del enrutador.
- `SUCCESS`: el trabajo de adición del enrutador se ha completado correctamente. El campo `message` contiene el ID de instancia del nuevo enrutador.
- `FAILED`: el trabajo de adición del enrutador ha fallado. El campo `message` contiene los detalles del error y cualquier acción que se pueda tomar como seguimiento del trabajo fallido.

> **Note**
>
> No hay ningún estado `PENDING` porque no es necesario finalizar las adiciones de enrutadores. No se produce ningún tiempo de inactividad.

Cancelación de la adición del enrutador

Puede cancelar la adición de un enrutador que tenga el estado `IN_PROGRESS`. Para cancelarlo, necesita el ID del trabajo.

```
SELECT * from rds_aurora.limitless_cancel_router_scale_jobs(job_id);
```

No se devuelve ningún resultado a menos que haya un error. Puede hacer un seguimiento de la cancelación mediante una consulta de seguimiento del trabajo.

Eliminación de un grupo de particiones de base de datos

Si es necesario, puede eliminar un grupo de particiones de base de datos. Al eliminar el grupo de particiones de base de datos, se eliminan los nodos de computación (particiones y enrutadores), pero no el almacenamiento.

Consola

Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

1. Acceda a la página Databases (Bases de datos).
2. Seleccione el grupo de particiones de base de datos que desee eliminar.
3. En Actions (Acciones), seleccione Delete (Eliminar).
4. Escriba **`delete me`** en el cuadro y, a continuación, elija Eliminar.

Se elimina el grupo de particiones de base de datos.

AWS CLI

Para eliminar su grupo de particiones de base de datos, utilice el comando `delete-db-shard-group` de la AWS CLI con los siguientes parámetros:

- `--db-shard-group-identifier`: es el nombre del grupo de particiones de base de datos.

En el siguiente ejemplo, se elimina un grupo de particiones de base de datos en el clúster de base de datos de Aurora PostgreSQL.

```
aws rds delete-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Adición de un grupo de particiones de base de datos a un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL existente

Puede crear un grupo de particiones de base de datos en un clúster de base de datos existente, por ejemplo, si está restaurando un clúster de base de datos o si ha eliminado el grupo de particiones de base de datos.

Para obtener más información sobre los requisitos del clúster de base de datos principal y del grupo de particiones de base de datos, consulte Requisitos y consideraciones sobre Base de datos ilimitada de Aurora PostgreSQL.

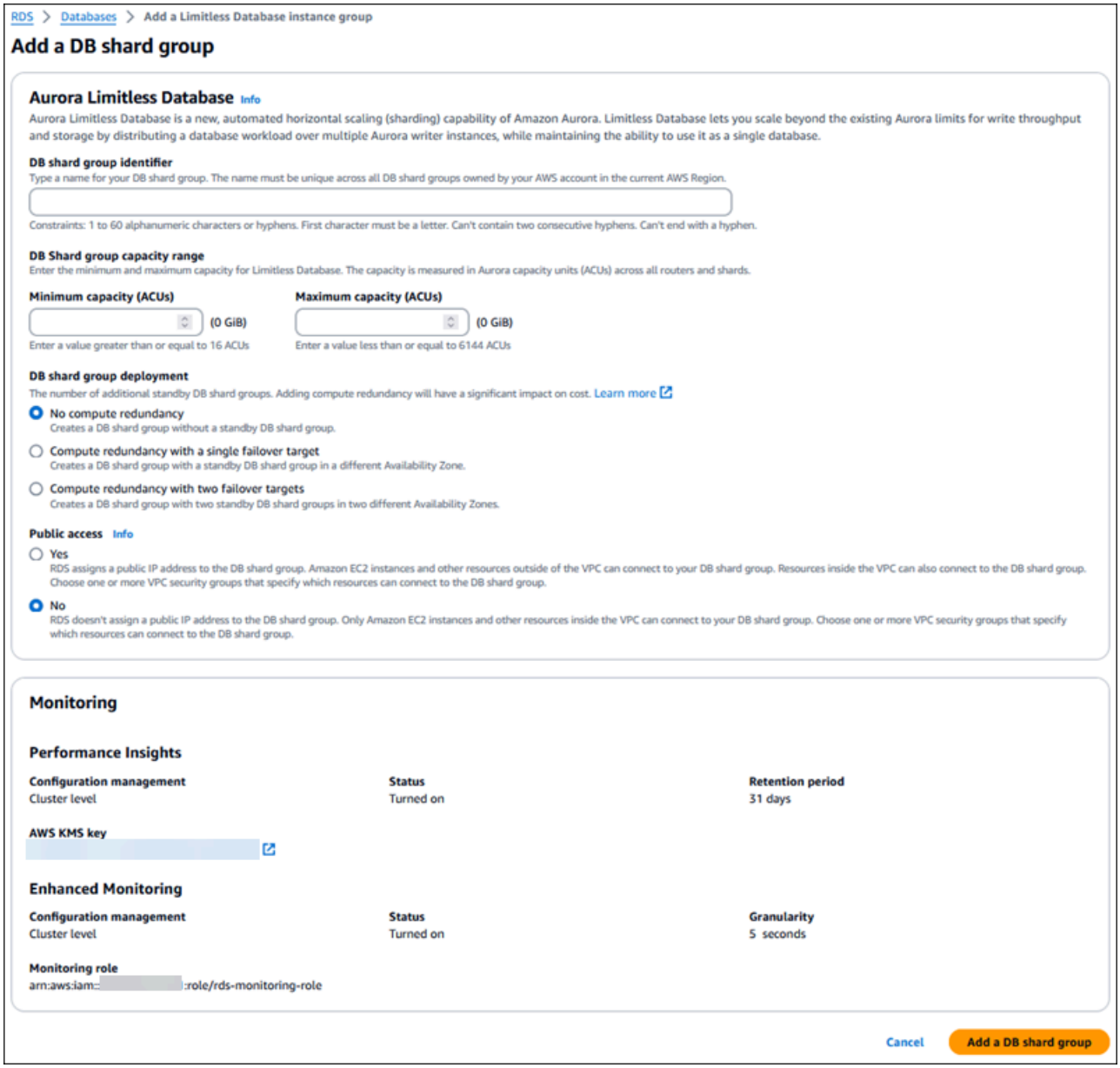
> **Note**
>
> Solo puede tener un grupo de particiones de base de datos por clúster.
> El clúster de base de datos de Base de datos ilimitada debe tener el estado `available` para poder crear un grupo de particiones de base de datos.

Consola

Puede usar la Consola de administración de AWS para añadir un grupo de particiones de base de datos a un clúster de base de datos existente.

Adición de un grupo de particiones de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Acceda a la página Databases (Bases de datos).
3. Seleccione el clúster de base de datos de Base de datos ilimitada al que desea agregar un grupo de particiones de base de datos.
4. En Acciones, elija Agregar un grupo de particiones de base de datos.



5. Introduzca un Identificador de grupo de particiones de base de datos.

> **Important**
>
> Tras crear el grupo de particiones de base de datos, ya no podrá cambiar el identificador del clúster de base de datos ni el identificador del grupo de particiones de base de datos.

6. Introduzca la Capacidad mínima (ACU). Utilice un valor de al menos 16 ACU.
7. Introduzca la Capacidad máxima (ACU). Utilice un valor entre 16 y 6144 ACU.

Para obtener más información, consulte Correlación de la capacidad máxima del grupo de particiones de base de datos con la cantidad de enrutadores y particiones creada.

8. Para Implementación del grupo de partición de base de datos, elija si desea crear esperas para el grupo de particiones de base de datos:

   - Sin redundancia de computación: crea un grupo de particiones de base de datos sin esperas para cada partición. Este es el valor predeterminado.
   - Redundancia de computación con un solo destino de conmutación por error: crea un grupo de particiones de base de datos con una espera de computación en una zona de disponibilidad (AZ) diferente.
   - Redundancia de computación con dos destinos de conmutación por error: crea un grupo de particiones de base de datos con dos esperas de computación en dos zonas de disponibilidad diferentes.

9. Elija si desea que el grupo de particiones de base de datos sea de acceso público.

   > **Note**
   >
   > No puede modificar esta configuración después de crear el grupo de particiones de base de datos.

10. Elija Agregar un grupo de particiones de base de datos.

AWS CLI

Utilice el comando `create-db-shard-group` de la AWS CLI para crear un grupo de particiones de base de datos.

Se requieren los siguientes parámetros:

- `--db-cluster-identifier`: es el clúster de base de datos al que pertenece el grupo de particiones de base de datos.
- `--db-shard-group-identifier`: es el nombre del grupo de particiones de base de datos.

  El identificador del grupo de particiones de base de datos tiene las siguientes restricciones:

  - Debe ser único en la Cuenta de AWS y en la Región de AWS donde se ha creado.
  - Debe contener entre 1 y 63 letras, números o guiones.
  - El primer carácter debe ser una letra.
  - No puede terminar con un guion ni contener dos guiones consecutivos.

  > **Important**
  >
  > Tras crear el grupo de particiones de base de datos, ya no podrá cambiar el identificador del clúster de base de datos ni el identificador del grupo de particiones de base de datos.

- `--max-acu`: es la capacidad máxima del grupo de particiones de base de datos. Utilice un valor entre 16 y 6144 ACU.

Los siguientes parámetros son opcionales:

- `--compute-redundancy`: indica si se deben crear esperas para el grupo de particiones de base de datos. Este parámetro puede tener uno de los siguientes valores:
  - `0`: crea un grupo de particiones de base de datos sin esperas para cada partición. Este es el valor predeterminado.
  - `1`: crea un grupo de particiones de base de datos con una espera de computación en una zona de disponibilidad (AZ) diferente.
  - `2`: crea un grupo de particiones de base de datos con dos esperas de computación en dos AZ diferentes.

> **Note**
>
> Si establece la redundancia de computación en un valor distinto de cero, el número total de nodos se duplicará o triplicará. Esto generará un costo adicional.

- `--min-acu`: es la capacidad mínima de su grupo de particiones de base de datos. Debe tener al menos 16 ACU, que es el valor predeterminado.
- `--publicly-accessible|--no-publicly-accessible`: indica si se deben asignar direcciones IP de acceso público al grupo de particiones de base de datos. El acceso al grupo de particiones de base de datos está controlado por los grupos de seguridad que utiliza el clúster.

  El valor predeterminado es `--no-publicly-accessible`.

> **Note**
>
> No puede modificar esta configuración después de crear el grupo de particiones de base de datos.

En el siguiente ejemplo, se crea un grupo de particiones de base de datos en un clúster de base de datos de Aurora PostgreSQL.

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-db-cluster \
    --db-shard-group-identifier my-new-shard-group \
    --max-acu 1000
```

La salida es similar al siguiente ejemplo.

```
{
    "Status": "CREATING",
    "Endpoint": "my-db-cluster.limitless-ckifpdyyyxxx.us-east-1.rds.amazonaws.com",
    "PubliclyAccessible": false,
    "DBClusterIdentifier": "my-db-cluster",
    "MaxACU": 1000.0,
    "DBShardGroupIdentifier": "my-new-shard-group",
    "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
    "ComputeRedundancy": 0
```

```
}
```

Creación de tablas de Base de datos ilimitada de Aurora PostgreSQL

Tres tipos de tabla contienen sus datos en Base de datos ilimitada de Aurora PostgreSQL:

- Estándar: este es el tipo de tabla predeterminado en Base de datos ilimitada de Aurora PostgreSQL. Las tablas estándar se crean con el comando CREATE TABLE y en ellas se pueden ejecutar operaciones de lenguaje de descripción de datos (DDL) y de lenguaje de manipulación de datos (DML).

  Las tablas estándar no son tablas distribuidas. Se almacenan en una de las particiones elegidas internamente por el sistema.
- Particionadas: estas tablas se distribuyen en varias particiones. Los datos se dividen entre las particiones en función de los valores de las columnas designadas de la tabla. Este conjunto de columnas se denomina clave de partición.
- Referencia: estas tablas se replican en todas las particiones. Se utilizan para datos de referencia que se modifican con poca frecuencia, como catálogos de productos y códigos postales.

  Las consultas de unión entre las tablas de referencia y las tablas particionadas se pueden ejecutar en particiones, lo que elimina el movimiento innecesario de datos entre particiones y enrutadores.

Hay dos formas de crear tablas ilimitadas:

- Creación de tablas ilimitadas con variables: utilice este método cuando quiera crear tablas particionadas y de referencia nuevas.
- Conversión de tablas estándar en tablas ilimitadas: utilice este método cuando desee convertir las tablas estándar existentes en tablas particionadas y de referencia.

También ofrecemos ejemplos de esquemas para Base de datos ilimitada de Aurora PostgreSQL.

Creación de tablas ilimitadas con variables

Puede usar variables para crear tablas particionadas y de referencia configurando el modo de creación de tablas en una sesión. A continuación, las tablas que cree utilizarán este modo hasta que establezca un modo diferente.

Utilice las siguientes variables para crear tablas particionadas y de referencia:

- `rds_aurora.limitless_create_table_mode`: defina esta variable de sesión en `sharded` o `reference`. El valor predeterminado de esta variable es `standard`.
- `rds_aurora.limitless_create_table_shard_key`: defina esta variable de sesión en una matriz de nombres de columnas para utilizarlos como claves de partición. Esta variable se ignora cuando `rds_aurora.limitless_create_table_mode` no es `sharded`.

  Formatee el valor como `untyped array literal`, de forma similar a cuando inserta literales en una columna de matriz. Para obtener más información, consulte Arrays en la documentación de PostgreSQL.
- `rds_aurora.limitless_create_table_collocate_with`: defina esta variable de sesión con un nombre de tabla específico para colocar las tablas recién creadas en esa tabla.

  Si dos o más tablas están particionadas con la misma clave de partición, puede alinear (colocar) esas tablas de forma explícita. Cuando se colocan dos o más tablas, las filas de esas tablas que tengan los mismos valores de clave de partición se colocarán en la misma partición. La colocación ayuda a restringir algunas operaciones a una sola partición, lo que se traduce en un mejor rendimiento.

> **Note**
>
> Todas las claves principales y únicas deben incluir la clave de partición. Esto significa que la clave de partición es un subconjunto de la clave principal o única.
> Las tablas ilimitadas tienen algunas limitaciones. Para obtener más información, consulte Limitaciones del DDL y otra información para Base de datos ilimitada de Aurora PostgreSQL.

Temas

- Ejemplos de uso de variables para crear tablas ilimitadas
- Vistas de tabla de Base de datos ilimitada de Aurora PostgreSQL

Ejemplos de uso de variables para crear tablas ilimitadas

En los siguientes ejemplos se muestra cómo utilizar estas variables para crear tablas particionadas y de referencia.

Cree una tabla particionada llamada `items` con la clave de partición `id`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items(id int, val int, item text);
COMMIT;
```

Cree una tabla particionada llamada `items` con una clave de partición compuesta por las columnas `item_id` y `item_cat`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
CREATE TABLE items(item_id int, item_cat varchar, val int, item text);
COMMIT;
```

Cree una tabla particionada llamada `item_description` con una clave de partición compuesta por las columnas `item_id` y `item_cat`, y colóquela con la tabla `items` del ejemplo anterior.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
SET LOCAL rds_aurora.limitless_create_table_collocate_with='items';
CREATE TABLE item_description(item_id int, item_cat varchar, color_id int);
COMMIT;
```

Cree una tabla de referencia denominada `colors`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE colors(color_id int primary key, color varchar);
COMMIT;
```

Para restablecer la variable de sesión `rds_aurora.limitless_create_table_mode` a `standard`, utilice la siguiente instrucción:

```
RESET rds_aurora.limitless_create_table_mode;
```

Tras restablecer esta variable, las tablas se crean como tablas estándar, que es la versión predeterminada. Para obtener más información sobre las tablas estándar, consulte Conversión de tablas estándar en tablas ilimitadas.

Vistas de tabla de Base de datos ilimitada de Aurora PostgreSQL

Puede obtener información acerca de las tablas de Base de datos ilimitada con las siguientes vistas.

rds_aurora.limitless_tables

La vista `rds_aurora.limitless_tables` contiene información sobre las tablas ilimitadas y sus tipos.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | standard    | active       | standard    |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     |
 HASH (order_id)
         2 |     18579 | public      | customer    | active       | sharded     |
 HASH (cust_id)
(4 rows)
```

rds_aurora.limitless_table_collocations

La vista `rds_aurora.limitless_table_collocations` contiene información sobre las tablas particionadas colocadas. Por ejemplo, las tablas `orders` y `customers` están colocadas y tienen el mismo `collocation_id`. Las tablas `users` y `followers` están colocadas y tienen el mismo `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
          16002 | public      | orders
          16002 | public      | customers
          16005 | public      | users
          16005 | public      | followers
(4 rows)
```

rds_aurora.limitless_table_collocation_distributions

`rds_aurora.limitless_table_collocation_distributions` muestra la distribución de claves de cada colocación.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
          16002 |             6 | -9223372036854775808 | -4611686018427387904
          16002 |             5 | -4611686018427387904 |                    0
          16002 |             4 |                    0 |  4611686018427387904
          16002 |             3 |  4611686018427387904 |  9223372036854775807
          16005 |             6 | -9223372036854775808 | -4611686018427387904
          16005 |             5 | -4611686018427387904 |                    0
          16005 |             4 |                    0 |  4611686018427387904
          16005 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Conversión de tablas estándar en tablas ilimitadas

Puede convertir tablas estándar en tablas particionadas o de referencia. Durante la conversión, los datos se trasladan de la tabla estándar a la tabla distribuida y, a continuación, se elimina la tabla estándar de origen. Los datos se trasladan con el comando `INSERT INTO SELECT FROM`.

Contenido

- Creación de tablas particionadas
- Creación de tablas colocadas
- Creación de tablas de referencia

Creación de tablas particionadas

Para crear tablas particionadas, ejecute el procedimiento `rds_aurora.limitless_alter_table_type_sharded` en tablas estándar. Este procedimiento toma una tabla estándar y una lista de columnas y, a continuación, distribuye la tabla en cuestión con la lista de columnas como clave de partición. El procedimiento se ejecuta de forma sincrónica y adquiere un bloqueo `ACCESS EXCLUSIVE` en la tabla.

Cuando el procedimiento finalice correctamente, se eliminará la tabla estándar de origen y aparecerá una tabla particionada con el mismo nombre.

El procedimiento `rds_aurora.limitless_alter_table_type_sharded` tiene la siguiente sintaxis:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('schema.table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn']);
```

El procedimiento requiere los siguientes parámetros:

- `schema`: es el esquema de la base de datos que contiene la tabla que se va a particionar. Si no se especifica el esquema, el procedimiento utiliza la `search_path`.
- `table`: es la tabla que se va a particionar.
- `shard_key`*n*: es una matriz de columnas de tabla que se usa como clave de partición.

  Los valores clave de la partición son literales de cadena y, por lo tanto, distinguen mayúsculas de minúsculas. Si una clave de partición contiene una comilla simple ('), utilice otra comilla

simple para evitarla. Por ejemplo, si una columna de la tabla se llama `customer's id`, utilice `customer''s id` como clave de partición. No es necesario que las barras invertidas (\) y las comillas dobles (“) estén ocultas.

> **Note**
>
> Todas las claves principales y únicas deben incluir la clave de partición. Esto significa que la clave de partición es un subconjunto de la clave principal o única.
> En las tablas particionadas, la restricción CHECK no admite expresiones.
> Para obtener más información, consulte Restricciones.

Creación de una tabla particionada

El siguiente ejemplo muestra cómo crear la tabla particionada `customer` con la clave de partición `customer_id`.

1. Cree la tabla estándar.

```
CREATE TABLE customer (customer_id INT PRIMARY KEY NOT NULL, zipcode INT, email
 VARCHAR);
```

2. Convierta la tabla estándar en una tabla particionada.

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.customer',
 ARRAY['customer_id']);

postgres=> \d

                  List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
(6 rows)
```

Creación de tablas colocadas

Si dos o más tablas están particionadas con la misma clave de partición, puede alinear (colocar) esas tablas de forma explícita. Cuando se colocan dos o más tablas, las filas de esas tablas que tengan los mismos valores de clave de partición se colocarán en la misma partición. La colocación ayuda a restringir algunas operaciones a una sola partición, lo que se traduce en un mejor rendimiento.

El procedimiento `rds_aurora.limitless_alter_table_type_sharded` tiene la siguiente sintaxis:

```
postgres=> CALL
 rds_aurora.limitless_alter_table_type_sharded('schema.collocated_table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn'], 'schema.sharded_table');
```

El procedimiento requiere los siguientes parámetros:

- `schema`: es el esquema de la base de datos que contiene las tablas que se van a colocar. Si no se especifica el esquema, el procedimiento utiliza la `search_path`.
- `collocated_table`: es la tabla que se va a colocar.
- `shard_key`*n*: es una matriz de columnas de tabla que se usa como clave de partición.

  Debe utilizar la misma clave de partición que para la tabla particionada original, incluidos los mismos nombres y tipos de columna.
- `sharded_table`: es la tabla particionada con la que va a colocar la `collocated_table`.

Creación de una tabla colocada

1. Cree la primera tabla particionada siguiendo el procedimiento descrito en Creación de tablas particionadas.
2. Cree la tabla estándar para la tabla colocada.

   ```
   CREATE TABLE mytable2 (customer_id INT PRIMARY KEY NOT NULL, column1 INT, column2
    VARCHAR);
   ```
3. Convierta la tabla estándar en una tabla colocada.

   ```
   postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.mytable2',
   ```

```
ARRAY['customer_id'], 'public.customer');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | mytable2     | partitioned table | postgres_limitless
 public | mytable2_fs1 | foreign table     | postgres_limitless
 public | mytable2_fs2 | foreign table     | postgres_limitless
 public | mytable2_fs3 | foreign table     | postgres_limitless
 public | mytable2_fs4 | foreign table     | postgres_limitless
 public | mytable2_fs5 | foreign table     | postgres_limitless
(12 rows)
```

Creación de tablas de referencia

Para crear tablas de referencia, ejecute el procedimiento `rds_aurora.limitless_alter_table_type_reference` en tablas estándar. Este procedimiento replica una tabla determinada en todas las particiones del grupo de particiones de base de datos y cambia el tipo de tabla por el de referencia. El procedimiento se ejecuta de forma sincrónica y adquiere un bloqueo `ACCESS EXCLUSIVE` en la tabla.

Cuando el procedimiento finalice correctamente, se eliminará la tabla estándar de origen y aparecerá una tabla de referencia con el mismo nombre.

El procedimiento `rds_aurora.limitless_alter_table_type_reference` tiene la siguiente sintaxis:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_reference('schema.table');
```

El procedimiento almacenado requiere los siguientes parámetros:

- `schema`: es el esquema de la base de datos que contiene la tabla que se va a replicar. Si no se especifica el esquema, el procedimiento utiliza la `search_path`.

- `table`: es la tabla que se va a replicar.

> **Note**
>
> La tabla estándar a partir de la que se crea la tabla de referencia debe contar con una clave principal.
> En las tablas de referencia, la restricción CHECK no admite expresiones.
> La función anterior, `limitless_table_alter_type_reference`, ha quedado obsoleta.

Creación de una tabla de referencia

En el siguiente ejemplo se muestra cómo crear la tabla de referencia `zipcodes`.

1. Cree la tabla estándar.

```
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);
```

2. Convierta la tabla estándar en una tabla de referencia.

```
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcodes');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | zipcodes     | foreign table     | postgres_limitless
(7 rows)
```

El resultado muestra la tabla particionada `customer` y la tabla de referencia `zipcodes`.

Esquemas de ejemplo de Base de datos ilimitada de Aurora PostgreSQL

Ofrecemos los siguientes esquemas de ejemplo para Base de datos ilimitada de Aurora PostgreSQL:

- Limitless E-Commerce sample schema
- Limitless pgbench

Puede utilizar estos esquemas para crear rápidamente una base de datos de ejemplo y cargar datos en las tablas de Base de datos ilimitada de Aurora PostgreSQL. Para más información, consulte el repositorio de GitHub.

Carga de datos en Base de datos ilimitada de Aurora PostgreSQL

Puede cargar datos en tablas de Base de datos ilimitadas de Aurora PostgreSQL con el comando `COPY` o la utilidad de carga de datos.

> **Note**
>
> Puede cargar datos en tablas estándar, particionadas y de referencia.

Contenido

- Uso del comando COPY con Base de datos ilimitada de Aurora PostgreSQL
  - Uso del comando COPY para cargar datos en Base de datos ilimitada de Aurora PostgreSQL
    - Distribución de los datos en distintos archivos
  - Uso del comando COPY para copiar datos de Base de datos ilimitada en un archivo
- Uso de la utilidad de carga de datos de Base de datos ilimitada de Aurora PostgreSQL
  - Limitaciones
  - Requisitos previos
  - Preparación de la base de datos de origen
  - Preparación de la base de datos de destino
  - Creación de credenciales de la base de datos
    - Creación de las credenciales de la base de datos de origen
    - Creación de las credenciales de la base de datos de destino
  - Configuración de la autenticación de bases de datos y el acceso a los recursos mediante un script
    - Configuración del script para la utilidad de carga de datos
    - Configuración del script de configuración para la utilidad de carga de datos
    - Limpieza de recursos fallida
  - Configuración de la autenticación de bases de datos y el acceso manual a los recursos
    - Creación de la AWS KMS key administrada por el cliente
    - Creación de los secretos de base de datos
    - Creación del rol de IAM
    - Actualización de la AWS KMS key administrada por el cliente

    - Adición de las políticas de permisos del rol de IAM
- Carga de datos desde un clúster de base de datos de Aurora PostgreSQL o una instancia de base de datos de RDS para PostgreSQL
- Supervisión de carga de datos
    - Enumeración de los trabajos de carga de datos
    - Visualización de los detalles de los trabajos de carga de datos mediante el ID de trabajo
    - Supervisión del grupo de registros de Amazon CloudWatch
    - Supervisión de eventos de RDS
- Cancelación de cargas de datos

Uso del comando COPY con Base de datos ilimitada de Aurora PostgreSQL

Puede utilizar la funcionalidad \copy de la utilidad `psql` para importar y exportar datos desde Base de datos ilimitada de Aurora PostgreSQL.

Uso del comando COPY para cargar datos en Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada de Aurora PostgreSQL admite la funcionalidad \copy de la utilidad `psql` para importar datos.

En Base de datos ilimitada y en Aurora PostgreSQL no se admite lo siguiente:

- Acceso SSH directo a las instancias de base de datos: no puede copiar un archivo de datos (por ejemplo, en formato .csv) al host de la instancia de base de datos y ejecutar `COPY` desde el archivo.
- Uso de archivos locales en la instancia de base de datos: utilice `COPY ... FROM STDIN` y `COPY ... TO STDOUT`.

El comando `COPY` en PostgreSQL tiene opciones para trabajar con archivos locales (`FROM/TO`) y transmitir datos mediante una conexión entre el cliente y el servidor (`STDIN/STDOUT`). Para obtener más información, consulte COPY en la documentación de PostgreSQL.

El comando `\copy` de la utilidad `psql` de PostgreSQL funciona con los archivos locales del equipo en el que ejecuta el cliente `psql`. Este invoca el comando `COPY ... FROM STDIN` o `COPY ... FROM STDOUT` correspondiente en el servidor remoto (por ejemplo, Base de datos ilimitada) al que se conecte. Lee los datos del archivo local en `STDIN` o escribe en él desde `STDOUT`.

Distribución de los datos en distintos archivos

Los datos se almacenan en varias particiones en Base de datos ilimitada de Aurora PostgreSQL. Para acelerar la carga de datos mediante `\copy`, puede dividir los datos en varios archivos. A continuación, importe cada archivo de datos de forma independiente ejecutando comandos `\copy` separados en paralelo.

Por ejemplo, tiene un archivo de datos de entrada en formato CSV con tres millones de filas para importar. Puede dividir el archivo en fragmentos que contengan 200 000 filas (quince fragmentos) cada uno:

```
split -l200000 data.csv data_ --additional-suffix=.csv -d
```

Esto da como resultado archivos desde `data_00.csv` a `data_14.csv`. A continuación, puede importar datos con quince comandos `\copy` paralelos, por ejemplo:

```
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table from '/tmp/data_00.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_01.csv';" postgres_limitless &
...
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_13.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_14.csv';" postgres_limitless
```

Con esta técnica, se importa la misma cantidad de datos aproximadamente diez veces más rápido que con un solo comando `\copy`.

Uso del comando COPY para copiar datos de Base de datos ilimitada en un archivo

Puede usar el comando \copy para copiar datos de una tabla ilimitada en un archivo, tal como se muestra en el siguiente ejemplo:

```
postgres_limitless=> \copy test_table TO '/tmp/test_table.csv' DELIMITER ',' CSV
 HEADER;
```

Uso de la utilidad de carga de datos de Base de datos ilimitada de Aurora PostgreSQL

Aurora proporciona una utilidad para cargar datos directamente en Base de datos ilimitada desde un clúster de base de datos de Aurora PostgreSQL o desde una instancia de base de datos de RDS para PostgreSQL.

Siga los siguientes pasos para utilizar la utilidad de carga de datos:

1. Requisitos previos
2. Preparación de la base de datos de origen
3. Preparación de la base de datos de destino
4. Creación de credenciales de la base de datos
5. Uno de los siguientes:
   - Configuración de la autenticación de bases de datos y el acceso a los recursos mediante un script (recomendado)
   - Configuración de la autenticación de bases de datos y el acceso manual a los recursos
6. Carga de datos desde un clúster de base de datos de Aurora PostgreSQL o una instancia de base de datos de RDS para PostgreSQL

Limitaciones

La utilidad de carga de datos tiene las siguientes limitaciones:

- No se admiten los siguientes tipos de datos: `enum`, `ARRAY`, `BOX`, `CIRCLE`, `LINE`, `LSEG`, `PATH`, `PG_LSN`, `PG_SNAPSHOT`, `POLYGON`, `TSQUERY`, `TSVECTOR` y `TXID_SNAPSHOT`.
- Los ceros iniciales (`0`) se eliminan del tipo de datos `VARBIT` durante la carga.
- La migración de datos falla si hay claves primarias compuestas en las tablas de origen.
- La migración de datos falla cuando hay claves externas en las tablas de destino.
- No se admite la carga de datos desde clústeres de bases de datos Multi-AZ de RDS para PostgreSQL.

Requisitos previos

La utilidad de carga de datos tiene los siguientes requisitos previos:

- La base de datos de origen utiliza Aurora PostgreSQL o RDS para PostgreSQL versión 11.x y posteriores.
- La base de datos de origen se encuentra en la misma Cuenta de AWS y Región de AWS que el grupo de particiones de base de datos de destino.
- El clúster de base de datos o la instancia de base de datos de origen tienen el estado `available`.
- Las tablas de la base de datos de origen y de la base de datos ilimitada tienen los mismos nombres de tabla, nombres de columnas y tipos de datos de columna.
- Las tablas de origen y destino tienen claves principales que utilizan las mismas columnas y el mismo orden de columnas.
- Debe disponer de un entorno para conectarse a una base de datos ilimitada para ejecutar comandos de carga de datos. Los comandos disponibles son los siguientes:
  - `rds_aurora.limitless_data_load_start`
  - `rds_aurora.limitless_data_load_cancel`
- Para CDC:
  - Tanto la base de datos de origen como el grupo de particiones de base de datos de destino deben usar el mismo grupo de subredes de base de datos, el mismo grupo de seguridad de VPC y el mismo puerto de base de datos. Estas configuraciones son para las conexiones de red tanto a la base de datos de origen como a los enrutadores del grupo de particiones de base de datos.
  - Debe activar la replicación lógica en la base de datos de origen. El usuario de la base de datos de origen debe tener privilegios para leer la replicación lógica.

Preparación de la base de datos de origen

Para acceder a la base de datos de origen para cargar datos, debe permitir la entrada de tráfico de red. Siga estos pasos.

Cómo permitir el acceso del tráfico de red a la base de datos de origen

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. Navegue a la página Grupos de seguridad.
3. Elija ID de grupo de seguridad para el grupo de seguridad utilizado por la instancia o el clúster de base de datos de origen.

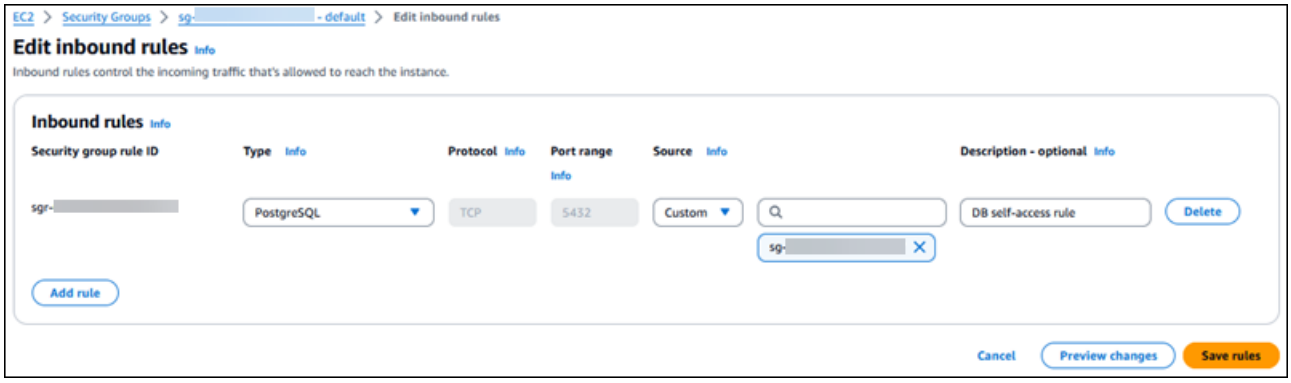
   Por ejemplo, el ID de grupo de seguridad es `sg-056a84f1712b77926`.

4. Vaya a la pestaña Reglas de entrada:

   a. Elija Editar reglas de entrada.

   b. Agregue una nueva regla de entrada para la instancia o el clúster de base de datos de origen:

      - Rango de puertos: puerto de base de datos para la base de datos de origen, normalmente `5432`
      - ID de grupo de seguridad: `sg-056a84f1712b77926` en este ejemplo

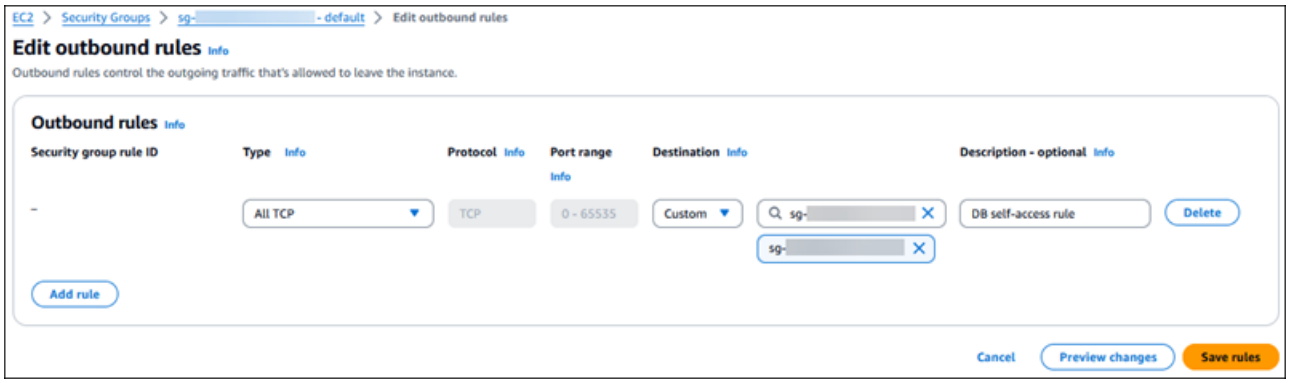


5. En la pestaña Reglas de salida:

   a. Elija Edit outbound rules.

   b. Agregue una nueva regla de salida para la instancia o el clúster de base de datos de origen:

      - Puerto de base de datos: `All traffic` (incluye los puertos `0-65535`)
      - ID de grupo de seguridad: `sg-056a84f1712b77926` en este ejemplo



6. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
7. Navegue a la página ACL de red.
8. Agregue la configuración de ACL de red predeterminada tal como se describe en la ACL de red predeterminada.

Preparación de la base de datos de destino

Siga los procedimientos descritos en Creación de tablas de Base de datos ilimitada de Aurora PostgreSQL para crear las tablas de destino en el grupo de particiones de base de datos.

Las tablas de destino deben tener los mismos esquemas, nombres de tabla y claves principales que las tablas de origen.

Creación de credenciales de la base de datos

Debe crear usuarios de bases de datos en las bases de datos de origen y destino y conceder los privilegios necesarios a los usuarios. Para obtener más información, consulte CREATE USER y GRANT en la documentación de PostgreSQL.

Creación de las credenciales de la base de datos de origen

El usuario de la base de datos de origen se pasa en el comando para iniciar la carga. El usuario debe tener privilegios para ejecutar la replicación desde la base de datos de origen.

1. Utilice el usuario maestro de la base de datos (u otro usuario con el rol `rds_superuser`) para crear un usuario de la base de datos de origen con privilegios `LOGIN`.

```
CREATE USER source_db_username WITH PASSWORD 'source_db_user_password';
```

2. Otorgue el rol `rds_superuser` al usuario de la base de datos de origen.

```
GRANT rds_superuser to source_db_username;
```

3. Si utiliza el modo `full_load_and_cdc`, otorgue el rol `rds_replication` al usuario de la base de datos de origen. El rol de `rds_replication` concede permisos para administrar ranuras lógicas y para transmitir datos mediante ranuras lógicas.

```
GRANT rds_replication to source_db_username;
```

Creación de las credenciales de la base de datos de destino

El usuario de la base de datos de destino debe tener permiso para escribir en las tablas de destino del grupo de particiones de base de datos.

1. Utilice el usuario maestro de la base de datos (u otro usuario con el rol `rds_superuser`) para crear un usuario de la base de datos de destino con privilegios `LOGIN`.

```
CREATE USER destination_db_username WITH PASSWORD 'destination_db_user_password';
```

2. Otorgue el rol `rds_superuser` al usuario de la base de datos de destino.

```
GRANT rds_superuser to destination_db_username;
```

Configuración de la autenticación de bases de datos y el acceso a los recursos mediante un script

El script de configuración crea una AWS KMS key administrada por el cliente, un rol de AWS Identity and Access Management (IAM) y dos secretos de AWS Secrets Manager.

Para configurar el script, siga los pasos descritos a continuación:

1. Asegúrese de tener la AWS CLI instalada y configurada con sus credenciales de la Cuenta de AWS.
2. Instale el procesador de línea de comandos JSON `jq`. Para obtener más información, consulte jqlang/jq.
3. Copie el archivo data_loading_script.zip en su ordenador y extraiga el archivo `data_load_aws_setup_script.sh`.
4. Edite el script para reemplazar las variables de marcador de posición por los valores adecuados para lo siguiente:
   - su Cuenta de AWS,
   - la Región de AWS,
   - las credenciales de la base de datos de origen,
   - las credenciales de la base de datos de destino.
5. Abra una terminal nueva en el equipo y ejecute el siguiente comando:

```
bash ./data_load_aws_setup_script.sh
```

Configuración del script para la utilidad de carga de datos

Aquí proporcionamos el texto del archivo `data_load_aws_setup_script.sh` como referencia.

```
#!/bin/bash
# Aurora Limitless data loading - AWS resources setup script #
# Set up the account credentials in advance. #
# Update the following script variables. #

###################################
#### Start of variable section ####

ACCOUNT_ID="12-digit_AWS_account_ID"
```

```
REGION="AWS_Region"
DATE=$(date +'%m%d%H%M%S')
RANDOM_SUFFIX="${DATE}"
SOURCE_SECRET_NAME="secret-source-${DATE}"
SOURCE_USERNAME="source_db_username"
SOURCE_PASSWORD="source_db_password"
DESTINATION_SECRET_NAME="secret-destination-${DATE}"
DESTINATION_USERNAME="destination_db_username"
DESTINATION_PASSWORD="destination_db_password"
DATA_LOAD_IAM_ROLE_NAME="aurora-data-loader-${RANDOM_SUFFIX}"
TMP_WORK_DIR="./tmp_data_load_aws_resource_setup/"

#### End of variable section ####
################################

# Main logic start
echo "DATE - [${DATE}]"
echo "RANDOM_SUFFIX - [${RANDOM_SUFFIX}]"
echo 'START!'

mkdir -p $TMP_WORK_DIR

# Create the symmetric KMS key for encryption and decryption.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_key_response.txt"
aws kms create-key --region $REGION | tee $TMP_FILE_PATH
KMS_KEY_ARN=$(cat $TMP_FILE_PATH | jq -r '.KeyMetadata.Arn')
aws kms create-alias \
    --alias-name alias/"${DATA_LOAD_IAM_ROLE_NAME}-key" \
    --target-key-id $KMS_KEY_ARN \
    --region $REGION

# Create the source secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_source_secret_response.txt"
aws secretsmanager create-secret \
    --name $SOURCE_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$SOURCE_USERNAME\",\"password\":\"$SOURCE_PASSWORD
\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
SOURCE_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the destination secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_destination_secret_response.txt"
```

```
aws secretsmanager create-secret \
    --name $DESTINATION_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$DESTINATION_USERNAME\",\"password\":
\"$DESTINATION_PASSWORD\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
DESTINATION_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the RDS trust policy JSON file.
# Use only rds.amazonaws.com for RDS PROD use cases.
TRUST_POLICY_PATH="${TMP_WORK_DIR}rds_trust_policy.json"
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}' > $TRUST_POLICY_PATH

# Create the IAM role.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_iam_role_response.txt"
aws iam create-role \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --assume-role-policy-document "file://${TRUST_POLICY_PATH}" \
    --tags Key=assumer,Value=aurora_limitless_table_data_load \
    --region $REGION \
    | tee $TMP_FILE_PATH
IAM_ROLE_ARN=$(cat $TMP_FILE_PATH | jq -r '.Role.Arn')

# Create the permission policy JSON file.
PERMISSION_POLICY_PATH="${TMP_WORK_DIR}data_load_permission_policy.json"
permission_json_policy=$(cat &lt;&lt;EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "$SOURCE_SECRET_ARN",
                "$DESTINATION_SECRET_ARN"
            ]
        },
        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "$KMS_KEY_ARN"
        },
        {
            "Sid": "RdsPermissions",
```

```
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $permission_json_policy > $PERMISSION_POLICY_PATH

# Add the inline policy.
aws iam put-role-policy \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --policy-name aurora-limitless-data-load-policy \
    --policy-document "file://${PERMISSION_POLICY_PATH}" \
    --region $REGION

# Create the key policy JSON file.
KEY_POLICY_PATH="${TMP_WORK_DIR}data_load_key_policy.json"
key_json_policy=$(cat &lt;&lt;EOF
{
    "Id": "key-aurora-limitless-data-load-$RANDOM_SUFFIX",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "$IAM_ROLE_ARN"
            },
            "Action": [
                "kms:Decrypt",
```

```
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $key_json_policy > $KEY_POLICY_PATH

# Add the key policy.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_put_key_policy_response.txt"
sleep 10 # sleep 10 sec for IAM role ready
aws kms put-key-policy \
    --key-id $KMS_KEY_ARN \
    --policy-name default \
    --policy "file://${KEY_POLICY_PATH}" \
    --region $REGION \
    | tee $TMP_FILE_PATH

echo 'DONE!'

echo "ACCOUNT_ID : [${ACCOUNT_ID}]"
echo "REGION : [${REGION}]"
echo "RANDOM_SUFFIX : [${RANDOM_SUFFIX}]"
echo "IAM_ROLE_ARN : [${IAM_ROLE_ARN}]"
echo "SOURCE_SECRET_ARN : [${SOURCE_SECRET_ARN}]"
echo "DESTINATION_SECRET_ARN : [${DESTINATION_SECRET_ARN}]"

# Example of a successful run:
# ACCOUNT_ID : [012345678912]
# REGION : [ap-northeast-1]
# RANDOM_SUFFIX : [0305000703]
# IAM_ROLE_ARN : [arn:aws:iam::012345678912:role/aurora-data-loader-0305000703]
# SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-source-0305000703-yQDtow]
# DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-destination-0305000703-5d5Jy8]

# If you want to manually clean up failed resource,
# please remove them in the following order:
# 1. IAM role.
```

```
 # aws iam delete-role-policy --role-name Test-Role --policy-name ExamplePolicy --
region us-east-1
# aws iam delete-role --role-name Test-Role --region us-east-1
# 2. Source and destination secrets.
# aws secretsmanager delete-secret --secret-id MyTestSecret --force-delete-without-
recovery --region us-east-1
# 3. KDM key.
# aws kms schedule-key-deletion --key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab --pending-window-in-days 7
 --region us-east-1
```

Configuración del script de configuración para la utilidad de carga de datos

El siguiente ejemplo muestra los resultados de una ejecución correcta del script.

```
% bash ./data_load_aws_setup_script.sh
DATE - [0305000703]
RANDOM_SUFFIX - [0305000703]
START!
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:ap-
northeast-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-03-05T00:07:49.852000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow",
    "Name": "secret-source-0305000703",
```

```
    "VersionId": "a017bebe-a71b-4220-b923-6850c2599c26"
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
destination-0305000703-5d5Jy8",
    "Name": "secret-destination-0305000703",
    "VersionId": "32a1f989-6391-46b1-9182-f65d242f5eb6"
}
{
    "Role": {
        "Path": "/",
        "RoleName": "aurora-data-loader-0305000703",
        "RoleId": "AROAYPX63ITQOYORQSC6U",
        "Arn": "arn:aws:iam::123456789012:role/aurora-data-loader-0305000703",
        "CreateDate": "2024-03-05T00:07:54+00:00",
        "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "rds.amazonaws.com"
                        ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "assumer",
                "Value": "aurora_limitless_table_data_load"
            }
        ]
    }
}
DONE!
ACCOUNT_ID : [123456789012]
REGION : [ap-northeast-1]
RANDOM_SUFFIX : [0305000703]
IAM_ROLE_ARN : [arn:aws:iam::123456789012:role/aurora-data-loader-0305000703]
SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow]
```

```
DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:123456789012:secret:secret-destination-0305000703-5d5Jy8]
```

Limpieza de recursos fallida

Si desea limpiar los recursos que han fallado de forma manual, elimínelos en el siguiente orden:

1. Rol de IAM, por ejemplo:

```
aws iam delete-role-policy \
--role-name Test-Role \
--policy-name ExamplePolicy

aws iam delete-role \
--role-name Test-Role
```

2. Secretos de origen y destino, por ejemplo:

```
aws secretsmanager delete-secret \
--secret-id MyTestSecret \
--force-delete-without-recovery
```

3. Clave de KMS, por ejemplo:

```
aws kms schedule-key-deletion \
--key-id arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab
 \
--pending-window-in-days 7
```

A continuación, puede volver a intentar el script.

Configuración de la autenticación de bases de datos y el acceso manual a los recursos

El proceso manual para configurar la autenticación de la base de datos y el acceso a los recursos consta de los siguientes pasos:

1. Creación de la AWS KMS key administrada por el cliente
2. Adición de las políticas de permisos del rol de IAM
3. Creación de los secretos de base de datos
4. Creación del rol de IAM
5. Actualización de la AWS KMS key administrada por el cliente

Este proceso es opcional y realiza las mismas tareas que en Configuración de la autenticación de bases de datos y el acceso a los recursos mediante un script. Se recomienda utilizar el script.

Creación de la AWS KMS key administrada por el cliente

Siga los procedimientos que se describen en Creating symmetric encryption keys para crear una clave de KMS administrada por el cliente. También puede utilizar una clave existente si cumple estos requisitos.

Creación de una clave de KMS administrada por el cliente

1. Inicie sesión en la Consola de administración de AWS y abra la consola de AWS KMS en https://console.aws.amazon.com/kms.
2. Navegue a la página Claves administradas por el cliente.
3. Elija Crear clave.
4. En la página Configurar clave:
    a. En Tipo de clave, seleccione Simétrica.
    b. En Uso de claves, seleccione Cifrar y descifrar.
    c. Elija Siguiente.
5. En la página Añadir etiquetas, introduzca un alias, como **`limitless`**, y elija Siguiente.
6. En la página Definir permisos de administración de claves, asegúrese de que la casilla de verificación Permitir que los administradores de claves eliminen esta clave esté seleccionada y elija Siguiente.
7. En la página Definir permisos de uso de claves, elija Siguiente.

8. En la página Revisar, elija Finalizar.

   La política de claves se actualiza más adelante.

Registre el nombre de recurso de Amazon (ARN) de la clave de KMS que va a usar en Adición de las políticas de permisos del rol de IAM.

Para obtener más información sobre el uso de AWS CLI para crear la clave de KMS administrada por el cliente, consulte create-key y create-alias.

Creación de los secretos de base de datos

Para permitir que la utilidad de carga de datos acceda a las tablas de la base de datos de origen y destino, debe crear dos secretos en AWS Secrets Manager: uno para la base de datos de origen y otro para la base de datos de destino. Estos secretos almacenan los nombres de usuario y las contraseñas para acceder a las bases de datos de origen y destino.

Siga los procedimientos descritos en Create an AWS Secrets Manager secret o para crear los secretos del par clave-valor.

Creación de los secretos de base de datos

1. Abra la consola de Secrets Manager enhttps://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.
3. Se abre la página Elija tipo de secreto.

   a. En Tipo de secreto, seleccione Otro tipo de secreto.

   b. Para Pares clave-valor, elija la pestaña Texto no cifrado.

   c. Introduzca el siguiente código JSON, dónde *sourcedbreader* y *sourcedbpassword* son las credenciales del usuario de la base de datos de origen de Creación de las credenciales de la base de datos de origen.

   ```
   {
       "username":"sourcedbreader",
       "password":"sourcedbpassword"
   }
   ```

   d. Para Clave de cifrado, elija la clave de KMS que ha creado en Creación de la AWS KMS key administrada por el cliente, por ejemplo, `limitless`.

e. Elija Siguiente.

4. En la página Configurar secreto, en Nombre del secreto, introduzca **`source_DB_secret`** y, luego, elija Siguiente.
5. En la página Configurar la rotación. Opcional, elija Siguiente.
6. En la página Revisar, elija Almacenar.
7. Repita el procedimiento para el secreto de la base de datos de destino:

   a. Introduzca el siguiente código JSON, dónde *`destinationdbwriter`* y *`destinationdbpassword`* son las credenciales del usuario de la base de datos de destino de Creación de las credenciales de la base de datos de destino.

   ```
   {
       "username":"destinationdbwriter",
       "password":"destinationdbpassword"
   }
   ```

   b. Escriba un Nombre del secreto, como **`destination_DB_secret`**.

Registre los ARN de los secretos que desee utilizar en Adición de las políticas de permisos del rol de IAM.

Creación del rol de IAM

La carga de datos requiere que proporcione acceso a los recursos de AWS. Para proporcionar acceso, debe crear el rol de IAM `aurora-data-loader` siguiendo los procedimientos descritos en Crear un rol para delegar permisos a un usuario de IAM.

Creación del rol de IAM

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Vaya a la página Roles.
3. Seleccione Crear rol.
4. En la página Seleccionar entidad de confianza:

   a. En Tipo de entidad de confianza, elija Política de confianza personalizada.

   b. Introduzca el siguiente código JSON para la política de confianza personalizada:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

c. Elija Siguiente.

5. En la página Agregar permisos, elija Siguiente.
6. En la página Asignar nombre, revisar y crear:

   a. En Nombre del rol, escriba **`aurora-data-loader`** u otro nombre que prefiera.

   b. Elija Agregar etiqueta e introduzca la siguiente etiqueta:

   - Clave: **`assumer`**
   - Valor: **`aurora_limitless_table_data_load`**

> **Important**
>
> Base de datos ilimitada de Aurora PostgreSQL solo puede asumir un rol de IAM que tenga esta etiqueta.

   c. Seleccione Crear rol.

Actualización de la AWS KMS key administrada por el cliente

Siga los procedimientos que se explican en Changing a key policy para añadir el rol de IAM `aurora-data-loader` a la política de claves predeterminada.

Adición del rol de IAM a la política de claves

1. Inicie sesión en la Consola de administración de AWS y abra la consola de AWS KMS en https://console.aws.amazon.com/kms.
2. Navegue a la página Claves administradas por el cliente.
3. Elija la clave de KMS que ha creado en Creación de la AWS KMS key administrada por el cliente, por ejemplo, `limitless`.
4. En la pestaña Política de claves, para Usuarios de claves, elija Agregar.
5. En la ventana Agregar usuarios de claves, seleccione el nombre del rol de IAM en el que ha creado en Creación del rol de IAM, por ejemplo, aurora-data-loader.
6. Elija Agregar.

Adición de las políticas de permisos del rol de IAM

Debe añadir políticas de permisos al rol de IAM que ha creado. Esto permite a la utilidad de carga de datos de Base de datos ilimitada de Aurora PostgreSQL acceder a los recursos de AWS relacionados para crear conexiones de red y recuperar los secretos de las credenciales de base de datos de origen y destino.

Para obtener más información, consulte Modificación de un rol.

Adición de las políticas de permisos

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Vaya a la página Roles.
3. Elija el rol de IAM que ha creado en Creación del rol de IAM, por ejemplo, aurora-data-loader.
4. En la pestaña Permisos, elija Agregar permisos para Políticas de permisos y, a continuación, Crear política insertada.
5. En la página Especificar permisos, seleccione el editor JSON.
6. Copie y pegue la siguiente plantilla en el editor de JSON y reemplace los marcadores de posición por los ARN de los secretos de la base de datos y la clave de KMS.

   JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:source_DB_secret-ABC123",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:destination_DB_secret-456DEF"
            ]
        },        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/aa11bb22-
####-####-####-fedcba123456"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Compruebe si hay errores y corríjalos.
8. Elija Siguiente.
9. En la página Revisar y crear, ingrese un Nombre de la política como **`data_loading_policy`** y, luego, elija Crear política.

Carga de datos desde un clúster de base de datos de Aurora PostgreSQL o una instancia de base de datos de RDS para PostgreSQL

Tras completar la configuración de los recursos y la autenticación, conecte con el punto de conexión del clúster y llame al procedimiento `rds_aurora.limitless_data_load_start` almacenado desde una base de datos ilimitada, como `postgres_limitless`. La base de datos ilimitada es una base de datos del grupo de particiones de base de datos a la que desea migrar los datos.

Esta función se conecta de forma asíncrona en segundo plano a la base de datos de origen especificada en el comando, lee los datos del origen y los carga en las particiones. Para lograr un mejor rendimiento, los datos se cargan mediante subprocesos paralelos. La función recupera una instantánea de tabla en un momento dado mediante la ejecución de un comando `SELECT` para leer los datos de las tablas proporcionadas en el comando.

Puede cargar datos en tablas estándar, particionadas y de referencia.

Puede cargar datos en el nivel de base de datos, esquema o tabla en las llamadas a `rds_aurora.limitless_data_load_start`.

- Base de datos: puede cargar una base de datos a la vez en cada llamada, sin límite en el recuento de esquemas o tablas dentro de la base de datos.
- Esquema: puede cargar un máximo de quince esquemas en cada llamada, sin límite en el número de tablas dentro de cada esquema.
- Tabla: puede cargar un máximo de quince tablas en cada llamada.

> **Note**
>
> Esta característica no utiliza instantáneas de Amazon RDS ni aísla la base de datos en un momento dado. Para mantener la coherencia entre las tablas, recomendamos clonar la base de datos de origen y dirigirla hacia esa base de datos clonada como origen.

El procedimiento almacenado tiene la siguiente sintaxis:

```
CALL rds_aurora.limitless_data_load_start('source_type',
    'source_DB_cluster_or_instance_ID',
    'source_database_name',
    'streaming_mode',
    'data_loading_IAM_role_arn',
    'source_DB_secret_arn',
    'destination_DB_secret_arn',
    'ignore_primary_key_conflict_boolean_flag',
    'is_dry_run',
    (optional parameter) schemas/tables => ARRAY['name1', 'name2', ...]);
```

Los parámetros de entrada son los siguientes:

- `source_type`: es el tipo de origen `aurora_postgresql` o `rds_postgresql`.
- `source_DB_cluster_or_instance_ID`: es el identificador de clúster de Base de datos ilimitada de Aurora PostgreSQL de origen o el identificador de instancia de base de datos de RDS para PostgreSQL.
- `source_database_name`: es el nombre de la base de datos de origen, como *`postgres`*.
- `streaming_mode`: indica si se debe incluir la captura de datos de cambios (CDC) `full_load` o `full_load_and_cdc`.
- `data_loading_IAM_role_arn`: es el nombre de recurso de Amazon (ARN) del rol de IAM para `aurora-data-loader`.
- `source_DB_secret_arn`: es el ARN secreto de la base de datos de origen.
- `destination_DB_secret_arn`: es el ARN secreto de la base de datos de destino.
- `ignore_primary_key_conflict_boolean_flag`: indica si se debe continuar si surge un conflicto con la clave principal.
  - Si se establece en `true`, la carga de datos ignora los nuevos cambios en las filas con un conflicto de clave principal.
  - Si se establece en `false`, la carga de datos sobrescribe las filas existentes en las tablas de destino cuando se produce un conflicto de clave principal.
- `is_dry_run`: indica si se debe comprobar si el trabajo de carga de datos se puede conectar a las bases de datos de origen y destino.
  - Si se establece en `true`, prueba las conexiones sin cargar datos.
  - Si se establece en `false`, carga los datos.

- (opcional) `schemas` o `tables`: una matriz de esquemas o tablas para cargar. Puede especificar cualquiera de los siguientes:
  - Una lista de tablas con el formato `tables => ARRAY['schema1.table1', 'schema1.table2', 'schema2.table1', ...]`
  - Una lista de esquemas con el formato `schemas => ARRAY['schema1', 'schema2', ...]`

  Si no incluye este parámetro, se migra toda la base de datos de origen especificada.

El parámetro de salida es el ID del trabajo con un mensaje.

En el ejemplo siguiente se muestra cómo utilizar el procedimiento almacenado `rds_aurora.limitless_data_load_start` para cargar datos desde un clúster de base de datos de Aurora PostgreSQL.

```
CALL rds_aurora.limitless_data_load_start('aurora_postgresql',
    'my-db-cluster',
    'postgres',
    'full_load_and_cdc',
    'arn:aws:iam::123456789012:role/aurora-data-loader-8f2c66',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-source-8f2c66-EWrr0V',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-destination-8f2c66-
d04fbD',
    'true',
    'false',
    tables => ARRAY['public.customer', 'public.order', 'public.orderdetails']);

INFO: limitless data load job id 1688761223647 is starting.
```

Supervisión de carga de datos

Base de datos ilimitada de Aurora PostgreSQL ofrece varias formas de supervisar los trabajos de carga de datos:

- Enumeración de los trabajos de carga de datos
- Visualización de los detalles de los trabajos de carga de datos mediante el ID de trabajo
- Supervisión del grupo de registros de Amazon CloudWatch
- Supervisión de eventos de RDS

Enumeración de los trabajos de carga de datos

Puede conectarse al punto de conexión del clúster y usar la vista `rds_aurora.limitless_data_load_jobs` para enumerar los trabajos de carga de datos.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_jobs LIMIT 6;

    job_id     |  status   | message |     source_db_identifier      | source_db_name |
 full_load_complete_time |
   progress_details                                                                  |
     start_time       |   last_updated_time    |  streaming_mode   | source_engine_type
 | ignore_primary_key_conflict | is_dryrun
--------------+-----------+---------+-------------------------------+---------------
+-------------------------
+-------------------------------------------------------------------------------------------------------------------------------
+-----------------------+-----------------------+------------------
+--------------------+-----------------------------+-----------
 1725697520693 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:48:15+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "9
 of 9 tables loaded", "COMPLETED_AT": "2024/09/07 08:48:15+00", "RECORDS_MIGRATED":
 600003}} | 2024-09-07 08:47:13+00 | 2024-09-07 08:48:15+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696114225 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:24:20+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "3
 of 3 tables loaded", "COMPLETED_AT": "2024/09/07 08:24:20+00", "RECORDS_MIGRATED":
 200001}} | 2024-09-07 08:23:56+00 | 2024-09-07 08:24:20+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696067630 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:23:45+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "6
 of 6 tables loaded", "COMPLETED_AT": "2024/09/07 08:23:45+00", "RECORDS_MIGRATED":
```

```
 400002}} | 2024-09-07 08:23:10+00 | 2024-09-07 08:23:45+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725694221753 | CANCELED  |         | persistent-kdm-auto-source-01 | postgres
    |                         | {}

           | 2024-09-07 07:31:18+00 | 2024-09-07 07:51:49+00 | full_load_and_cdc |
 aurora_postgresql  | t                           | f
 1725691698210 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:51+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:51+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:42+00 | 2024-09-07 07:10:52+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725691695049 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:48+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:48+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:41+00 | 2024-09-07 07:10:48+00 | full_load         |
 aurora_postgresql  | t                           | f
(6 rows)
```

Los registros de trabajos se eliminan pasados noventa días.

Visualización de los detalles de los trabajos de carga de datos mediante el ID de trabajo

Si conoce el ID de un trabajo, puede conectarse al punto de conexión del clúster y usar la vista `rds_aurora.limitless_data_load_job_details` para ver los detalles de ese trabajo de carga de datos, incluidos el nombre de la tabla, el estado del trabajo y el número de filas cargadas. Puede obtener el ID de trabajo en las respuestas a las funciones de inicio de carga de datos o en la vista `rds_aurora.limitless_data_load_jobs`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_job_details WHERE
 job_id='1725696114225';

job_id        | destination_table_name | destination_schema_name | start_time
      | status    | full_load_rows | full_load_total_rows | full_load_complete_time |
 cdc_insert | cdc_update | cdc_delete
--------------+------------------------+-------------------------
+-------------------------+-----------+----------------+---------------------
+-------------------------+------------+------------+------------
1725696114225 | standard_1             | public                  | 2024-09-07
 08:23:57+00 | COMPLETED | 100000         | 100000              | 2024-09-07
 08:24:08+00  | 0          | 0          | 0
```

```
1725696114225 | standard_2             | public                  | 2024-09-07
 08:24:08+00 | COMPLETED | 100000        | 100000               | 2024-09-07
 08:24:17+00  | 0          | 0          | 0
1725696114225 | standard_3             | public                  | 2024-09-07
 08:24:18+00 | COMPLETED | 1             | 1                    | 2024-09-07
 08:24:20+00  | 0          | 0          | 0
1725696114225 | standard_4             | public                  | 2024-09-07
 08:23:58+00 | PENDING   | 0             | 0                    |
    | 0          | 0          | 0
(4 rows)
```

Los registros de trabajos se eliminan pasados noventa días.

Supervisión del grupo de registros de Amazon CloudWatch

Cuando el estado del trabajo de carga de datos cambie a `RUNNING`, puede comprobar el progreso del tiempo de ejecución mediante Registros de Amazon CloudWatch.

Supervisión de los flujos de registros de CloudWatch

Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.

1. Vaya a Registros y, a continuación, a Grupos de registros.
2. Elija el grupo de registro /aws/rds/aurora-limitless-database.
3. Busque el flujo de registro de su trabajo de carga de datos con job_id.

   El flujo de registro tiene el patrón Data-Load-Job***job_id***.
4. Elija el flujo de registro para ver los eventos de registro.

Cada flujo de registro muestra los eventos que contienen el estado del trabajo y el número de filas cargadas en las tablas de destino de Base de datos ilimitada de Aurora PostgreSQL. Si se produce un error en un trabajo de carga de datos, también se crea un registro de errores que muestra el estado del error y el motivo.

Los registros de trabajos se eliminan pasados noventa días.

Supervisión de eventos de RDS

El trabajo de carga de datos también publica eventos de RDS, por ejemplo, cuando un trabajo concluye correctamente, falla o se cancela. Puede ver los eventos desde la base de datos de destino.

Para obtener más información, consulte Eventos de grupo de particiones de bases de datos.

Cancelación de cargas de datos

Para cancelar un trabajo de carga de datos, llame al procedimiento almacenado `rds_aurora.limitless_data_load_cancel` con el ID del trabajo como parámetro de entrada. Llame al procedimiento almacenado desde la misma base de datos del grupo de particiones de base de datos desde el que ha iniciado el trabajo de carga de datos específico. Por ejemplo:

```
CALL rds_aurora.limitless_data_load_cancel(12345);

INFO: limitless data load job with id 12345 is canceling without rollback.
```

No puede cancelar un trabajo de carga de datos que no existe o que no se esté ejecutando en el mismo grupo de particiones de base de datos.

La utilidad de carga de datos de Base de datos ilimitada de Aurora PostgreSQL deja los datos cargados en las tablas de destino sin revertirlos, tal como se muestra en la respuesta. Si no desea conservar los datos cargados, le recomendamos truncar las tablas de destino.

Consulta de Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada de Aurora PostgreSQL es compatible con la sintaxis de PostgreSQL para las consultas. Puede consultar su Base de datos ilimitada con `psql` o cualquier otra utilidad de conexión que funcione con PostgreSQL. Para ejecutar consultas, debe conectarse al punto de conexión ilimitado, tal como se muestra en Conexión a su clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL.

Todas las consultas `SELECT` de PostgreSQL son compatibles con Base de datos ilimitada de Aurora PostgreSQL. Sin embargo, las consultas se realizan en dos capas:

1. Enrutador al que el cliente envía la consulta
2. Particiones donde se encuentran los datos

El rendimiento depende de que se consulte la base de datos de una forma que le permita lograr un alto grado de procesamiento simultáneo de distintas consultas en distintas particiones. Las consultas se analizan primero en la capa de transacciones distribuidas (enrutador). Antes de planificar la ejecución de la consulta, hay una fase de análisis para identificar la ubicación de todas las relaciones que participan en la consulta. Si todas las relaciones son tablas particionadas con una clave de partición filtrada en la misma partición o tablas de referencia, la planificación de consultas se omite en la capa del enrutador y se pasa a la partición para su planificación y ejecución. Este proceso reduce el número de viajes de ida y vuelta entre los distintos nodos (enrutador y partición) y, en la mayoría de los casos, mejora el rendimiento. Para obtener más información, consulte Consultas de una sola partición en Base de datos ilimitada de Aurora PostgreSQL.

> **Note**
>
> Puede haber casos específicos, como un producto cartesiano (unión cruzada), en los que la consulta tenga un mejor rendimiento al recuperar los datos por separado de la partición.

Para obtener más información sobre los planes de ejecución de consultas, consulte EXPLAIN en la Referencia sobre Base de datos ilimitada de Aurora PostgreSQL. Para obtener información general sobre las consultas, vea Queries en la documentación de PostgreSQL.

Temas

- Consultas de una sola partición en Base de datos ilimitada de Aurora PostgreSQL

- Consultas distribuidas en Base de datos ilimitada de Aurora PostgreSQL
- Seguimiento de consultas distribuidas en Base de datos ilimitada de Aurora PostgreSQL
- Interbloqueos distribuidos en Base de datos ilimitada de Aurora PostgreSQL

Consultas de una sola partición en Base de datos ilimitada de Aurora PostgreSQL

Una consulta de una sola partición es una consulta que se puede ejecutar directamente en una partición y, al mismo tiempo, mantener la semántica ACID de SQL. Cuando el planificador de consultas del enrutador encuentra una consulta de este tipo, la detecta y envía toda la consulta SQL a la partición correspondiente.

Esta optimización reduce el número de recorridos de ida y vuelta a través de la red entre el enrutador y la partición, lo que mejora el rendimiento. Actualmente, esta optimización se realiza para las consultas `INSERT`, `SELECT`, `UPDATE` y `DELETE`.

Temas

- Ejemplos de consultas de una sola partición
- Restricciones para las consultas de una sola partición
- Uniones totalmente cualificadas (explícitas)
- Definición de una clave de partición activa

Ejemplos de consultas de una sola partición

En los siguientes ejemplos, tenemos la tabla particionada `customers`, con la clave de partición `customer_id` y la tabla de referencia `zipcodes`.

SELECT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM customers WHERE
 customer_id = 100;

                       QUERY PLAN
-------------------------------------------------------------
 Foreign Scan
   Output: customer_id, other_id, customer_name, balance
```

```
   Remote SQL:  SELECT customer_id,
     other_id,
     customer_name,
     balance
    FROM public.customers
   WHERE (customer_id = 100)
 Single Shard Optimized
(9 rows)
```

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM orders
    LEFT JOIN zipcodes ON orders.zipcode_id = zipcodes.zipcode_id
    WHERE customer_id = 11;

                                                QUERY PLAN

-------------------------------------------------------------------------------------------------
 Foreign Scan
   Output: customer_id, order_id, zipcode_id, customer_name, balance,
 zipcodes.zipcode_id, zipcodes.city
   Remote SQL:  SELECT orders.customer_id,
     orders.order_id,
     orders.zipcode_id,
     orders.customer_name,
     orders.balance,
     zipcodes.zipcode_id,
     zipcodes.city
    FROM (public.orders
      LEFT JOIN public.zipcodes ON ((orders.zipcode_id = zipcodes.zipcode_id)))
   WHERE (orders.customer_id = 11)
 Single Shard Optimized
(13 rows)
```

INSERT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) INSERT INTO customers
    (customer_id, other_id, customer_name, balance)
    VALUES (1, 10, 'saikiran', 1000);

                      QUERY PLAN
------------------------------------------------------
 Insert on public.customers
   ->  Result
         Output: 1, 10, 'saikiran'::text, '1000'::real
```

```
 Single Shard Optimized
(4 rows)
```

UPDATE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) UPDATE orders SET balance = balance +
 100
    WHERE customer_id = 100;

                                        QUERY PLAN

-----------------------------------------------------------------------------------------------
 Update on public.orders
   Foreign Update on public.orders_fs00002 orders_1
   ->  Foreign Update
         Remote SQL:  UPDATE public.orders SET balance = (balance + (100)::double
 precision)
   WHERE (customer_id = 100)
 Single Shard Optimized
(6 rows)
```

DELETE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) DELETE FROM orders
    WHERE customer_id = 100 and balance = 0;

                             QUERY PLAN
---------------------------------------------------------------------
 Delete on public.orders
   Foreign Delete on public.orders_fs00002 orders_1
   ->  Foreign Delete
         Remote SQL:  DELETE FROM public.orders
   WHERE ((customer_id = 100) AND (balance = (0)::double precision))
 Single Shard Optimized
(6 rows)
```

Restricciones para las consultas de una sola partición

Las consultas de una sola partición tienen las siguientes restricciones:

Funciones

Si una consulta de una sola partición contiene una función, la consulta solo es apta para la optimización de una sola partición si se cumple una de las siguientes condiciones:

- La función es inmutable. Para obtener más información, consulte Volatilidad de las funciones.
- La función es mutable, pero está registrada en la vista `rds_aurora.limitless_distributed_functions`. Para obtener más información, consulte Distribución de funciones.

Vistas

Si una consulta contiene una o más vistas, la optimización de una sola partición estará deshabilitada para la consulta si cumple una de las siguientes condiciones:

- Cualquier vista tiene el atributo `security_barrier`.
- Los objetos utilizados en la consulta requieren varios privilegios de usuario. Por ejemplo, una consulta contiene dos vistas y las vistas se ejecutan con dos usuarios diferentes.

```
CREATE VIEW v1 AS SELECT customer_name FROM customers c WHERE c.customer_id =  1;
CREATE VIEW v2 WITH (security_barrier) AS SELECT customer_name FROM customers c
 WHERE c.customer_id =  1;

postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v1;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Output: customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 c  (cost=0.00..24.12 rows=6 width=32)
           Output: c.customer_name
           Filter: (c.customer_id = 1)
         Query Identifier: -6005737533846718506
   Remote SQL:  SELECT customer_name
    FROM ( SELECT c.customer_name
             FROM public.customers c
            WHERE (c.customer_id = 1)) v1
 Query Identifier: -5754424854414896228
(12 rows)


postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v2;
                                        QUERY PLAN
```

```
-------------------------------------------------------------------------------------------
 Foreign Scan on public.customers_fs00001 c  (cost=100.00..128.41 rows=7 width=32)
   Output: c.customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 customers  (cost=0.00..24.12 rows=6
 width=32)
           Output: customers.customer_name
           Filter: (customers.customer_id = 1)
         Query Identifier: 4136563775490008117
   Remote SQL: SELECT customer_name FROM public.customers WHERE ((customer_id = 1))
 Query Identifier: 5056054318010163757
(9 rows)
```

Instrucciones PREPARE y EXECUTE

La base de datos ilimitada de Aurora PostgreSQL admite la optimización de una sola partición para las instrucciones `SELECT`, `UPDATE` y `DELETE` preparadas.

Sin embargo, si usa instrucciones preparadas para `PREPARE` y `EXECUTE` con `plan_cache_mode` establecido en `'force_generic_plan'`, el planificador de consultas rechaza la optimización de una sola partición para esa consulta.

PL/pgSQL

Las consultas con variables PL/pgSQL se ejecutan como instrucciones preparadas de forma implícita. Si una consulta contiene variables PL/pgSQL, el planificador de consultas rechaza la optimización de una sola partición.

El bloque PL/pgSQL admite la optimización si la instrucción no contiene ninguna variable PL/pgSQL.

Uniones totalmente cualificadas (explícitas)

La optimización de una sola partición se basa en la eliminación de particiones. El optimizador de PostgreSQL elimina las particiones en función de condiciones constantes. Si Base de datos ilimitada de Aurora PostgreSQL descubre que todas las particiones y tablas restantes se encuentran en la misma partición, marca la consulta apta para la optimización de una sola partición. Todas las condiciones del filtro deben ser explícitas para que la eliminación de particiones funcione. Base de datos ilimitada de Aurora PostgreSQL no puede eliminar las particiones sin uno o más predicados de unión o predicados de filtrado en las claves de partición de cada tabla particionada de la instrucción.

Supongamos que hemos dividido las tablas `customers`, `orders` y `order_details` en función de la columna `customer_id`. En este esquema, la aplicación intenta mantener todos los datos de un cliente en una única partición.

Analice la siguiente consulta:

```
SELECT * FROM
    customers c, orders o, order_details od
WHERE c.customer_id = o.customer_id
    AND od.order_id = o.order_id
    AND c.customer_id = 1;
```

Esta consulta recupera todos los datos de un cliente (`c.customer_id = 1`). Los datos de este cliente se encuentran en una sola partición, pero Base de datos ilimitada de Aurora PostgreSQL no cualifica esta consulta como consulta de una sola partición. El proceso del optimizador de la consulta es el siguiente:

1. El optimizador puede eliminar las particiones de `customers` y `orders` de la siguiente condición:

   ```
   c.customer_id = 1
   c.customer_id = o.customer_id
   o.customer_id =  1 (transitive implicit condition)
   ```

2. El optimizador no puede eliminar ninguna partición de `order_details` porque no hay ninguna condición constante en la tabla.
3. El optimizador concluye que ha leído todas las particiones de `order_details`. Por lo tanto, la consulta no puede cualificarse para la optimización de una sola partición.

Para convertirla en una consulta de una sola partición, agregamos la siguiente condición de unión explícita:

```
o.customer_id = od.customer_id
```

El cambio de consulta tiene un aspecto similar al siguiente:

```
SELECT * FROM
    customers c, orders o,  order_details od
WHERE c.customer_id = o.customer_id
     AND o.customer_id = od.customer_id
```

```
    AND od. order_id = o. order_id
AND c.customer_id =  1;
```

Ahora el optimizador puede eliminar las particiones de `order_details`. La nueva consulta se convierte en una consulta de una sola partición y cumple los requisitos para la optimización.

Definición de una clave de partición activa

Esta característica le permite establecer una única clave de partición al consultar la base de datos, lo que hace que todas las consultas `SELECT` y DML se asocien a la clave de partición como predicado constante. Esta característica resulta útil si ha migrado a Base de datos ilimitada de Aurora PostgreSQL y ha desnormalizado el esquema al añadir claves de partición a las tablas.

Puede añadir un predicado de clave de partición de forma automática a la lógica SQL existente, sin cambiar la semántica de las consultas. La adición de un predicado de clave de partición activa solo se realiza en el caso de las tablas compatibles.

La característica de clave de partición activa utiliza la variable `rds_aurora.limitless_active_shard_key`, que tiene la siguiente sintaxis:

```
SET [session | local] rds_aurora.limitless_active_shard_key = '{"col1_value",
 "col2_value", ...}';
```

Algunas consideraciones sobre las claves de partición activas y las claves externas:

- Una tabla particionada puede tener una restricción de clave externa si las tablas principal y secundaria están colocadas y la clave externa es un superconjunto de la clave de partición.
- Una tabla particionada puede tener una restricción de clave externa a una tabla de referencia.
- Una tabla particionada puede tener una restricción de clave externa a otra tabla de referencia.

Supongamos que tenemos una tabla `customers` particionada en la columna `customer_id`.

```
BEGIN;
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE customers(customer_id int PRIMARY KEY, name text , email text);
COMMIT;
```

Con un conjunto de claves de partición activo, las consultas sufren las siguientes transformaciones.

SELECT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
SELECT * FROM customers;

-- This statement is changed to:
SELECT * FROM customers WHERE customer_id = '123'::int;
```

INSERT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
INSERT INTO customers(name, email) VALUES('Alex', 'alex@example.com');

-- This statement is changed to:
INSERT INTO customers(customer_id, name, email) VALUES('123'::int, 'Alex',
 'alex@example.com');
```

UPDATE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
UPDATE customers SET email = 'alex_new_email@example.com';

-- This statement is changed to:
UPDATE customers SET email = 'alex_new_email@example.com' WHERE customer_id =
 '123'::int;
```

DELETE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
DELETE FROM customers;

-- This statement is changed to:
DELETE FROM customers WHERE customer_id = '123'::int;
```

Uniones

Al realizar operaciones de unión en tablas con una clave de partición activa, el predicado de la clave de partición se añade automáticamente a todas las tablas implicadas en la unión. Esta adición automática del predicado de la clave de partición solo se produce cuando todas las tablas de la consulta pertenecen al mismo grupo de colocación. Si la consulta incluye tablas de diferentes grupos de colocación, se genera un error.

Supongamos que también tenemos las tablas `orders` y `order_details` y que están colocadas junto a la tabla `customers`.

```
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_collocate_with='customers';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE orders (id int , customer_id int, total_amount int, date date);
CREATE TABLE order_details (id int , order_id int, customer_id int, product_name
 VARCHAR(100), price int);
COMMIT;
```

Recupere las últimas diez facturas de pedidos de un cliente cuyo ID de cliente sea 10.

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    order_details.order_id = orders.order_id AND
    customers.customer_id = 10
    order by order_date limit 10;
```

Esta consulta se convertirá en lo siguiente:

```
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    orders.order_id = order_details.order_id AND
    customers.customer_id = 10 AND
    order_details.customer_id = 10 AND
    orders.customer_id = 10 AND
    ORDER BY "order_date" LIMIT 10;
```

Tablas que admiten claves de partición activas

El predicado de la clave de partición se agrega solo a las tablas que son compatibles con la clave de partición activa. Una tabla se considera compatible si tiene el mismo número de columnas en su clave de partición que el especificado en la variable `rds_aurora.limitless_active_shard_key`. Si la consulta incluye tablas que son incompatibles con la clave de partición activa, el sistema genera un error en lugar de continuar con la consulta.

Por ejemplo:

```
-- Compatible table
SET rds_aurora.limitless_active_shard_key = '{"10"}';

-- The following query works because the customers table is sharded on one column.
SELECT * FROM customers;

-- Incompatible table
SET rds_aurora.limitless_active_shard_key = '{"10","20"}';

-- The following query raises a error because the customers table isn't sharded on
 two columns.
 SELECT * FROM customers;
```

Consultas distribuidas en Base de datos ilimitada de Aurora PostgreSQL

Las consultas distribuidas se ejecutan en un enrutador y en más de una partición. Uno de los enrutadores recibe la consulta. El enrutador crea y administra la transacción distribuida, que se envía a las particiones participantes. Las particiones crean una transacción local con el contexto proporcionado por el enrutador y se ejecuta la consulta.

Cuando se confirma la transacción, el enrutador utiliza un protocolo de confirmación bifásico optimizado, si es necesario, y un control de concurrencia multiversión (MVCC) basado en el tiempo para proporcionar la semántica ACID en un sistema de base de datos distribuido.

El MVCC basado en el tiempo registra el tiempo de confirmación de cada transacción y utiliza el tiempo de inicio de la transacción para generar la instantánea de los datos. Para identificar si una transacción está confirmada (visible) a partir de una instantánea del lector, la base de datos compara su tiempo de confirmación con el tiempo de la instantánea. Si su tiempo de confirmación es inferior al tiempo de la instantánea del lector, será visible; de lo contrario, no se verá. Con este protocolo, siempre verá datos muy consistentes en Base de datos ilimitada de Aurora PostgreSQL.

Seguimiento de consultas distribuidas en Base de datos ilimitada de Aurora PostgreSQL

El seguimiento de consultas distribuidas es una herramienta para hacer el seguimiento y correlacionar las consultas en los registros de PostgreSQL en Base de datos ilimitada de Aurora PostgreSQL. En Aurora PostgreSQL, se utiliza el ID de transacción para identificar una transacción. Sin embargo, en Base de datos ilimitada de Aurora PostgreSQL, el ID de transacción puede repetirse en varios enrutadores. Por lo tanto, recomendamos que utilice el ID de seguimiento en Base de datos ilimitada.

A continuación se muestran casos de uso claves:

- Utilice la función `rds_aurora.limitless_get_last_trace_id()` para buscar el ID de seguimiento único de la última consulta ejecutada en la sesión actual. A continuación, busque el grupo de registros del clúster de base de datos en Registros de Amazon CloudWatch con ese ID de seguimiento para encontrar todos los registros relacionados.

  Puede usar los parámetros `log_min_messages` y `log_min_error_statement` juntos para controlar el volumen de registros impresos e imprimir una instrucción que contenga el ID de seguimiento.
- Utilice el parámetro `log_min_duration_statement` para determinar el tiempo de ejecución a partir del cual todas las consultas imprimen su duración de ejecución y su ID de seguimiento. Luego, este tiempo de ejecución se puede buscar en el grupo de registros del clúster de base de datos de Registros de CloudWatch para ayudar a determinar los nodos con cuello de botella y los esfuerzos de optimización del planificador.

  El parámetro `log_min_duration_statement` habilita el ID de seguimiento de todos los nodos, independientemente de los valores de `log_min_messages` y los parámetros `log_min_error_statement`.

Temas

- ID de seguimiento
- Uso del seguimiento de consultas
- Ejemplos de registro

ID de seguimiento

El elemento central de esta característica es un identificador único conocido como ID de seguimiento. El ID de seguimiento es una cadena de 31 dígitos que se asocia a las líneas de registro STATEMENT de los registros de PostgreSQL y que actúa como un identificador preciso para correlacionar los registros relacionados con una ejecución de una consulta específica. Algunos ejemplos son `11262533757194085040000000000011` y `11262533757194084950000000000090`.

El ID de seguimiento se compone de lo siguiente:

- Identificador de la transacción: los primeros 20 dígitos, que identifican de forma exclusiva la transacción.
- Identificador del comando: los primeros 30 dígitos, que identifican una consulta individual dentro de una transacción.

  Si se ejecutan más de 4 294 967 294 consultas dentro de un bloque de transacciones explícito, el identificador del comando se reduce a 1. Si ocurre esto, recibirá una notificación mediante el siguiente mensaje `LOG` en el registro de PostgreSQL:

  ```
  wrapping around the tracing ID back to 1 after running 4294967294 (4.2 billion or
   2^32-2) queries inside of an explicit transaction block
  ```
- Identificador del tipo de nodo: es el último dígito que indica si el nodo funciona como enrutador coordinador (1) o nodo participante (`0`).

En los siguientes ejemplos se ilustran los componentes del ID de seguimiento:

- `11262533757194085040000000000011`:
  - Identificador de la transacción: `1126253375719408504`
  - Identificador del comando: `11262533757194085040000000000011` indica el primer comando del bloque de transacciones
  - Identificador de tipo de nodo: 1 indica un enrutador coordinador
- `11262533757194084950000000000090`:
  - Identificador de la transacción: `1126253375719408495`
  - Identificador del comando: `11262533757194084950000000000090` indica el noveno comando del bloque de transacciones
  - Identificador de tipo de nodo: `0` indica un nodo participante

Uso del seguimiento de consultas

Realice las siguientes tareas para utilizar el seguimiento de consultas:

1. Asegúrese de que el seguimiento esté habilitado.

   Puede utilizar este comando para realizar la comprobación:

   ```
   SHOW rds_aurora.limitless_log_distributed_trace_id;
   ```

   Está habilitado de forma predeterminada (on). Si no está habilitado, configúrelo con el comando siguiente:

   ```
   SET rds_aurora.limitless_log_distributed_trace_id = on;
   ```

2. Controle el volumen de registros a imprimir configurando el nivel de gravedad del registro.

   El volumen de registros lo controla el parámetro `log_min_messages`. El parámetro `log_min_error_statement` se utiliza para imprimir la línea `STATEMENT` con el ID de seguimiento. Ambos se establecen en `ERROR` de forma predeterminada. Puede utilizar estos comandos para realizar la comprobación:

   ```
   SHOW log_min_messages;
   SHOW log_min_error_statement;
   ```

   Para actualizar el nivel de gravedad e imprimir la línea `STATEMENT` de la sesión actual, utilice los siguientes comandos con uno de estos niveles de gravedad:

   ```
   SET log_min_messages = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO | NOTICE
    | WARNING | ERROR | LOG | FATAL | PANIC';
   SET log_min_error_statement = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO
    | NOTICE | WARNING | ERROR | LOG | FATAL | PANIC';
   ```

   Por ejemplo:

   ```
   SET log_min_messages = 'WARNING';
   SET log_min_error_statement = 'WARNING';
   ```

3. Habilite la impresión del ID de seguimiento en los registros por encima de un tiempo de ejecución específico.

El parámetro `log_min_duration_statement` se puede cambiar al tiempo mínimo de ejecución de la consulta por encima del cual la consulta imprime una línea de registro con la duración de la ejecución, junto con los ID de seguimiento en todo el clúster de base de datos. Este parámetro está configurado en `-1` de forma predeterminada, lo que significa que está deshabilitado. Puede utilizar este comando para realizar la comprobación:

```
SHOW log_min_duration_statement;
```

Al cambiarlo a `0`, se imprimen la duración y el ID de seguimiento en los registros de cada consulta del clúster de base de datos. Puede configurarlo en `0` para la sesión actual mediante el siguiente comando:

```
SET log_min_duration_statement = 0;
```

4. Obtenga el ID de seguimiento.

   Tras ejecutar una consulta (incluso dentro de un bloque de transacciones explícito), llame a la función `rds_aurora.limitless_get_last_trace_id` para obtener el ID de seguimiento de la última consulta ejecutada:

   ```
   SELECT * FROM rds_aurora.limitless_get_last_trace_id();
   ```

   Esta función devuelve el identificador de la transacción y el identificador del comando. No devuelve el identificador del tipo de nodo.

   ```
   => SELECT * FROM customers;
    customer_id | fname | lname
   -------------+-------+-------
   (0 rows)

   => SELECT * FROM rds_aurora.limitless_get_last_trace_id();
    transaction_identifier |       command_identifier
   ------------------------+--------------------------------
    10104661421959001813   | 10104661421959001813000000000001
   (1 row)
   ```

   La función devuelve una línea en blanco para las consultas no distribuidas, ya que no tienen un ID de seguimiento.

```
=> SET search_path = public;
SET

=> SELECT * FROM rds_aurora.limitless_get_last_trace_id();
 transaction_identifier | command_identifier
------------------------+--------------------
                        |
(1 row)
```

> **Note**
>
> En el caso de las consultas VACUUM y ANALYZE, la instrucción de duración no se registra con el ID de seguimiento. Por lo tanto, `limitless_get_last_trace_id()` no devuelve el ID de seguimiento. Si VACUUM o ANALYZE son una operación de larga duración, puede usar la siguiente consulta para obtener el ID de seguimiento de esa operación:
>
> ```
> SELECT * FROM rds_aurora.limitless_stat_activity
> WHERE distributed_tracing_id IS NOT NULL;
> ```
>
> Si el servidor se detiene antes de que pueda encontrar el último ID de seguimiento, tendrá que buscar manualmente en los registros de PostgreSQL para encontrar los ID de seguimiento en los registros justo antes de que se produjera el error.

5. Busque el ID de seguimiento en los registros del clúster de base de datos mediante CloudWatch.

   Utilice Información de CloudWatch para consultar el grupo de registro del clúster de base de datos, tal como se muestra en los siguientes ejemplos.

   - Busque un identificador de transacción concreto y todos los comandos que se ejecuten dentro de él:

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813/
     | sort @timestamp desc
     ```

   - Consulte un identificador de comando concreto:

     ```
     fields @timestamp, @message
     ```

```
| filter @message like /1010466142195900181300000000001/
| sort @timestamp desc
```

6. Examine todos los registros del clúster de base de datos generados por la consulta distribuida.

Ejemplos de registro

Los siguientes ejemplos muestran el uso del seguimiento de consultas.

Registros correlacionados para consultas propensas a errores

En este ejemplo, el comando `TRUNCATE` se ejecuta en la tabla `customers` cuando esa tabla no existe.

Sin seguimiento de consultas

Archivo de registro de PostgreSQL en el enrutador coordinador:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:CONTEXT: remote
 SQL command: truncate public.customers;
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:STATEMENT:
 truncate customers;
```

Archivo de registro de PostgreSQL en una partición participante:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:STATEMENT:
 truncate customers;
```

Estos registros son típicos. Carecen de los identificadores precisos necesarios para correlacionar fácilmente las consultas en el clúster de base de datos.

Con seguimiento de consultas

Archivo de registro de PostgreSQL en el enrutador coordinador:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:CONTEXT: remote SQL
 command: truncate public.customers;
```

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT: /* tid =
 11262533757194085027000000000011 */ truncate customers;
```

Archivo de registro de PostgreSQL en una partición participante:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR:  failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT:  /* tid
 = 11262533757194085027000000000010 */ truncate customers;
```

Si hay un seguimiento de consultas, a cada línea de registro se le añade un identificador único de 31 dígitos. Aquí, `1126253375719408502700000000011` y `1126253375719408502700000000010` representan los ID de seguimiento de los nodos coordinador y participante, respectivamente.

- Identificador de la transacción: `11262533757194085027`
- Identificador del comando: `112625337571940850270000000001`
- Identificador del tipo de nodo: el último dígito, `1` o `0`, indica un enrutador coordinador y un nodo participante, respectivamente.

Correlación de los registros para encontrar el tiempo de ejecución de la consulta en varios nodos

En este ejemplo, el parámetro `log_min_duration_statement` se ha actualizado a `0` para imprimir la duración de todas las consultas.

Sin seguimiento de consultas

```
2024-01-15 07:28:46 UTC:[local]:postgres@postgres_limitless:[178322]:LOG: duration:
 12.779 ms statement: select * from customers;
```

Con seguimiento de consultas

Archivo de registro de PostgreSQL en el enrutador coordinador:

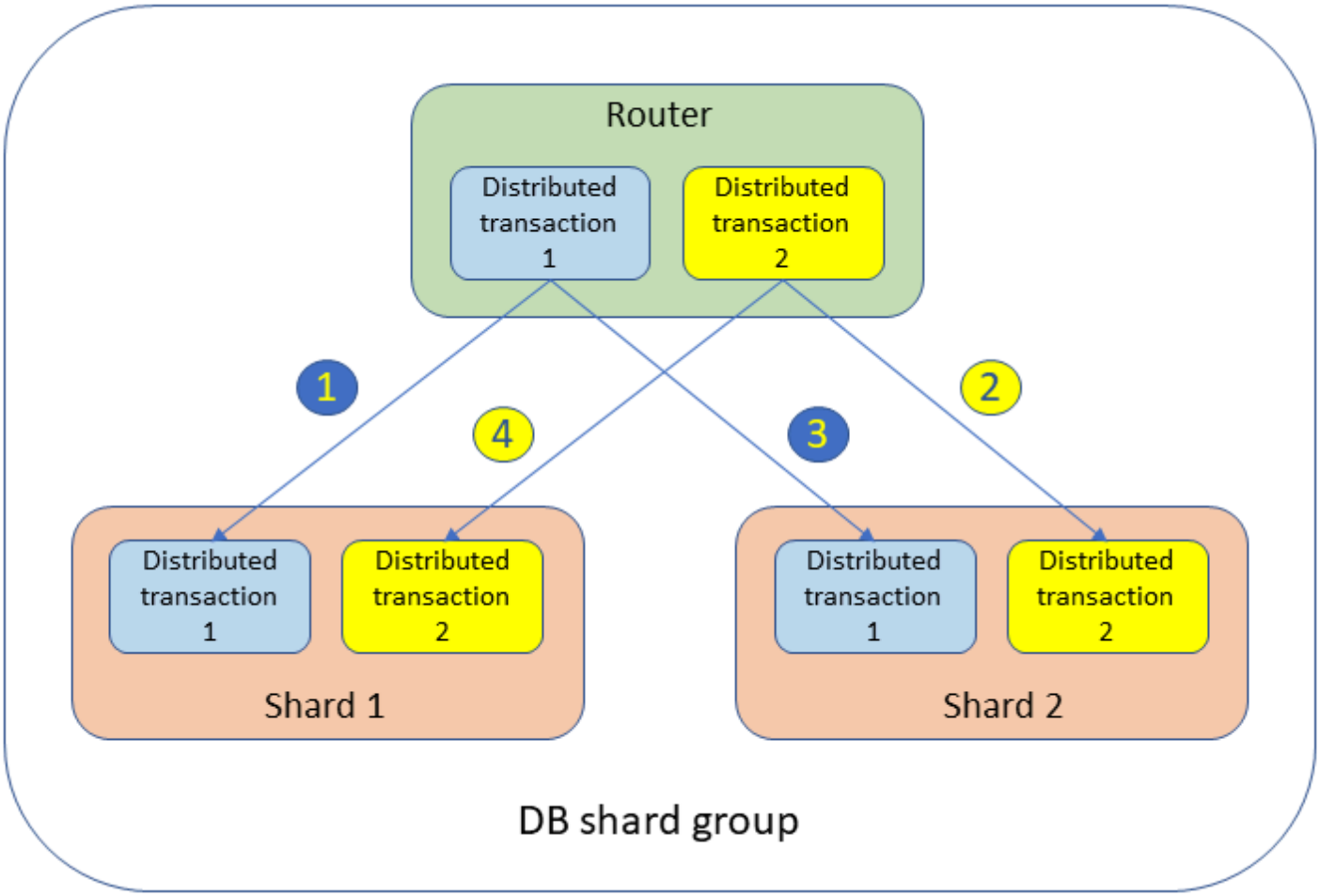
```
2024-01-15 07:32:08 UTC:[local]:postgres@postgres_limitless:[183877]:LOG: duration:
 12.618 ms statement: /* tid = 045766956624049708840000000011 */ select * from
 customers;
```

Archivo de registro de PostgreSQL en una partición participante:

```
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.279 ms statement: /* tid = 045766956624049708840000000010 */ START
 TRANSACTION ISOLATION LEVEL READ COMMITTED
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.249 ms parse <unnamed>: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.398 ms bind <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.019 ms execute <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.073 ms statement: /* tid = 045766956624049708840000000010 */ COMMIT
 TRANSACTION
```

Interbloqueos distribuidos en Base de datos ilimitada de Aurora PostgreSQL

En un grupo de particiones de base de datos, pueden producirse interbloqueos entre transacciones distribuidas entre distintos enrutadores y particiones. Por ejemplo, se ejecutan dos transacciones distribuidas simultáneas que abarcan dos particiones, tal y como se muestra en la siguiente figura.



Las transacciones bloquean las tablas y crean eventos de espera en las dos particiones de la siguiente manera:

1. Transacción distribuida 1:

```
UPDATE table SET value = 1 WHERE key = 'shard1_key';
```

   Esta acción coloca un bloqueo en la partición 1.

2. Transacción distribuida 2:

```
UPDATE table SET value = 2 WHERE key = 'shard2_key';
```

Esta acción coloca un bloqueo en la partición 2.

3. Transacción distribuida 1:

```
UPDATE table SET value = 3 WHERE key = 'shard2_key';
```

La transacción distribuida 1 está esperando la partición 2.

4. Transacción distribuida 2:

```
UPDATE table SET value = 4 WHERE key = 'shard1_key';
```

La transacción distribuida 2 está esperando la partición 1.

En este escenario, ni la partición 1 ni la partición 2 ven el problema: la transacción 1 espera la transacción 2 de la partición 2 y la transacción 2 espera la transacción 1 de la partición 1. Desde una perspectiva global, la transacción 1 espera la transacción 2 y la transacción 2 espera la transacción 1. Esta situación en la que dos transacciones de dos particiones diferentes se esperan una a la otra se denomina interbloqueo distribuido.

Base de datos ilimitada de Aurora PostgreSQL puede detectar y resolver los interbloqueos distribuidos automáticamente. Cuando una transacción tarda demasiado tiempo en adquirir un recurso, se notifica a un enrutador del grupo de particiones de base de datos. El enrutador que recibe la notificación comienza a recopilar la información necesaria de todos los enrutadores y particiones del grupo de particiones de base de datos. A continuación, el enrutador finaliza las transacciones que participan en un interbloqueo distribuido, hasta que el resto de las transacciones del grupo de particiones de base de datos pueda continuar sin que se bloqueen entre sí.

Recibe el siguiente error si la transacción formaba parte de un interbloqueo distribuido y, a continuación, el enrutador la ha finalizado:

```
ERROR: aborting transaction participating in a distributed deadlock
```

El parámetro del clúster de base de datos `rds_aurora.limitless_distributed_deadlock_timeout` establece el tiempo que debe esperar cada transacción en un recurso antes de notificar al enrutador que compruebe si hay un

interbloqueo distribuido. Puede aumentar el valor del parámetro si su carga de trabajo es menos propensa a sufrir situaciones de interbloqueo. El valor predeterminado es de `1000` milisegundos (un segundo).

El ciclo de interbloqueo distribuido se publica en los registros de PostgreSQL cuando se encuentra y se resuelve un interbloqueo entre nodos. La información sobre cada proceso que forma parte del interbloqueo incluye lo siguiente:

- Nodo coordinador que ha iniciado la transacción
- ID de transacción virtual (xid) de la transacción en el nodo coordinador con el formato *`backend_id`*/*`backend_local_xid`*
- ID de sesión distribuido de la transacción

Administración de Base de datos ilimitada de Aurora PostgreSQL

En los temas siguientes se describe cómo administrar los clústeres de base de datos de Base de datos ilimitada de Aurora PostgreSQL.

Temas

- Base de datos y consideraciones de tamaño de tabla
- Recuperación de espacio de almacenamiento mediante el vaciado

Base de datos y consideraciones de tamaño de tabla

En el caso de Base de datos ilimitada de Aurora PostgreSQL, en cada partición, una tabla particionada se divide en varios segmentos de tabla que varían según el número de particiones disponibles en el grupo de particiones de base de datos. Cada segmento de la tabla puede crecer hasta 32 TiB, pero cada partición tiene una capacidad máxima de 128 TiB. Las tablas de referencia tienen un límite de tamaño de 32 TiB para todo el grupo de particiones de base de datos.

> **Note**
>
> La capacidad máxima de cada nodo (enrutador o partición) es de 128 TiB, ya que es la capacidad máxima de un clúster de base de datos de Aurora PostgreSQL.

El número máximo de relaciones por base de datos (incluidas tablas, vistas e índices) tanto en Aurora PostgreSQL como en Base de datos ilimitada de Aurora PostgreSQL es de 1 431 650 303.

Para obtener más información, consulte Appendix K. PostgreSQL limits en la documentación de PostgreSQL y Límites de tamaño de Amazon Aurora.

Recuperación de espacio de almacenamiento mediante el vaciado

El control de simultaneidad multiversión (MVCC) de PostgreSQL ayuda a preservar la integridad de los datos al guardar una copia interna de las filas actualizadas o eliminadas hasta que se confirme o anule una transacción. Estas copias, también denominadas tuplas, pueden sobrecargar las tablas si no se limpian con regularidad. Las instancias de PostgreSQL ordenan las transacciones por sus ID de transacción. PostgreSQL utiliza el MVCC basado en el ID de transacción para controlar la visibilidad de las tuplas y aislar las transacciones. Cada transacción establece una instantánea de los datos y cada tupla tiene una versión. Tanto la instantánea como la versión se basan en el ID de la transacción.

Para limpiar los datos, la utilidad `VACUUM` realiza cuatro funciones clave en PostgreSQL:

- `VACUUM`: elimina las versiones de filas caducadas, lo que permite reutilizar el espacio que ocupaban.
- `VACUUM FULL`: proporciona una desfragmentación completa al eliminar las versiones con filas muertas y compactar las tablas, lo que reduce el tamaño y aumenta la eficiencia.
- `VACUUM FREEZE`: protege contra los problemas relacionados con los ID de las transacciones al marcar las versiones antiguas de las filas como inmovilizadas.
- `VACUUM ANALYZE`: elimina las versiones de las filas inactivas y actualiza las estadísticas de planificación de consultas de la base de datos. Es una combinación de las funciones `VACUUM` y `ANALYZE`. Para obtener más información sobre cómo funciona `ANALYZE` en Base de datos ilimitada de Aurora PostgreSQL, consulte ANALYZE.

De igual modo que con el MVCC, el vaciado en Aurora PostgreSQL se basa en el ID de la transacción. Si hay una transacción en curso cuando se inicia el vaciado, no se eliminan las filas que aún estén visibles de esa transacción.

Para obtener más información acerca de la utilidad `VACUUM`, consulte VACUUM en la documentación de PostgreSQL. Para obtener más información sobre la compatibilidad con `VACUUM` en Base de datos ilimitada de Aurora PostgreSQL, consulte VACUUM.

Temas

- AUTOVACUUM
- Vaciado basado en el tiempo en Base de datos ilimitada de Aurora PostgreSQL
- Uso de las estadísticas de base de datos para vaciar

- Diferencias en el comportamiento de vaciado entre Aurora PostgreSQL y Base de datos ilimitada de Aurora PostgreSQL

AUTOVACUUM

Aurora PostgreSQL utiliza las `VACUUM` y `AUTOVACUUM` para eliminar las tuplas innecesarias. El mecanismo subyacente para `AUTOVACUUM` y el `VACUUM` manual son los mismos; la única diferencia es la automatización.

`AUTOVACUUM` en Aurora PostgreSQL y en Base de datos ilimitada de Aurora PostgreSQL es una combinación de las utilidades `VACUUM` y `ANALYZE`. `AUTOVACUUM` determina qué bases de datos y tablas se van a limpiar, de acuerdo con una regla predefinida, como el porcentaje de tuplas inactivas y el número de inserciones.

Por ejemplo, `AUTOVACUUM` se despierta periódicamente para realizar la limpieza. El intervalo de tiempo lo controla el parámetro `autovacuum_naptime`. El valor predeterminado es 1 minuto. Los valores predeterminados para los parámetros de configuración `AUTOVACUUM` y `VACUUM` son los mismos para Base de datos ilimitada de Aurora PostgreSQL y para Aurora PostgreSQL.

El daemon `AUTOVACUUM`, si está activado, emite comandos `ANALYZE` automáticamente siempre que el contenido de una tabla haya cambiado lo suficiente. En Base de datos ilimitada de Aurora PostgreSQL, `AUTOVACUUM` emite `ANALYZE` tanto en los enrutadores como en las particiones.

Para obtener más información sobre los parámetros de almacenamiento de tablas y el daemon `AUTOVACUUM` asociados a `AUTOVACUUM`, consulte The autovacuum daemon y Storage parameters en la documentación de PostgreSQL.

Vaciado basado en el tiempo en Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada de Aurora PostgreSQL es un sistema distribuido, lo que significa que pueden participar varias instancias en una transacción. Por lo tanto, no se aplica la visibilidad basada en el ID de la transacción. En cambio, Base de datos ilimitada de Aurora PostgreSQL emplea la utilidad basada en el tiempo, pues los ID de transacción no están unificados en todas las instancias, sino que el tiempo se puede unificar en todas las instancias. Cada instantánea de transacción y cada versión de la tupla obedecen al tiempo en lugar de al ID de la transacción. Para ser más específicos, una instantánea de transacción tiene una hora de inicio de la instantánea y una tupla tiene una hora de creación (cuando ocurre `INSERT` o `UPDATE`) y una hora de eliminación (cuando ocurre `DELETE`).

Para mantener la coherencia de datos en todas las instancias del grupo de particiones de base de datos, Base de datos ilimitada de Aurora PostgreSQL debe asegurarse de que al vaciar no se elimine ninguna tupla que aún esté visible para alguna transacción activa del grupo de particiones de base de datos. Por ello, el vaciado en Base de datos ilimitada de Aurora PostgreSQL también se basa en el tiempo. Otros aspectos de `VACUUM` siguen siendo los mismos, como el hecho de que, para ejecutar `VACUUM` en una tabla en particular, el usuario debe tener acceso a esa tabla.

> **Note**
>
> Recomendamos encarecidamente que no deje las transacciones abiertas durante períodos de tiempo prolongados.
> El vaciado basado en el tiempo consume más memoria que el vaciado basado en el ID de transacción.

En el siguiente ejemplo, se ilustra cómo funciona el vaciado basado en el tiempo.

1. Una tabla de clientes se distribuye en cuatro particiones.
2. La transacción 1 comienza con una lectura repetible y va dirigida a solo una partición (partición 1). Esta transacción permanece abierta.

   La transacción 1 es más antigua que cualquier otra transacción iniciada después de ella.
3. La transacción 2 comienza más tarde, elimina todas las tuplas de una tabla y, a continuación, se confirma.
4. Si `AUTOVACUUM` o `VACUUM` manual intenta limpiar las tuplas inactivas (inactivas debido a la transacción 2), no se elimina nada.

   Esto es cierto no solo para la partición 1, sino también para las particiones 2 a 4, ya que es posible que la transacción 1 aún necesite acceder a estas tuplas. Gracias al MVCC, siguen siendo visibles para la transacción 1.

El último paso se realiza mediante la sincronización, de modo que todas las particiones estén al tanto de la transacción 1, aunque la transacción 1 no afecte a todas.

Uso de las estadísticas de base de datos para vaciar

Para obtener información sobre las tuplas que pueda necesitar para la limpieza, use la vista limitless_stat_all_tables, cuyo funcionamiento es similar al de la vista pg_stat_all_tables. El siguiente ejemplo consulta la vista.

```
SELECT * FROM rds_aurora.limitless_stat_all_tables WHERE relname LIKE '%customer%';
```

Del mismo modo, use limitless_stat_database en lugar de pg_stat_database, y limitless_stat_activity en lugar de pg_stat_activity para las estadísticas de base de datos.

Para comprobar el uso del disco, utilice la función limitless_stat_relation_sizes, que funciona de forma similar a pg_relation_size. El siguiente ejemplo consulta la función.

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customer');
```

Para hacer el seguimiento del progreso de una operación VACUUM en Base de datos ilimitada de Aurora PostgreSQL, use la vista limitless_stat_progress_vacuum en lugar de pg_stat_progress_vacuum. El siguiente ejemplo consulta la vista.

```
SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;
```

Para obtener más información, consulte Vistas de Base de datos ilimitada de Aurora PostgreSQL y Funciones de Base de datos ilimitada de Aurora PostgreSQL.

Diferencias en el comportamiento de vaciado entre Aurora PostgreSQL y Base de datos ilimitada de Aurora PostgreSQL

A continuación se enumeran otras diferencias entre Aurora PostgreSQL y Base de datos ilimitada de Aurora PostgreSQL en cuanto al funcionamiento del vaciado:

- Aurora PostgreSQL realiza operaciones de VACUUM con los ID de transacción hasta la transacción en curso más antigua. Si no hay ninguna transacción en curso en la base de datos, VACUUM realiza la operación hasta la última transacción.
- Base de datos ilimitada de Aurora PostgreSQL sincroniza la instantánea de tiempo más antigua cada diez segundos. Por lo tanto, es posible que VACUUM no realice la operación en ninguna transacción que se haya ejecutado en los últimos diez segundos.

Para obtener más información sobre la compatibilidad de `VACUUM` en Base de datos ilimitada de Aurora PostgreSQL, consulte VACUUM en la Referencia sobre Base de datos ilimitada de Aurora PostgreSQL.

Supervisión de Base de datos ilimitada de Aurora PostgreSQL

Puede usar Amazon CloudWatch, Monitorización mejorada e Información de rendimiento para supervisar Base de datos ilimitada de Aurora PostgreSQL. También hay nuevas funciones y vistas de estadísticas y eventos de espera para Base de datos ilimitada de Aurora PostgreSQL que puede utilizar para la supervisión y el diagnóstico.

Temas

- Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Amazon CloudWatch
- Supervisión de la base de datos ilimitada de Aurora PostgreSQL con información de base de datos de CloudWatch
- Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Registros de Amazon CloudWatch
- Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Monitorización mejorada
- Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Información de rendimiento
- Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Amazon GuardDuty para protección de RDS
- Funciones y vistas de Base de datos ilimitada de Aurora PostgreSQL
- Eventos de espera de Base de datos ilimitada de Aurora PostgreSQL

Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Amazon CloudWatch

Las métricas de CloudWatch para Base de datos ilimitada de Aurora PostgreSQL se presentan en las siguientes dimensiones:

- DBShardGroup
- DBShardGroupRouterAggregation
- DBShardGroupInstance
- DBClusterIdentifier

Para obtener más información sobre las métricas de CloudWatch, consulte Supervisión de métricas de Amazon Aurora con Amazon CloudWatch.

Métricas de DBShardGroup

Para ver las métricas de `DBShardGroup` para Base de datos ilimitada de Aurora PostgreSQL en la consola de CloudWatch, elija RDS y, a continuación, DBShardGroup.

Puede realizar un seguimiento de las siguientes métricas de CloudWatch:

- `DBShardGroupACUUtilization`: uso de la unidad de capacidad de Aurora (ACU) como porcentaje calculado a partir de `DBShardGroupCapacity` divido entre `DBShardGroupMaxACU`.
- `DBShardGroupCapacity`: número de ACU consumidas por las instancias del escritor del grupo de particiones de base de datos.
- `DBShardGroupComputeRedundancyCapacity`: número de ACU consumidas por las instancias de espera del grupo de particiones de base de datos.
- `DBShardGroupMaxACU`: número máximo de ACU configurado por el grupo de particiones de base de datos.
- `DBShardGroupMinACU`: número mínimo de ACU requeridas por el grupo de particiones de base de datos.

La clave de dimensión `DBShardGroupIdentifier` está disponible para agregar las métricas de `DBShardGroup`.

Métricas de DBShardGroupRouterAggregation

Para ver las métricas de `DBShardGroupRouterAggregation` para Base de datos ilimitada de Aurora PostgreSQL en la consola de CloudWatch, elija RDS y, a continuación, DBShardGroupRouterAggregation.

Puede realizar un seguimiento de las siguientes métricas de CloudWatch:

- `CommitThroughput`: número medio de operaciones de confirmación por segundo en todos los nodos del enrutador del grupo de particiones de bases de datos.
- `DatabaseConnections`: suma de todas las conexiones en todos los nodos del enrutador del grupo de particiones de base de datos.

Métricas de DBShardGroupInstance

Una DBShardGroupInstance es la instancia de base de datos individual dentro de cada partición o subclúster de enrutadores.

Para ver las métricas de `DBShardGroupInstance` para Base de datos ilimitada de Aurora PostgreSQL en la consola de CloudWatch, elija RDS y, a continuación, DBShardGroupInstance.

Puede realizar un seguimiento de las siguientes métricas de CloudWatch:

- `ACUUtilization`: el porcentaje calculado como la métrica `ServerlessDatabaseCapacity` dividida por el valor máximo de ACU asignado del subclúster.
- `AuroraReplicaLag`: para los clústeres ilimitados con redundancia de procesamiento habilitada, esta es la cantidad de retraso al replicar actualizaciones desde la instancia principal en el subclúster.
- `AuroraReplicaLagMaximum`: para los clústeres ilimitados con redundancia de procesamiento habilitada, esta es la cantidad máxima de retraso al replicar actualizaciones desde la instancia principal en el subclúster. Cuando las réplicas de lectura se eliminan o se les cambia el nombre, es posible que se produzca un aumento temporal en el retraso de la replicación a medida que el recurso antiguo se recicla. Utilice esta métrica para averiguar si se ha producido una conmutación por error debido a un gran retraso en la replicación en uno de sus lectores.
- `AuroraReplicaLagMinimum`: para los clústeres ilimitados con redundancia de procesamiento habilitada, esta es la cantidad mínima de retraso al replicar actualizaciones desde la instancia principal en el subclúster.

- `BufferCacheHitRatio`: es el porcentaje de datos e índices servidos desde la caché de memoria de una instancia (en contraposición al volumen de almacenamiento).
- `CommitLatency`: es el tiempo medio que tarda el motor y el almacenamiento en completar las operaciones de confirmación de un nodo en particular (enrutador o partición).
- `CommitThroughput`: número medio de operaciones de confirmación por segundo.
- `CPUUtilization`: uso de la CPU como porcentaje del valor máximo de ACU asignado del subclúster.
- `FreeableMemory`: la cantidad de memoria no usada que está disponible cuando el grupo de particiones se escala a su capacidad máxima. Esto viene determinado por las ACU asignadas al grupo de particiones. Por cada ACU cuya capacidad actual esté por debajo de la capacidad máxima, este valor aumenta aproximadamente 2 GiB. Por lo tanto, esta métrica no se aproxima a cero hasta que el grupo de particiones de base de datos se escale hasta el límite máximo.
- `MaximumUsedTransactionIDs`: es la antigüedad del ID de transacción sin vaciar más antiguo, en transacciones. Si este valor alcanza 2 146 483 648 (2^31 - 1 000 000), la base de datos se fuerza en modo de solo lectura, a fin de evitar el reinicio de los ID de transacción. Para obtener más información, consulte Prevención de fallos en la identificación de las transacciones en la documentación de PostgreSQL.
- `NetworkReceiveThroughput`: es la cantidad de rendimiento de red que recibe de los clientes cada instancia en el grupo de particiones de base de datos. Este rendimiento no incluye el tráfico de red entre las instancias del grupo de particiones de bases de datos y el volumen de clúster.
- `NetworkThroughput`: es el rendimiento agregado de la red (tanto transmitido como recibido) entre clientes y enrutadores, y entre enrutadores y particiones del grupo de particiones de base de datos. Este rendimiento no incluye el tráfico de red entre las instancias del grupo de particiones de bases de datos y el volumen de clúster.
- `NetworkTransmitThroughput`: es la cantidad de rendimiento de red que envía a los clientes con cada instancia en el grupo de particiones de base de datos. Este rendimiento no incluye el tráfico de red entre las instancias del grupo de particiones de bases de datos y el volumen de clúster.
- `ReadIOPS`: es el número medio de operaciones de entrada/salida por segundo (IOPS) de lectura de disco.
- `ReadLatency`: es el tiempo medio de cada operación de entrada/salida (E/S) de lectura de disco.
- `ReadThroughput`: es el número medio de bytes leídos del disco por segundo.
- `ServerlessDatabaseCapacity`: la capacidad actual de la partición de base de datos o subclúster de enrutador dentro del grupo de particiones de base de datos.

- `StorageNetworkReceiveThroughput`: es la cantidad de rendimiento de red que recibe del subsistema de almacenamiento de Aurora cada instancia del grupo de particiones de base de datos.
- `StorageNetworkThroughput`: es el rendimiento de red agregado que emite y recibe del subsistema de almacenamiento de Aurora cada instancia del grupo de particiones de base de datos.
- `StorageNetworkTransmitThroughput`: es la cantidad de rendimiento de red que envía al subsistema de almacenamiento de Aurora cada instancia en el clúster de base de datos de Aurora.
- `SwapUsage`: la cantidad de espacio de intercambio utilizado por el grupo de particiones de la base de datos.
- `TempStorageIOPS`: es el número medio de operaciones de E/S realizadas en el almacenamiento local asociadas a la instancia de base de datos. Incluye operaciones de E/S de lectura y escritura.

  `TempStorageIOPS` se puede usar con `TempStorageThroughput` para diagnosticar los raros casos en los que la actividad de red para transferencias entre sus instancias de base de datos y los dispositivos de almacenamiento locales es responsable de aumentos de capacidad no esperados.
- `TempStorageThroughput`: es la cantidad de datos transferidos desde y hacia el almacenamiento local asociado al enrutador o la partición.
- `WriteIOPS`: es el número medio de IOPS de escritura en disco.
- `WriteLatency`: es el tiempo medio de cada operación de E/S de escritura en disco.
- `WriteThroughput`: es el número medio de bytes que se escribe en el disco por segundo.

Las siguientes claves de dimensión están disponibles para agregar las métricas de `DBShardGroupInstance`:

- `DBClusterIdentifier`: es el nombre del clúster de Base de datos ilimitada de Aurora PostgreSQL.
- `DBShardGroupIdentifier`: es el grupo de particiones de base de datos al que pertenece la instancia.
- `DBShardGroupSubClusterType`: es el tipo de nodo, ya sea `Distributed Transaction Router` (enrutador) o `Data Access Shard` (partición).
- `DBShardGroupSubClusterIdentifier`: es el nombre del enrutador o la partición al que pertenece la instancia.

A continuación, se muestran ejemplos de la agregación de métricas de CloudWatch:

- `CPUUtilization` total de todas las instancias que pertenecen a una partición o enrutador concretos de un grupo de particiones de base de datos.
- `CPUUtilization` total de todas las instancias de un grupo de particiones de base de datos.

Métricas de DBClusterIdentifier

Para ver las métricas de `DBClusterIdentifier` para Base de datos ilimitada de Aurora PostgreSQL en la consola de CloudWatch, elija RDS y, a continuación, DBClusterIdentifier.

Al utilizar Base de datos ilimitada de Aurora PostgreSQL, es posible que tenga más operaciones de entrada/salida (E/S) que las que tendría en un clúster de base de datos de Aurora. Puede realizar un seguimiento de las siguientes métricas de CloudWatch para su clúster de Base de datos ilimitada:

- `VolumeReadIops`: es el número de operaciones de E/S de lectura facturadas de un volumen de clúster a intervalos de 5 minutos.
- `VolumeWriteIops`: es el número de operaciones de E/S de escritura en el volumen de clúster a intervalos de 5 minutos.

Base de datos ilimitada de Aurora PostgreSQL utiliza la configuración de almacenamiento en clúster Aurora I/O-Optimized. Con Aurora I/O-Optimized, paga un único precio mensual por todas las operaciones de E/S, en lugar de pagar por cada millón de solicitudes de E/S. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de base de datos de Amazon Aurora.

También es posible que utilice más almacenamiento del que utilizaría para un clúster de base de datos de Aurora. Puede realizar un seguimiento de las siguientes métricas de CloudWatch para el almacenamiento:

- `BackupRetentionPeriodStorageUsed`: es el uso total del almacenamiento de copia de seguridad continuo facturado de su clúster de Base de datos ilimitada de Aurora PostgreSQL.
- `SnapshotStorageUsed`: es el uso total del almacenamiento de instantáneas facturado de su clúster de Base de datos ilimitada de Aurora PostgreSQL.
- `TotalBackupStorageBilled`: es la suma de los costos de la retención automática de las copias de seguridad y las instantáneas del clúster de base de datos.

Para obtener más información acerca de los costos de almacenamiento de las copias de seguridad, consulte Descripción del uso de almacenamiento de copias de seguridad en Amazon Aurora.

- `VolumeBytesUsed`: es la cantidad de almacenamiento que utiliza su clúster de Base de datos ilimitada de Aurora PostgreSQL a intervalos de 5 minutos.

Supervisión de la base de datos ilimitada de Aurora PostgreSQL con información de base de datos de CloudWatch

Se requiere el modo estándar de información de base de datos como parte de la habilitación de la base de datos ilimitada de Aurora PostgreSQL. Puede usarla para supervisar la carga de la base de datos (carga de base de datos) de las instancias de base de datos (DB) ilimitada en tiempo real. La carga de la base de datos mide el nivel de actividad de la sesión en una base de datos. Puede utilizar información sobre las bases de datos para analizar y solucionar problemas del rendimiento de las instancias de bases de datos (DB) ilimitadas de Aurora PostgreSQL a escala.

Para obtener más información acerca de la información de base datos de CloudWatch, consulte lo siguiente.

- Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch
- Información de bases de datos de CloudWatch en la Guía del usuario de Amazon CloudWatch
- Introducción a información de bases de datos de CloudWatch en la Guía del usuario de Amazon CloudWatch
- Configuración de la base de datos para supervisar las consultas SQL lentas con Información sobre las bases de datos para Amazon Aurora

Para obtener información sobre cómo activar el modo avanzado o el modo estándar de Información sobre las bases de datos, consulte los siguientes temas.

Temas

- Activación del modo avanzado de información sobre las bases de datos para base de datos ilimitada de Aurora PostgreSQL
- Activación del modo estándar de información sobre las bases de datos para base de datos ilimitada de Aurora PostgreSQL

Activación del modo avanzado de información sobre las bases de datos para base de datos ilimitada de Aurora PostgreSQL

Para activar el modo avanzado de información sobre las bases de datos para la base de datos ilimitada de Aurora PostgreSQL, use los siguientes procedimientos.

Activación del modo avanzado de información sobre las bases de datos al crear un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL

Active el modo avanzado de información sobre las bases de datos al crear una base de datos para la base de datos ilimitada de Aurora PostgreSQL.

Console

En la consola, puede activar el modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo avanzado de información sobre las bases de datos al crear un clúster de base de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija Creación de base de datos.
4. En la sección Información sobre bases de datos, seleccione el Modo avanzado. A continuación, elija las siguientes opciones:
   - Retención: el número de días durante los que se conservan los datos de Información de rendimiento. El período de retención debe ser de 15 a 24 meses para el modo avanzado de Información sobre las bases de datos.
   - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Creación de base de datos.

AWS CLI

Para activar el modo avanzado de la Información sobre las bases de datos al crear un clúster de bases de datos, llame al comando create-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--db-cluster-identifier`: el identificador del clúster de bases de datos.
- `--database-insights-mode advanced` para activar el modo avanzado de Información sobre las bases de datos.
- `--engine`: el clúster de base de datos debe utilizar el motor de base de datos `aurora-postgresql`.
- `--engine-version`: el clúster de base de datos debe utilizar una de las versiones del motor de base de datos:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: el clúster de base de datos debe utilizar la configuración de almacenamiento del clúster de base de datos `aurora-iopt1`.
- `--cluster-scalability-type`: especifica el modo de escalabilidad del clúster de base de datos de Aurora. Cuando se establece en `limitless`, el clúster funciona como una Base de datos ilimitada de Aurora PostgreSQL. Si se establece en `standard` (valor predeterminado), el clúster utiliza la creación normal de instancias de base de datos.

  > **Note**
  >
  > No puede modificar esta configuración después de crear el clúster de base de datos.

- `--master-username`: es el nombre del usuario maestro del clúster de base de datos.
- `--master-user-password`: es la contraseña para el usuario maestro.
- `--enable-performance-insights` para activar Información sobre rendimiento para Información sobre las bases de datos.
- `--performance-insights-retention-period`: es el período de retención de los datos de su clúster de bases de datos. Para activar Información sobre las bases de datos, el período de retención debe ser de al menos 465 días.
- `--monitoring-interval`: es el intervalo, en segundos, entre puntos cuando se recopilan las métricas de Monitorización mejorada para el clúster de base de datos. Este valor no puede ser `0`.
- `--monitoring-role-arn`: es el nombre de recurso de Amazon (ARN) del rol de IAM que permite a RDS enviar métricas de Monitorización mejorada a Registros de Amazon CloudWatch.

- `--enable-cloudwatch-logs-exports`: debe exportar los registros de `postgresql` a Registros de CloudWatch.

En el siguiente ejemplo se habilita el modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode advanced \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 465 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Para Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode advanced ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 465 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Para activar el modo avanzado de Información sobre las bases de datos al crear un clúster de bases de datos multi-AZ, especifique los siguientes parámetros para la operación de la API CreateDBCluster de Amazon RDS.

- De `DatabaseInsightsMode` a `advanced`
- De `Engine` a `aurora-postgresql`
- `EngineVersion` a una versión de motor disponible para bases de datos ilimitadas
- De `StorageType` a `aurora-iopt1`
- De `ClusterScalabilityType` a `limitless`
- `MasterUsername`
- `MasterUserPassword`
- De `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` en al menos `465` días
- `MonitoringInterval` a un valor que no sea `0`
- `MonitoringRoleArn` al nombre de recurso de Amazon (ARN) del rol de IAM que permite a RDS enviar métricas de supervisión mejorada a Registros de Amazon CloudWatch

Activación del modo avanzado de información sobre bases de datos al modificar un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL

Active la información sobre las bases de datos al modificar una base de datos para la base de datos ilimitada de Aurora PostgreSQL.

> **Note**
>
> Para habilitar información sobre las bases de datos, cada instancia de base de datos de un clúster de base de datos debe tener la misma configuración de información de rendimiento y supervisión mejorada.

Console

En la consola, puede activar el modo avanzado de Información sobre las bases de datos al modificar un clúster de bases de datos. La configuración de Información sobre las bases de datos

se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo avanzado de información sobre las bases de datos al modificar un clúster de base de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija un clúster de base de datos y elija Modificar.
4. En la sección Información sobre bases de datos, seleccione el Modo avanzado. A continuación, elija las siguientes opciones:
   - Retención: el número de días durante los que se conservan los datos de Información de rendimiento. El período de retención debe ser de 15 a 24 meses para el modo avanzado de Información sobre las bases de datos.
   - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Continuar.
6. En Programación de modificaciones, elija Aplicar inmediatamente. Si elige Aplicar durante el próximo periodo de mantenimiento programado, la base de datos ignora esta configuración y activa de inmediato el modo avanzado de Información sobre las bases de datos.
7. Elija Modificar clúster.

AWS CLI

Para activar el modo avanzado de Información sobre las bases de datos al modificar un clúster de bases de datos, llame al comando modify-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--database-insights-mode advanced` para activar el modo avanzado de Información sobre las bases de datos.
- `--db-cluster-identifier`: el identificador del clúster de bases de datos.

- `--enable-performance-insights` para activar Información sobre rendimiento para Información sobre las bases de datos.
- `--performance-insights-retention-period`: es el periodo de retención de los datos del clúster de bases de datos. Para activar el modo avanzado de Información sobre las bases de datos, el período de retención debe ser de al menos 465 días.

En el siguiente ejemplo se habilita el modo avanzado de Información sobre las bases de datos al modificar un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Para activar el modo avanzado de información sobre las bases de datos al modificar un clúster de base de datos, especifique los siguientes parámetros para la operación de la API de Amazon RDS ModifyDBCluster.

- De `DatabaseInsightsMode` a `advanced`
- De `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` en al menos 465 días

Activación del modo estándar de información sobre las bases de datos para base de datos ilimitada de Aurora PostgreSQL

Para activar el modo estándar de información sobre las bases de datos para la base de datos ilimitada de Aurora PostgreSQL, use los siguientes procedimientos.

Activación del modo estándar de información sobre las bases de datos al crear un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL

Active el modo estándar de información sobre las bases de datos al crear una base de datos para la base de datos ilimitada de Aurora PostgreSQL.

Console

En la consola, puede activar el modo estándar de Información sobre las bases de datos al crear un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo estándar de información sobre las bases de datos al crear un clúster de base de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija Creación de base de datos.
4. En la sección Información sobre las bases de datos, seleccione el Modo estándar. A continuación, elija las siguientes opciones:
   - Retención: el número de días durante los que se conservan los datos de Información de rendimiento. Para crear un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL, el periodo de retención debe ser de al menos 31 días.
   - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Creación de base de datos.

AWS CLI

Para activar el modo estándar de la Información sobre las bases de datos al crear un clúster de bases de datos, llame al comando create-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--db-cluster-identifier`: el identificador del clúster de bases de datos.
- `--database-insights-mode standard` para activar el modo estándar de Información sobre las bases de datos.
- `--engine`: el clúster de base de datos debe utilizar el motor de base de datos `aurora-postgresql`.
- `--engine-version`: el clúster de base de datos debe utilizar una de las versiones del motor de base de datos:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: el clúster de base de datos debe utilizar la configuración de almacenamiento del clúster de base de datos `aurora-iopt1`.
- `--cluster-scalability-type`: especifica el modo de escalabilidad del clúster de base de datos de Aurora. Cuando se establece en `limitless`, el clúster funciona como una Base de datos ilimitada de Aurora PostgreSQL. Si se establece en `standard` (valor predeterminado), el clúster utiliza la creación normal de instancias de base de datos.

  > **Note**
  >
  > No puede modificar esta configuración después de crear el clúster de base de datos.

- `--master-username`: es el nombre del usuario maestro del clúster de base de datos.
- `--master-user-password`: es la contraseña para el usuario maestro.
- `--enable-performance-insights` para activar Información sobre rendimiento para Información sobre las bases de datos.
- `--performance-insights-retention-period`: es el período de retención de los datos de su clúster de bases de datos. Para crear un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL, el periodo de retención debe ser de al menos 31 días.

- `--monitoring-interval`: es el intervalo, en segundos, entre puntos cuando se recopilan las métricas de Monitorización mejorada para el clúster de base de datos. Este valor no puede ser `0`.
- `--monitoring-role-arn`: es el nombre de recurso de Amazon (ARN) del rol de IAM que permite a RDS enviar métricas de Monitorización mejorada a Registros de Amazon CloudWatch.
- `--enable-cloudwatch-logs-exports`: debe exportar los registros de `postgresql` a Registros de CloudWatch.

En el siguiente ejemplo se habilita el modo estándar de información sobre las bases de datos al crear un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode standard \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 31 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Para Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode standard ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
```

```
--enable-performance-insights ^
--performance-insights-retention-period 31 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Para activar el modo estándar de información sobre las bases de datos al crear una , especifique los siguientes parámetros para la operación de la API de Amazon RDS CreateDBCluster.

- De `DatabaseInsightsMode` a `standard`
- De `Engine` a `aurora-postgresql`
- `EngineVersion` a una versión de motor disponible para bases de datos ilimitadas
- De `StorageType` a `aurora-iopt1`
- De `ClusterScalabilityType` a `limitless`
- `MasterUsername`
- `MasterUserPassword`
- De `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` en al menos `31` días
- `MonitoringInterval` a un valor que no sea `0`
- `MonitoringRoleArn` al nombre de recurso de Amazon (ARN) del rol de IAM que permite a RDS enviar métricas de supervisión mejorada a Registros de Amazon CloudWatch

Activación del modo estándar de información sobre las bases de datos al modificar un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL

Active la información sobre las bases de datos al modificar una base de datos para la base de datos ilimitada de Aurora PostgreSQL.

> **Note**
>
> Para habilitar Información sobre las bases de datos, cada instancia de base de datos de un clúster de bases de datos debe tener la misma configuración de Información de rendimiento y Supervisión mejorada.

Console

En la consola, puede activar el modo estándar de Información sobre las bases de datos al crear un clúster de bases de datos. La configuración de Información sobre las bases de datos se aplica a todas las instancias de bases de datos para todas las instancias de su clúster de bases de datos.

Activación del modo estándar de información sobre las bases de datos al modificar un clúster de base de datos con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija un clúster de base de datos y elija Modificar.
4. En la sección Información sobre las bases de datos, seleccione el Modo estándar. A continuación, elija las siguientes opciones:
   - Retención: el número de días durante los que se conservan los datos de Información de rendimiento. Para crear un clúster de base de datos para la base de datos ilimitada de Aurora PostgreSQL, el periodo de retención debe ser de al menos 31 días.
   - AWS KMS key: especifique su clave de KMS. Performance Insights cifra todos los datos potencialmente confidenciales con su propia clave de KMS. Las datos se cifran en reposo y en tránsito. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora.
5. Elija Continuar.
6. En Programación de modificaciones, elija Aplicar inmediatamente. Si elige Aplicar durante el próximo periodo de mantenimiento programado, la base de datos ignora esta configuración y activa de inmediato el modo estándar de Información sobre las bases de datos.
7. Elija Modificar clúster.

AWS CLI

Para activar el modo estándar de Información sobre las bases de datos al modificar un clúster de bases de datos, llame al comando modify-db-cluster de la AWS CLI e introduzca los siguientes valores:

- `--db-cluster-identifier`: el identificador del clúster de bases de datos.

- `--database-insights-mode standard` para activar el modo estándar de Información sobre las bases de datos.
- `--enable-performance-insights` para activar Información sobre rendimiento para Información sobre las bases de datos.
- `--performance-insights-retention-period`: es el período de retención de los datos de su clúster de bases de datos. Para activar el modo estándar de información sobre las bases de datos, el periodo de retención debe ser de al menos 31 días.

En el siguiente ejemplo se habilita el modo estándar de información sobre las bases de datos al modificar un clúster de bases de datos.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 31
```

Para Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 31
```

RDS API

Para activar el modo estándar de información sobre las bases de datos al modificar un clúster de base de datos, especifique los siguientes parámetros para la operación de la API de Amazon RDS ModifyDBCluster.

- De `DatabaseInsightsMode` a `standard`
- De `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` en al menos 31 días

Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Registros de Amazon CloudWatch

Para habilitar Base de datos ilimitada de Aurora PostgreSQL, es necesario exportar los registros de PostgreSQL a Registros de CloudWatch. Puede acceder a estos registros y analizarlos en Información de registros de CloudWatch, que es similar a acceder a los registros de PostgreSQL para un clúster de base de datos Aurora PostgreSQL estándar. Para obtener más información, consulte Análisis de los registros de PostgreSQL mediante CloudWatch Logs Insights.

El nombre del grupo de registro del clúster de base de datos es el mismo que en Aurora PostgreSQL:

```
/aws/rds/cluster/DB_cluster_ID/postgresql
```

El nombre del grupo de registro del grupo de particiones de base de datos tiene el siguiente formato:

```
/aws/rds/cluster/DB_cluster_ID/DB_shard_group_ID/postgresql
```

Hay flujos de registro para cada nodo (enrutador o partición). Los nombres tienen el siguiente formato:

```
[DistributedTransactionRouter|
DataAccessShard]/node_cluster_serial_ID-node_instance_serial_ID/n
```

Por ejemplo:

- Enrutador: `DistributedTransactionRouter/6-6.2`
- Partición: `DataAccessShard/22-22.0`

> **Note**
>
> No puede ver los archivos de registro de PostgreSQL del grupo de particiones de base de datos directamente en la consola de RDS, en AWS CLI o en la API de RDS, como hace en el clúster de base de datos. Para verlos necesita Información de registros de CloudWatch.

Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Monitorización mejorada

Se requiere Monitorización mejorada para activar Base de datos ilimitada de Aurora PostgreSQL. Puede usarla para supervisar el sistema operativo de sus instancias de base de datos de Base de datos ilimitada en tiempo real.

Aurora publica las métricas de Monitorización mejorada en Registros de CloudWatch Algunas de las métricas clave disponibles incluyen las conexiones a las bases de datos, el uso del almacenamiento y la latencia de las consultas. Estas métricas pueden ayudarle a identificar los cuellos de botella de rendimiento.

Para obtener más información acerca de las métricas de Supervisión mejorada, consulte Métricas del sistema operativo para Aurora.

Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Información de rendimiento

Utilice Información de rendimiento para supervisar su clúster de Base de datos ilimitada de Aurora PostgreSQL. Información de rendimiento para Base de datos ilimitada de Aurora PostgreSQL funciona de manera similar que para los clústeres de bases de datos de Aurora estándar. Sin embargo, usted realiza un seguimiento de las métricas en el nivel de grupo de particiones para Base de datos ilimitada de Aurora PostgreSQL.

Las dos métricas principales de Información de rendimiento que hay que seguir son las siguientes:

- Carga de base de datos: mide el nivel de actividad en la base de datos. La métrica clave en Información sobre rendimiento es `DBLoad`, que se recopila cada segundo.

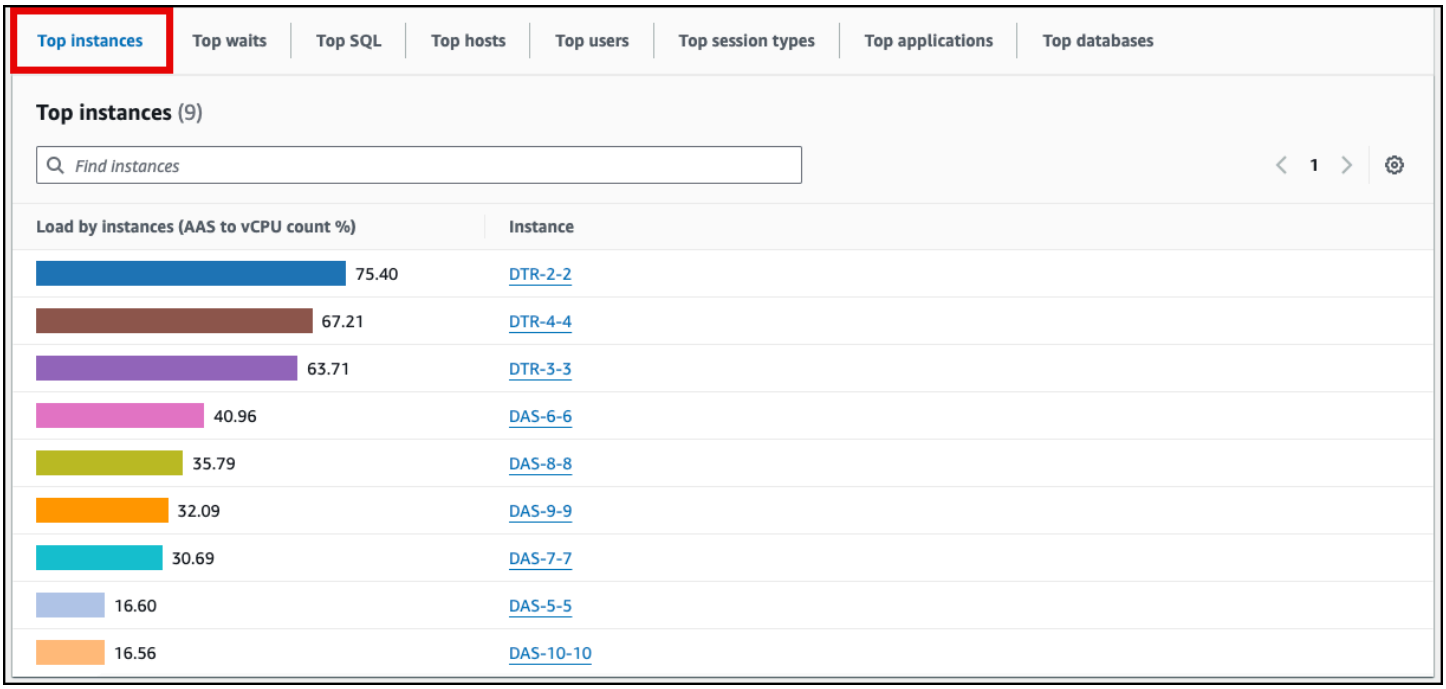
  El promedio de sesiones activas (AAS) es la unidad para la métrica `DBLoad` en Información de rendimiento. Para obtener un promedio de sesiones activas, la característica de Performance Insights hace un muestreo del número de sesiones que ejecutan una consulta al mismo tiempo. El AAS es el total del número de sesiones dividido entre el total del número de muestras para un periodo de tiempo específico. Para obtener más información sobre `DBLoad` y el AAS, consulte Carga de base de datos.
- CPU máxima: es la potencia computacional máxima disponible para la base de datos. Para ver si las sesiones activas superan el máximo de la CPU, observe su relación con la línea `Max vCPU`. El valor `Max vCPU` viene determinado por el número de núcleos de vCPU (CPU virtual) de la instancia de base de datos. Para obtener más información sobre `Max vCPU`, consulte Máximo de la CPU.

Además, puede “dividir” la métrica `DBLoad` en dimensiones, que son subcategorías de la métrica. Las dimensiones más útiles son las siguientes:

- Instancias principales: muestran la carga de base de datos relativa de sus instancias (particiones y enrutadores) en orden descendente.
- Eventos de espera: provocan que una instrucción SQL espere a que ocurran eventos específicos antes de que pueda continuar ejecutándose. Los eventos de espera indican dónde se impide el trabajo.
- SQL principal: muestra qué consultas contribuyen más a la carga de la base de datos.

Para obtener más información acerca de las dimensiones de la información de rendimiento, consulte Dimensiones.

En la siguiente figura se muestra la dimensión Instancias principales de un grupo de particiones de base de datos.



Temas

- [Análisis de la carga de base de datos para Base de datos ilimitada de Aurora PostgreSQL con el panel Información de rendimiento]

Análisis de la carga de base de datos para Base de datos ilimitada de Aurora PostgreSQL con el panel Información de rendimiento

Con Información de rendimiento, puede realizar un seguimiento de las métricas por grupo de partición o instancia para una Base de datos ilimitada de Aurora PostgreSQL. Al analizar la carga de base de datos de una Base de datos ilimitada de Aurora PostgreSQL, se recomienda comparar la carga de base de datos de cada partición y enrutador con la vCPU máxima.

> **Note**
>
> Base de datos ilimitada de Aurora PostgreSQL siempre tiene habilitados Información de rendimiento y Monitorización mejorada. El período mínimo de retención de los datos de Información de rendimiento para Base de datos ilimitada es de 31 días (1 mes).

La vista Absoluta muestra el número promedio de sesiones activas (AAS) y la vCPU estimada. La vista Relativa muestra la relación entre el AAS y la vCPU estimada.

Temas

- Análisis de la carga de base de datos relativa para Base de datos ilimitada de Aurora PostgreSQL con el panel Información de rendimiento
- Análisis de la carga de base de datos por esperas para Base de datos ilimitada de Aurora PostgreSQL con el panel Información de rendimiento
- Análisis de la distribución de la carga para Base de datos ilimitada de Aurora PostgreSQL con el panel de Información de rendimiento

Análisis de la carga de base de datos relativa para Base de datos ilimitada de Aurora PostgreSQL con el panel Información de rendimiento

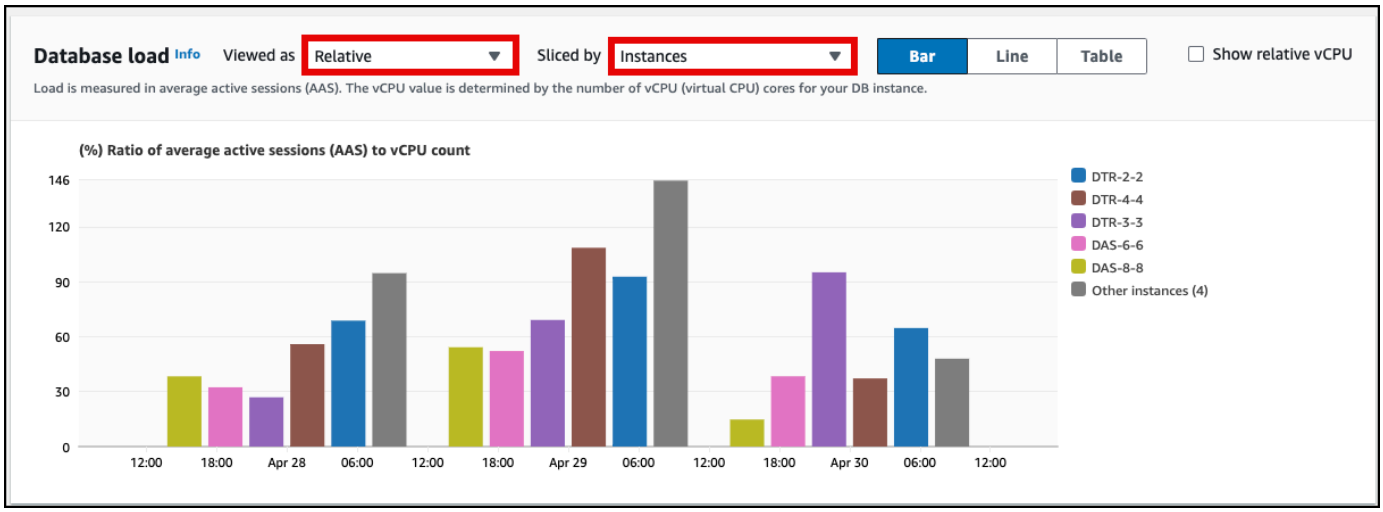
Es posible que desee mejorar el rendimiento de su Base de datos ilimitada de Aurora PostgreSQL mediante el seguimiento de la carga de base de datos relativa. Para analizar la carga de la base de datos relativa por instancia de su Base de datos ilimitada de Aurora PostgreSQL, utilice el siguiente procedimiento.

Análisis de la carga de la base de datos relativa para Base de datos ilimitada de Aurora PostgreSQL mediante la consola

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

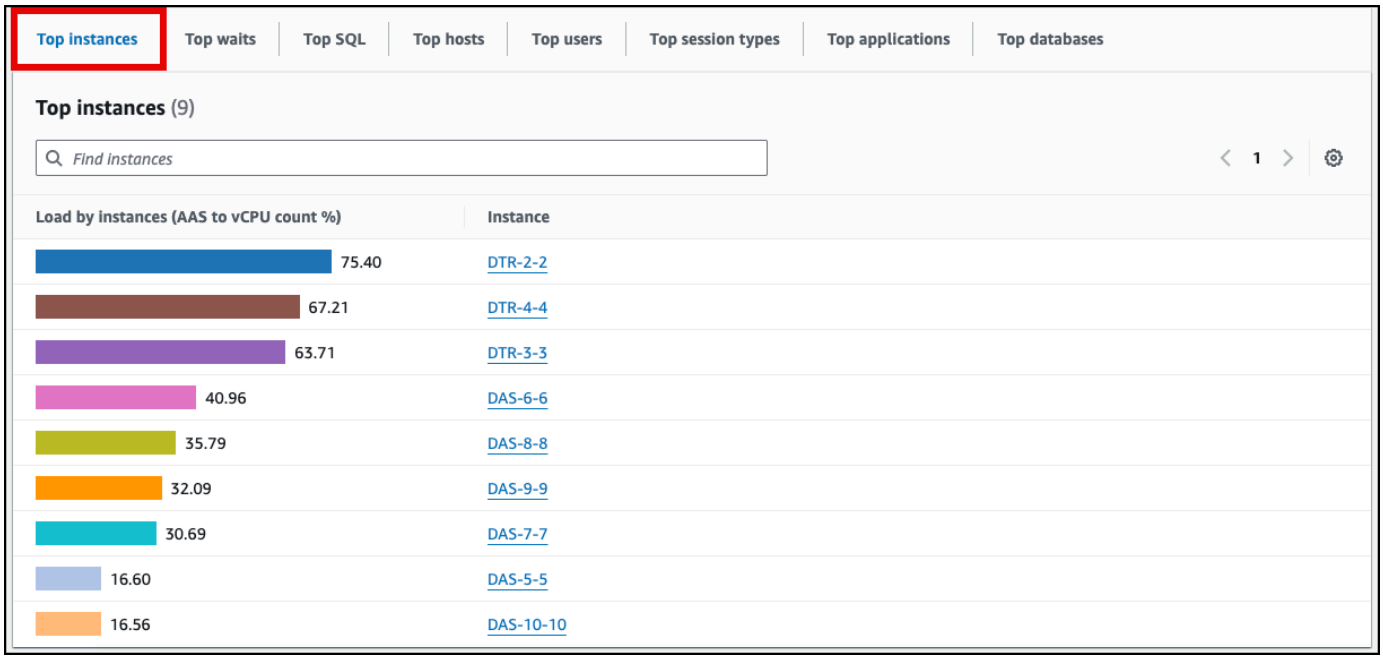
2. En el panel de navegación, seleccione Información de rendimiento.
3. Seleccione una Base de datos ilimitada de Aurora PostgreSQL. Aparecerá el panel Información de rendimiento para esa Base de datos ilimitada de Aurora PostgreSQL.
4. En la sección Carga de base de datos, elija Instancias junto a Dividido por. Para ver la proporción entre el promedio de sesiones activas (AAS) y los núcleos de vCPU de todas las instancias de Base de datos ilimitada de Aurora PostgreSQL, seleccione Relativo en Visto como.

   El gráfico de sesiones activas promedio muestra la carga de datos para las instancias de su Base de datos ilimitada de Aurora PostgreSQL.



5. Para ver las instancias principales, seleccione la pestaña Instancias principales.

   En el siguiente ejemplo, la instancia con carga de base de datos más alta es `DTR-2-2`.



6. (Opcional) Para analizar la carga de base de datos de una instancia en su Base de datos ilimitada de Aurora PostgreSQL, elija el nombre de la instancia en la columna Instancias. Para ver la carga de base de datos para `DTR-2-2`, elija `DTR-2-2` en la columna Instancias.

> **Note**
>
> Puede ver las métricas de Información de rendimiento solo para las instancias de Base de datos ilimitada de Aurora PostgreSQL.

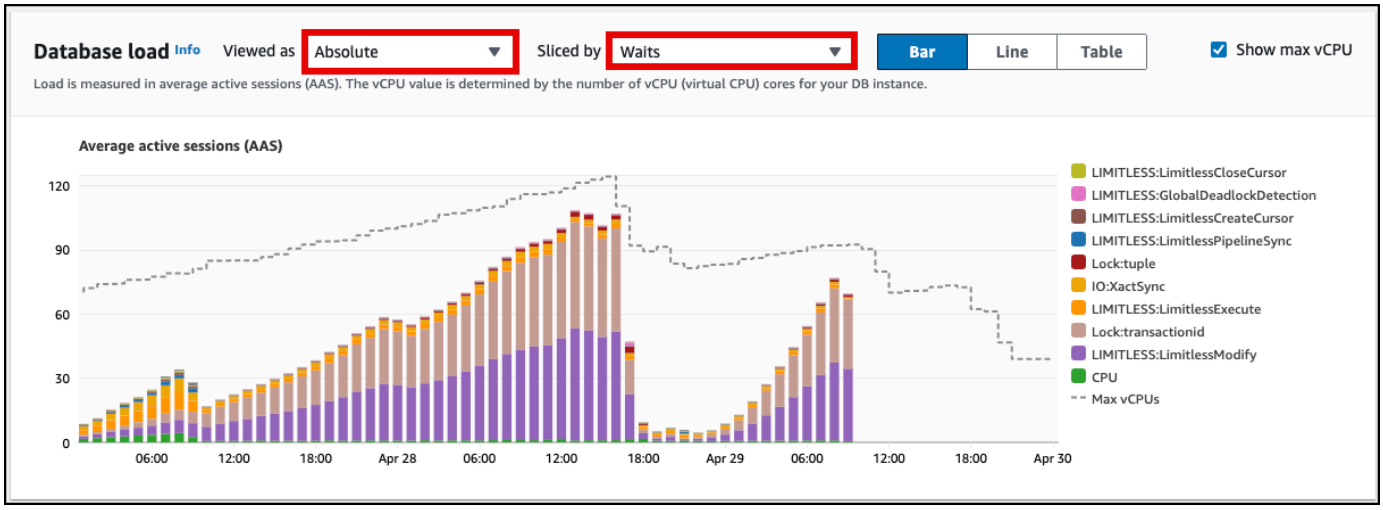
Análisis de la carga de base de datos por esperas para Base de datos ilimitada de Aurora PostgreSQL con el panel Información de rendimiento

Es posible que desee mejorar el rendimiento de su Base de datos ilimitada de Aurora PostgreSQL mediante el seguimiento de los eventos de espera. Para analizar la carga de la base de datos por eventos de espera de su Base de datos ilimitada de Aurora PostgreSQL, utilice el siguiente procedimiento.

Análisis de la carga de la base de datos por esperas para Base de datos ilimitada de Aurora PostgreSQL mediante la consola

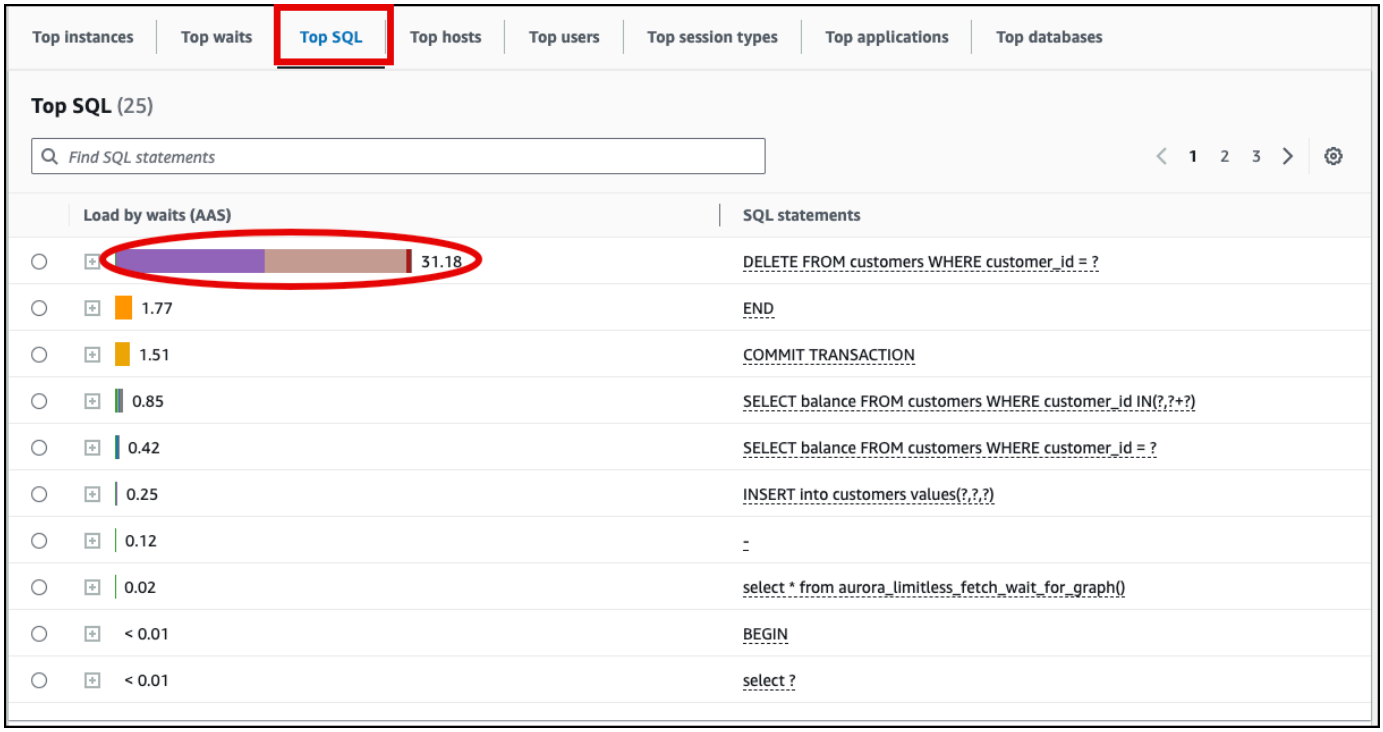
1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información de rendimiento.
3. Seleccione una Base de datos ilimitada de Aurora PostgreSQL. Aparecerá el panel Información de rendimiento para esa Base de datos ilimitada de Aurora PostgreSQL.
4. En la sección Carga de base de datos, elija Esperas junto a Dividido por. Para ver la cantidad de AAS y la vCPU estimada, seleccione Absoluta en Visto como.

   El gráfico de sesiones activas promedio muestra la carga de datos para las instancias de su Base de datos ilimitada de Aurora PostgreSQL.



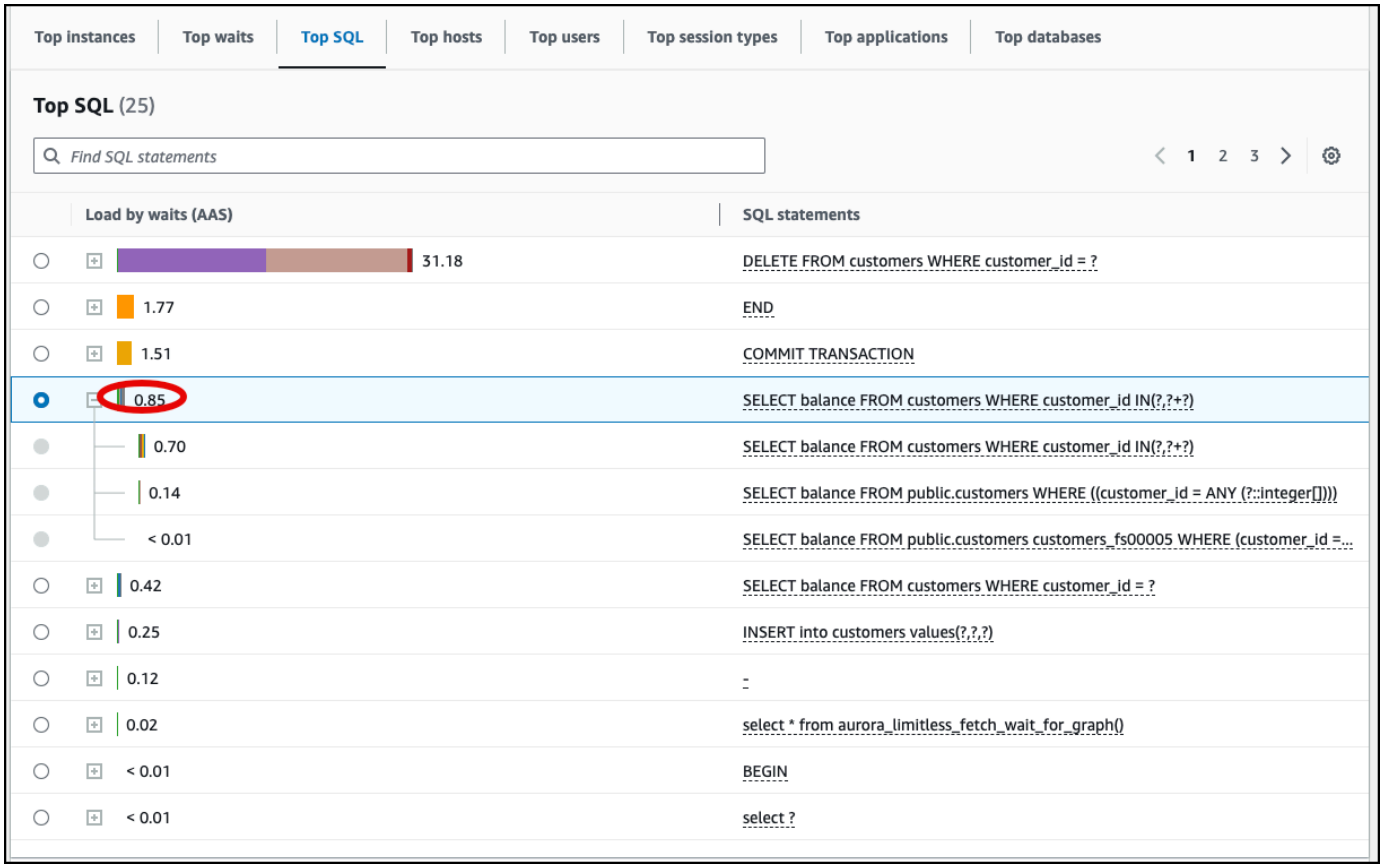
5. Desplácese hasta la pestaña SQL principal.

   En el siguiente ejemplo, la instrucción SQL con la mayor carga por esperas es la instrucción `DELETE`.



6. Elija la instrucción SQL para expandirla hasta mostrar sus instrucciones componentes.

   En el siguiente ejemplo, la instrucción `SELECT` tiene 3 instrucciones componentes.



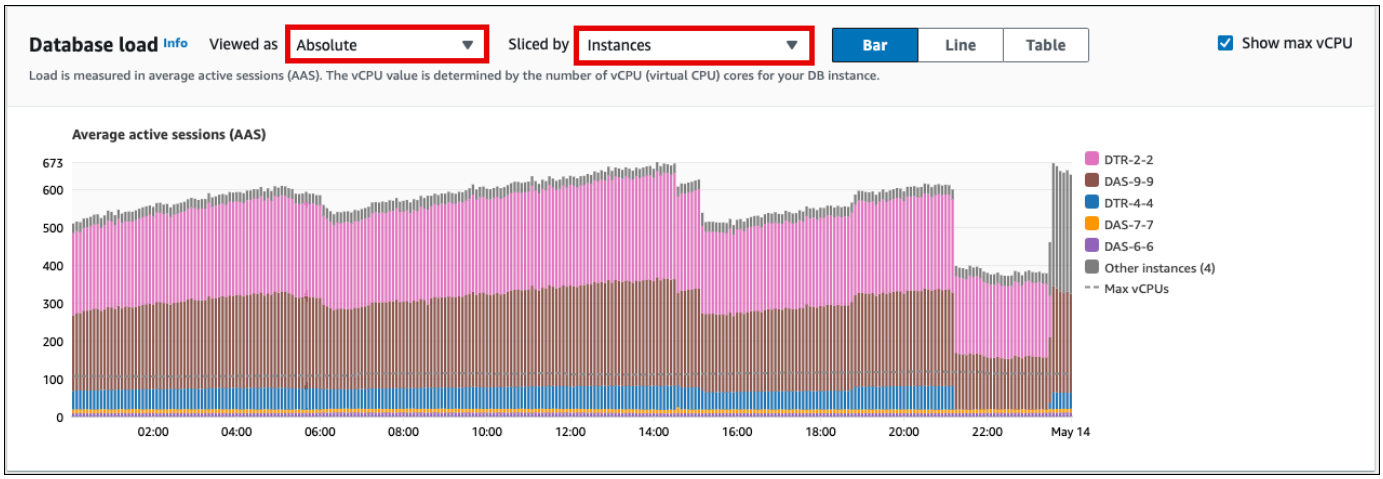
Análisis de la distribución de la carga para Base de datos ilimitada de Aurora PostgreSQL con el panel de Información de rendimiento

Es posible que quiera equilibrar la distribución de carga de las instancias de Base de datos ilimitada de Aurora PostgreSQL. Para analizar la distribución de carga de las instancias en una Base de datos ilimitada de Aurora PostgreSQL, utilice el siguiente procedimiento.

Análisis de la distribución de carga de las instancias en una Base de datos ilimitada de Aurora PostgreSQL mediante la consola

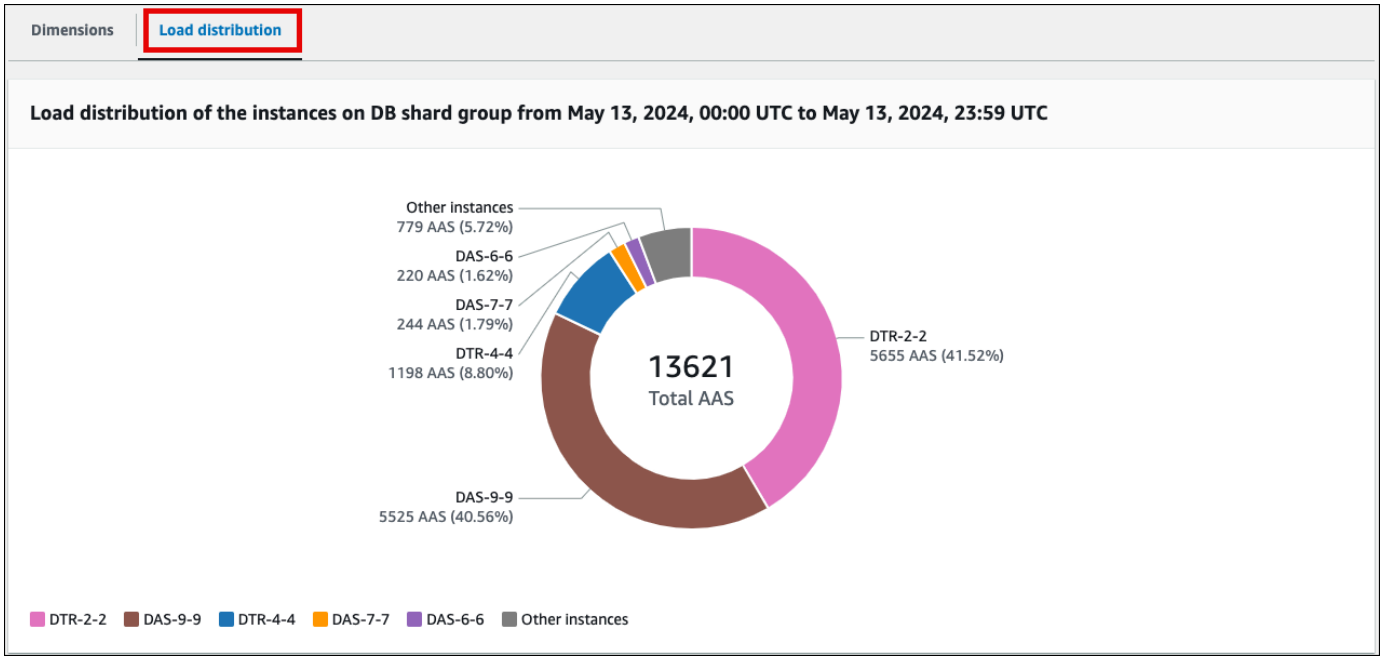
1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Información de rendimiento.
3. Seleccione una Base de datos ilimitada de Aurora PostgreSQL. Aparecerá el panel Información de rendimiento para esa Base de datos ilimitada de Aurora PostgreSQL.
4. En la sección Carga de base de datos, elija Instancias junto a Dividido por. Para ver la cantidad de AAS y la vCPU estimada para todas las instancias de su Base de datos ilimitada de Aurora PostgreSQL, seleccione Absoluta para Dividido por.

El gráfico de sesiones activas promedio muestra la carga de datos para las instancias de su Base de datos ilimitada de Aurora PostgreSQL.



5. Para ver un gráfico de la distribución de carga de las instancias de su Base de datos ilimitada de Aurora PostgreSQL, seleccione la pestaña Distribución de carga.

   En el siguiente ejemplo, la instancia con carga de base de datos más alta es `DTR-2-2`.



Supervisión de Base de datos ilimitada de Aurora PostgreSQL con Amazon GuardDuty para protección de RDS

Amazon GuardDuty es un servicio de detección de amenazas que ayuda a proteger las cuentas, los contenedores, las cargas de trabajo y los datos de su entorno de AWS. Mediante modelos de machine learning (ML) y capacidades de detección de anomalías y amenazas, GuardDuty supervisa continuamente los diferentes orígenes de registro y la actividad en tiempo de ejecución para identificar y priorizar los posibles riesgos de seguridad y actividades maliciosas en su entorno.

GuardDuty RDS Protection analiza y perfila los eventos de inicio de sesión para detectar posibles amenazas de acceso a sus bases de datos de Amazon Aurora. Al activar RDS Protection, GuardDuty consume los eventos de inicio de sesión de RDS de sus bases de datos de Aurora. RDS Protection supervisa estos eventos y los perfila para detectar posibles amenazas internas o agentes externos.

Para obtener más información sobre GuardDuty para protección de RDS en Aurora, consulte Supervisión de amenazas con Amazon GuardDuty para protección de RDS para Amazon Aurora.

Para obtener más información sobre la forma de habilitar la protección de RDS de Amazon GuardDuty, consulte GuardDuty RDS Protection (Protección de RDS de Amazon GuardDuty) en la Guía del usuario de Amazon GuardDuty.

Funciones y vistas de Base de datos ilimitada de Aurora PostgreSQL

Base de datos ilimitada de Aurora PostgreSQL incluye funciones y vistas, que se basan en las funciones y vistas de Aurora PostgreSQL correspondientes.

> **Note**
>
> Algunas estadísticas pueden arrojar resultados incoherentes si hay transacciones en curso.

Temas

- Funciones de Base de datos ilimitada de Aurora PostgreSQL
- Vistas de Base de datos ilimitada de Aurora PostgreSQL

Funciones de Base de datos ilimitada de Aurora PostgreSQL

En la tabla siguiente se muestran las nuevas funciones de Base de datos ilimitada de Aurora PostgreSQL.

> **Note**
>
> Las funciones enumeradas en esta tabla se encuentran en el esquema `rds_aurora`. Cuando utilice una función de Base de datos ilimitada, asegúrese de incluir el nombre del objeto totalmente cualificado: `rds_aurora.`*`object_name`*.

Función de Base de datos ilimitada de Aurora PostgreSQL	Función de Aurora PostgreSQL correspon diente
limitless_backend_dsid	pg_backend_pid
limitless_cancel_session	pg_cancel_backend
limitless_stat_clear_snapshot	pg_stat_clear_snapshot
limitless_stat_database_size	pg_database_size

Función de Base de datos ilimitada de Aurora PostgreSQL	Función de Aurora PostgreSQL correspon diente
limitless_stat_get_snapshot_timestamp	pg_stat_get_snapshot_timestamp
limitless_stat_prepared_xacts	pg_prepared_xacts
limitless_stat_relation_sizes	pg_indexes_size, pg_relation_size, pg_table_size, pg_total_relation_size
limitless_stat_reset	pg_stat_reset
limitless_stat_statements_reset	pg_stat_statements_reset
limitless_stat_system_waits	aurora_stat_system_waits
limitless_terminate_session	pg_terminate_backend
limitless_wait_report	aurora_wait_report

Los siguientes ejemplos proporcionan detalles sobre las funciones de Base de datos ilimitada de Aurora PostgreSQL. Para obtener más información acerca de las funciones de PostgreSQL, consulte Functions and operators en la documentación de PostgreSQL.

limitless_backend_dsid

La función `limitless_backend_dsid` devuelve el ID de la sesión actual distribuido. Una sesión distribuida se ejecuta en un enrutador de un grupo de particiones de base de datos e implica procesos de backend en una o más particiones del grupo de particiones de base de datos.

El siguiente ejemplo muestra cómo utilizar la función `limitless_backend_dsid`.

```
SELECT rds_aurora.limitless_backend_dsid();

limitless_backend_dsid
------------------------
8CACD7B04D0FC2A5
(1 row)
```

limitless_cancel_session

La función `limitless_cancel_session` es similar a `pg_cancel_backend`, pero intenta cancelar todos los procesos de backend relacionados con el ID de sesión distribuido proporcionado mediante el envío de una `SIGINT` (señal de interrupción).

El parámetro de entrada es el siguiente:

- `distributed_session_id` (texto): es el ID de la sesión distribuida que se va a cancelar.

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `pid` (texto): es el ID del proceso de backend.
- `success` (booleano): indica si la cancelación se ha realizado correctamente.

El siguiente ejemplo muestra cómo utilizar la función `limitless_cancel_session`.

```
SELECT * FROM rds_aurora.limitless_cancel_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_stat_clear_snapshot

La función `limitless_stat_clear_snapshot` descarta la instantánea actual de las estadísticas o la información almacenada en caché en todos los nodos.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_clear_snapshot`.

```
SELECT rds_aurora.limitless_stat_clear_snapshot();
```

limitless_stat_database_size

La función `limitless_stat_database_size` devuelve los tamaños de una base de datos del grupo de particiones de base de datos.

El parámetro de entrada es el siguiente:

- `dbname` (nombre): es la base de datos de la que se van a obtener los tamaños.

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `db_size`: es el tamaño de la base de datos de este subclúster en bytes.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_database_size`.

```
SELECT * FROM rds_aurora.limitless_stat_database_size('postgres_limitless');

 subcluster_id | subcluster_type | db_size
---------------+-----------------+----------
             1 | router          |  8895919
             2 | router          |  8904111
             3 | shard           | 21929391
             4 | shard           | 21913007
             5 | shard           | 21831087
(5 rows)
```

limitless_stat_get_snapshot_timestamp

La función `limitless_stat_get_snapshot_timestamp` devuelve la marca de tiempo de la instantánea actual de las estadísticas o `NULL` si no se ha realizado ninguna instantánea de estadísticas. Se toma una instantánea la primera vez que se accede a las estadísticas acumuladas de una transacción si `stats_fetch_consistency` se establece en `snapshot`. Devuelve una vista consolidada de las marcas temporales de las instantáneas de todos los nodos. Las columnas `subcluster_id` y `subcluster_type` muestran de qué nodo provienen los datos.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_get_snapshot_timestamp`.

```
SELECT * FROM rds_aurora.limitless_stat_get_snapshot_timestamp();

 subcluster_id | subcluster_type | snapshot_timestamp
---------------+-----------------+--------------------
             1 | router          |
             2 | router          |
             3 | shard           |
             4 | shard           |
```

```
             5 | shard           |
(5 rows)
```

limitless_stat_prepared_xacts

La función `limitless_stat_prepared_xacts` devuelve información sobre las transacciones de todos los nodos que están actualmente preparados para la confirmación en dos fases. Para más información, consulte pg_prepared_xacts en la documentación de PostgreSQL.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_prepared_xacts`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_prepared_xacts;

 subcluster_id | subcluster_type | transaction_id |             gid              |
         prepared            |  owner_id  |    database_id
---------------+-----------------+----------------+------------------------------
+-------------------------------+------------+--------------------
 8             | shard           |        5815978 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659603+00 | auroraperf | postgres_limitless
 12            | shard           |        4599138 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659637+00 | auroraperf | postgres_limitless
(2 rows)
```

limitless_stat_relation_sizes

La función `limitless_stat_relation_sizes` devuelve tamaños distintos de una tabla del grupo de particiones de base de datos.

Los parámetros de entrada son los siguientes:

- `relnspname` (nombre): es el nombre del esquema que contiene la tabla.
- `relname` (nombre): es el nombre de la tabla.

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `main_size`: es el tamaño en bytes de la bifurcación de datos principal de este nodo.
- `fsm_size`: es el tamaño en bytes del mapa del espacio libre de la tabla de este nodo.
- `vm_size`: es el tamaño en bytes del mapa de visibilidad de la tabla de este nodo.

- `init_size`: es el tamaño en bytes de la inicialización de la tabla en este nodo.
- `toast_size`: es el tamaño en bytes de la tabla de TOAST asociada a la tabla de esta bifurcación.
- `index_size`: es el tamaño en bytes de todos los índices de la tabla en este nodo.
- `total_size`: es el tamaño en bytes de todos los segmentos de la tabla en este nodo.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_relation_sizes` (se han omitido algunas columnas).

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customers');

 subcluster_id | subcluster_type | main_size | fsm_size | vm_size | toast_size |
 table_size | total_size
---------------+-----------------+-----------+----------+---------+------------
+------------+------------
             1 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             2 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             3 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             4 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             5 | shard           |   3981312 |  4227072 | 1409024 |    1409024 |
 11026432 |   11026432
(5 rows)
```

limitless_stat_reset

La función `limitless_stat_reset` restablece todos los contadores de estadísticas de la base de datos actual a cero (0). Si `track_functions` está activado, la columna `stats_reset` de `limitless_stat_database` mostrará la última vez que se restablecieron las estadísticas de la base de datos. De forma predeterminada, solo un superusuario puede ejecutar `limitless_stat_reset`. Se puede conceder permiso a otros usuarios con el privilegio `EXECUTE`.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_reset`.

```
SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';
```

```
 tup_inserted | tup_deleted
--------------+-------------
          896 |           0
(1 row)

SELECT rds_aurora.limitless_stat_reset();

limitless_stat_reset
---------------------
(1 row)

SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

tup_inserted | tup_deleted
-------------+-------------
           0 |           0
(1 row)
```

limitless_stat_statements_reset

La función `limitless_stat_statements_reset` descarta las estadísticas recopiladas hasta el momento por `limitless_stat_statements` correspondientes a los parámetros `username`, `dbname`, `distributed_query_id` y `queryid` especificados. Si no se especifica alguno de los parámetros, se utiliza el valor predeterminado `""` o `0` (no válido) para cada uno de ellos y se restablecen las estadísticas que coincidan con otros parámetros. Si no se especifica ningún parámetro, o si todos los parámetros especificados son `""` o `0` (no son válidos), la función descarta todas las estadísticas. Si se descartan todas las estadísticas de la vista `limitless_stat_statements`, la función también restablece las estadísticas de la vista `limitless_stat_statements_info`.

Los parámetros de entrada son los siguientes:

- `username` (nombre): es el usuario que ha consultado la instrucción.
- `dbname` (nombre): es la base de datos en la que se ha ejecutado la consulta.
- `distributed_query_id` (bigint): es el ID de la consulta principal del nodo coordinador. Esta columna es `NULL` si se trata de la consulta principal. El nodo coordinador envía el ID de consulta distribuida a los nodos participantes. Por lo tanto, para los nodos participantes, los valores del ID de consulta distribuida y del ID de consulta son diferentes.
- `queryid` (bigint): es el ID de consulta de la instrucción.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_statements_reset` para restablecer todas las estadísticas recopiladas por `limitless_stat_statements`.

```
SELECT rds_aurora.limitless_stat_statements_reset();
```

limitless_stat_system_waits

La función `limitless_stat_system_waits` devuelve una vista consolidada de los datos de los eventos de espera de `aurora_stat_system_waits`, que informa de la actividad de espera en todo el sistema en una instancia, desde todos los nodos. Las columnas `subcluster_id` y `subcluster_type` muestran de qué nodo provienen los datos.

El siguiente ejemplo muestra cómo utilizar la función `limitless_stat_system_waits`.

```
postgres_limitless=> SELECT *
FROM rds_aurora.limitless_stat_system_waits() lssw, pg_catalog.aurora_stat_wait_event()
 aswe
WHERE lssw.event_id=aswe.event_id and aswe.event_name='LimitlessTaskScheduler';

 subcluster_id | subcluster_type | type_id | event_id  | waits  |  wait_time   |
 event_name
---------------+-----------------+---------+-----------+--------+--------------
+------------------------
             1 | router          |      12 | 201326607 | 677068 | 616942216307 |
 LimitlessTaskScheduler
             2 | router          |      12 | 201326607 | 678586 | 616939897111 |
 LimitlessTaskScheduler
             3 | shard           |      12 | 201326607 | 756640 | 616965545172 |
 LimitlessTaskScheduler
             4 | shard           |      12 | 201326607 | 755184 | 616958057620 |
 LimitlessTaskScheduler
             5 | shard           |      12 | 201326607 | 757522 | 616963183539 |
 LimitlessTaskScheduler
(5 rows)
```

limitless_terminate_session

La función `limitless_terminate_session` es similar a `pg_terminate_backend`, pero intenta finalizar todos los procesos de backend relacionados con el ID de sesión distribuido proporcionado mediante el envío de una `SIGTERM` (señal de interrupción).

El parámetro de entrada es el siguiente:

- `distributed_session_id` (texto): es el ID de la sesión distribuida que se va a finalizar.

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `pid` (texto): es el ID del proceso de backend.
- `success` (booleano): indica si el proceso ha finalizado correctamente.

El siguiente ejemplo muestra cómo utilizar la función `limitless_terminate_session`.

```
SELECT * FROM rds_aurora.limitless_terminate_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_wait_report

La función `limitless_wait_report` devuelve la actividad del evento de espera durante un período de tiempo desde todos los nodos. Las columnas `subcluster_id` y `subcluster_type` muestran de qué nodo provienen los datos.

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.

El resto de las columnas son las mismas que en `aurora_wait_report`.

El siguiente ejemplo muestra cómo utilizar la función `limitless_wait_report`.

```
postgres_limitless=> select * from rds_aurora.limitless_wait_report();

 subcluster_id | subcluster_type | type_name | event_name | waits | wait_time |
 ms_per_wait | waits_per_xact | ms_per_xact
---------------+-----------------+-----------+------------+-------+-----------
+-------------+--------------- +-------------
             1 | router          | Client    | ClientRead |    57 | 741550.14 |
 13009.652 |           0.19 |    2505.237
             5 | shard           | Client    | ClientRead |    54 | 738897.68 |
 13683.290 |           0.18 |    2496.276
```

```
            4 | shard           | Client    | ClientRead |    54 | 738859.53 |
13682.584 |           0.18 |    2496.147
            2 | router          | Client    | ClientRead |    53 | 719223.64 |
13570.257 |           0.18 |    2429.810
            3 | shard           | Client    | ClientRead |    54 | 461720.40 |
8550.378 |           0.18 |    1559.86
```

Vistas de Base de datos ilimitada de Aurora PostgreSQL

En la tabla siguiente se muestran las nuevas vistas de Base de datos ilimitada de Aurora PostgreSQL.

> **Note**
>
> Las vistas enumeradas en esta tabla se encuentran en el esquema `rds_aurora`. Cuando utilice una vista de Base de datos ilimitada, asegúrese de incluir el nombre del objeto totalmente cualificado: `rds_aurora.`*`object_name`*.

Vista de Base de datos ilimitada de Aurora PostgreSQL	Vista de Aurora PostgreSQL correspondiente
limitless_database	pg_database
limitless_locks	pg_locks
limitless_stat_activity	pg_stat_activity
limitless_stat_all_indexes	pg_stat_all_indexes
limitless_stat_all_tables	pg_stat_all_tables
limitless_stat_database	pg_stat_database
limitless_stat_progress_vacuum	pg_stat_progress_vacuum
limitless_stat_statements	pg_stat_statements
limitless_stat_subclusters	Ninguno
limitless_stat_statements_info	pg_stat_statements_info
limitless_statio_all_indexes	pg_statio_all_indexes
limitless_statio_all_tables	pg_statio_all_tables
limitless_tables	pg_tables

Vista de Base de datos ilimitada de Aurora PostgreSQL	Vista de Aurora PostgreSQL correspondiente
limitless_table_collocations	Ninguno
limitless_table_collocation_distributions	Ninguno

Los siguientes ejemplos proporcionan detalles sobre las funciones de Base de datos ilimitada de Aurora PostgreSQL. Para obtener más información acerca de las vistas de PostgreSQL, consulte Viewing statistics en la documentación de PostgreSQL.

> **Note**
>
> Algunas vistas de estadística pueden arrojar resultados incoherentes si hay transacciones en curso.

limitless_database

Esta vista contiene información sobre las bases de datos disponibles en el grupo de particiones de base de datos. Por ejemplo:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, oid, datname, datacl
 FROM rds_aurora.limitless_database;

 subcluster_id | subcluster_type |  oid  |      datname       |
                                  datacl

---------------+-----------------+-------+--------------------
+-------------------------------------------------------------------------------------------
 2             | router          |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 2             | router          |     5 | postgres           |
 2             | router          | 16384 | rdsadmin           |
 {rdsadmin=CTc/rdsadmin,rds_aurora_limitless_metadata_admin=c/
rdsadmin,rds_aurora_limitless_heat_mgmt_admin=c/rdsadmin}
 2             | router          | 16477 | postgres_limitless |
 2             | router          |     1 | template1          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
```

```
 6             | shard           |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
```

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster (nodo)
- `subcluster_type` (texto): es el tipo de subclúster (nodo), enrutador o partición

El resto de las columnas son las mismas que en `pg_database`.

limitless_locks

Esta vista contiene una fila por proceso y nodo. Proporciona acceso a la información sobre los bloqueos que mantienen los procesos activos en el servidor de la base de datos.

Example de creación de un bloqueo con dos transacciones

En este ejemplo, ejecutamos dos transacciones simultáneamente en dos enrutadores.

```
# Transaction 1 (run on router 1)
BEGIN;
SET search_path = public;
SELECT * FROM customers;
INSERT INTO customers VALUES (400,'foo','bar');

# Transaction 2 (run on router 2)
BEGIN;
SET search_path = public;
ALTER TABLE customers ADD COLUMN phone VARCHAR;
```

Se ejecuta la primera transacción. Las transacciones posteriores deben esperar hasta que se complete la primera transacción. Por lo tanto, la segunda transacción se bloquea con un candado. Para comprobar la causa raíz, ejecutamos un comando uniendo `limitless_locks` con `limitless_stat_activity`.

```
# Run on router 2
SELECT distributed_session_id, state, usename, query, query_start
FROM rds_aurora.limitless_stat_activity
WHERE distributed_session_id in (
SELECT distributed_session_id
FROM rds_aurora.limitless_locks
WHERE relname = 'customers'
);
```

```
 distributed_session_id | state               | usename                | query
                                  | query_start
-----------------------+---------------------+-------------------------
+-------------------------------- -------------+------------------------------
 47BDE66E9A5E8477       | idle in transaction | limitless_metadata_admin | INSERT INTO
 customers VALUES (400,'foo','bar'); | 2023-04-13 17:44:45.152244+00
 2AD7F370202D0FA9       | active              | limitless_metadata_admin | ALTER TABLE
 customers ADD COLUMN phone VARCHAR; | 2023-04-13 17:44:55.113388+00
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
(6 rows)
```

Example de creación de un bloqueo de forma explícita

En este ejemplo, creamos un bloqueo de forma explícita y, a continuación, utilizamos la vista `limitless_locks` para ver los bloqueos (se omiten algunas columnas).

```
BEGIN;
SET search_path = public;
LOCK TABLE customers IN ACCESS SHARE MODE;
SELECT * FROM rds_aurora.limitless_locks WHERE relname = 'customers';

 subcluster_id | subcluster_type | distributed_session_id | locktype |      datname
   | relnspname |  relname  | virtualtransaction |  pid  |      mode
---------------+-----------------+------------------------+----------
+-------------------+------------+ ----------+--------------------+-------
+-----------------
             1 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600787          | 59564 |
 AccessShareLock
             2 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600405          | 67130 |
 AccessShareLock
             3 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 15/473401          | 27735 |
 AccessShareLock
```

```
             4 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473524          | 27734 |
 AccessShareLock
             5 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/472935          | 27737 |
 AccessShareLock
             6 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473015          | 48660 |
 AccessShareLock
(6 rows)
```

limitless_stat_activity

Esta vista contiene una fila por proceso y nodo. Se puede utilizar para realizar un seguimiento del estado general del sistema y de los procesos de clasificación que están tardando mucho tiempo. Por ejemplo:

```
postgres=# SELECT
    subcluster_id,
    subcluster_type,
    distributed_session_id,
    distributed_session_state,
    datname,
    distributed_query_id,
    is_sso_query
FROM
    rds_aurora.limitless_stat_activity
WHERE
    distributed_session_id in ('D2470C97E3D07E06', '5A3CD7B8E5FD13FF')
    order by  distributed_session_id;

 subcluster_id | subcluster_type | distributed_session_id | distributed_session_state |
      datname       | distributed_query_id | is_sso_query
---------------+-----------------+------------------------+---------------------------
+--------------------+----------------------+-------------
 2             | router          | 5A3CD7B8E5FD13FF       | coordinator               |
 postgres_limitless |                      | f
 3             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 4             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 2             | router          | D2470C97E3D07E06       | coordinator               |
 postgres_limitless |                      | t
```

```
 3             | shard           | D2470C97E3D07E06       | participant               |
 postgres_limitless |  4058400544464210222 |
(5 rows)
```

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `distributed_session_id` (texto): es el ID de la sesión distribuida a la que pertenece este proceso.
- `distributed_session_state` (texto): indica si se trata de un proceso coordinador, participante o independiente o no distribuido (se muestra como `NULL`).
- `datname` (texto): es la base de datos a la que está conectado este proceso.
- `distributed_query_id` (bigint): es el ID de la consulta principal del nodo coordinador. Esta columna es `NULL` si se trata de la consulta principal. El nodo coordinador envía el ID de consulta distribuida a los nodos participantes. Por lo tanto, para los nodos participantes, los valores del ID de consulta distribuida y del ID de consulta son diferentes.
- `is_sso_query` (texto): nos permite saber si la consulta está optimizada para una sola partición o no lo está.

El resto de las columnas son las mismas que en `pg_stat_activity`.

limitless_stat_all_indexes

Esta vista contiene estadísticas de uso de los índices del grupo de particiones de base de datos. Por ejemplo:

```
postgres_limitless=> SELECT schemaname, relname, indexrelname, idx_scan
  FROM rds_aurora.limitless_stat_all_indexes
  WHERE relname LIKE 'orders_ts%' ORDER BY indexrelname LIMIT 10;

 schemaname |    relname     |    indexrelname     | idx_scan
------------+----------------+---------------------+----------
 ec_sample  | orders_ts00001 | orders_ts00001_pkey |   196801
 ec_sample  | orders_ts00002 | orders_ts00002_pkey |   196703
 ec_sample  | orders_ts00003 | orders_ts00003_pkey |   196376
 ec_sample  | orders_ts00004 | orders_ts00004_pkey |   197966
 ec_sample  | orders_ts00005 | orders_ts00005_pkey |   195301
```

```
 ec_sample  | orders_ts00006 | orders_ts00006_pkey |   195673
 ec_sample  | orders_ts00007 | orders_ts00007_pkey |   194475
 ec_sample  | orders_ts00008 | orders_ts00008_pkey |   191694
 ec_sample  | orders_ts00009 | orders_ts00009_pkey |   193744
 ec_sample  | orders_ts00010 | orders_ts00010_pkey |   195421
(10 rows)
```

limitless_stat_all_tables

Esta vista contiene estadísticas sobre todas las tablas de la base de datos actual del grupo de particiones de base de datos. Esto resulta útil para realizar un seguimiento de las operaciones de vaciado y las operaciones del lenguaje de manipulación de datos (DML). Por ejemplo:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, relname,
 n_ins_since_vacuum, n_tup_ins, last_vacuum
  FROM rds_aurora.limitless_stat_all_tables
  WHERE relname LIKE 'orders_ts%' ORDER BY relname LIMIT 10;

 subcluster_id | subcluster_type |    relname     | n_ins_since_vacuum | n_tup_ins |
 last_vacuum
---------------+-----------------+----------------+--------------------+-----------
+-------------
 5             | shard           | orders_ts00001 |              34779 |    196083 |
 5             | shard           | orders_ts00002 |              34632 |    194721 |
 5             | shard           | orders_ts00003 |              34950 |    195965 |
 5             | shard           | orders_ts00004 |              34745 |    197283 |
 5             | shard           | orders_ts00005 |              34879 |    195754 |
 5             | shard           | orders_ts00006 |              34340 |    194605 |
 5             | shard           | orders_ts00007 |              33779 |    192203 |
 5             | shard           | orders_ts00008 |              33826 |    191293 |
 5             | shard           | orders_ts00009 |              34660 |    194117 |
 5             | shard           | orders_ts00010 |              34569 |    195560 |
(10 rows)
```

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `relname` (nombre): es el nombre de la tabla.

El resto de las columnas son las mismas que en `pg_stat_all_tables`.

limitless_stat_database

Esta vista contiene estadísticas sobre todas las bases de datos del grupo de particiones de base de datos. Devuelve una fila por base de datos y por nodo. Por ejemplo:

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    datname,
    blks_read,
    blks_hit
FROM
    rds_aurora.limitless_stat_database
WHERE
    datname='postgres_limitless';
 subcluster_id | subcluster_type |      datname       | blks_read | blks_hit
---------------+-----------------+--------------------+-----------+----------
             1 | router          | postgres_limitless |       484 | 34371314
             2 | router          | postgres_limitless |       673 | 33859317
             3 | shard           | postgres_limitless |      1299 | 17749550
             4 | shard           | postgres_limitless |      1094 | 17492849
             5 | shard           | postgres_limitless |      1036 | 17485098
             6 | shard           | postgres_limitless |      1040 | 17437257
(6 rows)
```

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `datname` (nombre): es el nombre de la base de datos.

El resto de las columnas son las mismas que en `pg_stat_database`.

limitless_stat_progress_vacuum

Esta vista contiene información sobre las operaciones de vaciado en curso. Por ejemplo:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;

-[ RECORD 1 ]----------+------------------
subcluster_id          | 3
subcluster_type        | shard
```

```
distributed_session_id | A56D96E2A5C9F426
pid                    | 5270
datname                | postgres
nspname                | public
relname                | customer_ts2
phase                  | vacuuming heap
heap_blks_total        | 130500
heap_blks_scanned      | 100036
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 11184810
num_dead_tuples        | 0

-[ RECORD 2 ]----------+------------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | 56DF26A89EC23AB5
pid                    | 6854
datname                | postgres
nspname                | public
relname                | sales_ts1
phase                  | vacuuming heap
heap_blks_total        | 43058
heap_blks_scanned      | 24868
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 8569523
num_dead_tuples        | 0
```

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `distributed_session_id` (texto): es el identificador de la sesión que ha iniciado la operación de vaciado.
- `datname` (nombre): es la base de datos en la que se vacía.
- `nspname` (nombre): es el nombre del esquema de la tabla que se está vaciando. Es `null` si la tabla que se va a vaciar no está en la misma base de datos a la que está conectado el usuario.
- `relname` (nombre): es el nombre de la tabla que se está vaciando. Es `null` si la tabla que se va a vaciar no está en la misma base de datos a la que está conectado el usuario.

El resto de las columnas son las mismas que en `pg_stat_progress_vacuum`.

limitless_stat_statements

Esta vista proporciona un medio para realizar un seguimiento de las estadísticas de planificación y ejecución de todas las instrucciones SQL que se ejecutan en todos los nodos.

> **Note**
>
> Debe instalar la extensión pg_stat_statements para usar la vista `limitless_stat_statements`.
>
> ```
> -- CREATE EXTENSION must be run by a superuser
> CREATE EXTENSION pg_stat_statements;
>
> -- Verify that the extension is created on all nodes in the DB shard group
> SELECT distinct node_id
>     FROM rds_aurora.limitless_stat_statements
>     LIMIT 10;
> ```

En el siguiente ejemplo se muestra el uso de la vista `limitless_stat_statements`.

```
postgres_limitless=> SELECT
 subcluster_id,
 subcluster_type,
 distributedqueryid,
 username,
 dbname,
 sso_calls
FROM
 rds_aurora.limitless_stat_statements;

 subcluster_id | subcluster_type |  distributedqueryid  |             username
       |       dbname       | sso_calls
---------------+-----------------+----------------------
+-----------------------------------+--------------------+-----------
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
```

```
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         1
 3             | shard           | -7975178695405682176 | postgres
       | postgres_limitless |
[...]
```

Los parámetros de salida son los siguientes:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.
- `subcluster_type` (texto): es el tipo de subclúster al que pertenece este proceso: `router` o `shard`.
- `distributedqueryid` (bigint): es el ID de la consulta principal del nodo coordinador. Esta columna es NULL si se trata de la consulta principal. El nodo coordinador envía el ID de consulta distribuida a los nodos participantes. Por lo tanto, para los nodos participantes, los valores del ID de consulta distribuida y del ID de consulta son diferentes.
- `username` (nombre): es el usuario que ha consultado la instrucción.
- `dbname` (nombre): es la base de datos en la que se ha ejecutado la consulta.
- `sso_calls` (nombre): el número de veces que la instrucción se ha optimizado para una sola partición.

El resto de las columnas son las mismas que en pg_stat_statements.

limitless_stat_statements_info

Esta vista contiene las estadísticas de la vista `limitless_stat_statements`. Cada fila contiene datos de la vista pg_stat_statements_info de cada nodo. La columna `subcluster_id` identifica cada nodo.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_statements_info;

 subcluster_id | subcluster_type | dealloc |          stats_reset
---------------+-----------------+---------+-------------------------------
             1 | router          |       0 | 2023-06-30 21:22:09.524781+00
             2 | router          |       0 | 2023-06-30 21:21:40.834111+00
             3 | shard           |       0 | 2023-06-30 21:22:10.709942+00
```

```
             4 | shard           |       0 | 2023-06-30 21:22:10.740179+00
             5 | shard           |       0 | 2023-06-30 21:22:10.774282+00
             6 | shard           |       0 | 2023-06-30 21:22:10.808267+00
(6 rows)
```

El parámetro de salida es el siguiente:

- `subcluster_id` (texto): es el ID del subclúster al que pertenece este proceso.

El resto de las columnas son las mismas que en pg_stat_statements_info.

limitless_stat_subclusters

Esta vista contiene estadísticas de red entre enrutadores y otros nodos. Contiene una fila por par de enrutador y otro nodo, por ejemplo:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_subclusters;

 orig_subcluster | orig_instance_az | dest_subcluster | dest_instance_az |
 latency_us |       latest_collection       | failed_requests | received_bytes |
 sent_bytes | same_az_requests | cross_az_requests |     stat_reset_timestamp
----------------+------------------+-----------------+------------------
+------------+-------------------------------+-----------------+---------------
+------------+------------------+-------------------+-------------------------------
 3               | us-west-2b       | 2               | us-west-2a       |
 847 | 2024-10-07 17:25:39.518617+00 |               0 |       35668633 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.239675+00
 3               | us-west-2b       | 4               | us-west-2b       |
 419 | 2024-10-07 17:25:39.546376+00 |               0 |      101190464 |  248795719
 |           883478 |                0 | 2024-10-05 12:39:55.231218+00
 3               | us-west-2b       | 5               | us-west-2c       |
    1396 | 2024-10-07 17:25:39.52122+00  |               0 |       72864849 |
 172086292 |                0 |           557726 | 2024-10-05 12:39:55.196412+00
 3               | us-west-2b       | 6               | us-west-2c       |
 729 | 2024-10-07 17:25:39.54828+00  |               0 |       35668584 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.247334+00
 3               | us-west-2b       | 7               | us-west-2a       |
    1702 | 2024-10-07 17:25:39.545307+00 |               0 |       71699576 |
 171634844 |                0 |           556278 | 2024-10-05 12:39:52.715168+00
 2               | us-west-2a       | 3               | us-west-2b       |
 868 | 2024-10-07 17:25:40.293927+00 |               0 |       35659611 |   92011872
 |                0 |           302817 | 2024-10-05 12:39:54.420758+00
 2               | us-west-2a       | 4               | us-west-2b       |
 786 | 2024-10-07 17:25:40.296863+00 |               0 |      102437253 |  251838024
 |                0 |           895060 | 2024-10-05 12:39:54.404081+00
```

```
2               | us-west-2a       | 5               | us-west-2c       |
   1232 | 2024-10-07 17:25:40.292021+00 |               0 |       71990027 |
168828110 |                0 |            545453 | 2024-10-05 12:39:36.769549+00
```

Los parámetros de salida son los siguientes:

- `orig_subcluster` (texto): es el ID del enrutador donde se originan las comunicaciones
- `orig_subcluster_az` (texto): es la zona de disponibilidad (AZ) del enrutador originador
- `dest_subcluster` (texto): es el ID del nodo de destino
- `dest_subcluster_az` (texto): es la última AZ recopilada del nodo de destino
- `latency_us` (bigint): es la última latencia de red recopilada entre nodos, en microsegundos. El valor es `0` si no se puede acceder al nodo.
- `latest_collection` (marca de tiempo): es la marca de tiempo de la última recopilación de AZ y latencia del nodo de destino
- `failed_requests` (bigint): es el recuento acumulado de solicitudes internas fallidas
- `received_bytes` (bigint): es el número acumulado estimado de bytes recibidos desde este nodo
- `sent_bytes` (bigint): es el número acumulado estimado de bytes enviados desde este nodo
- `same_az_requests` (bigint): es el número acumulado de solicitudes de base de datos internas a este nodo cuando se encuentra en la misma AZ que el enrutador de origen
- `cross_az_requests` (bigint): es el número acumulado de solicitudes de base de datos internas a este nodo cuando se encuentra en una AZ distinta que el enrutador de origen
- `stat_reset_timestamp` (marca de tiempo): es la marca de tiempo en la que se restablecieron por última vez las estadísticas acumuladas de esta vista

limitless_statio_all_indexes

Esta vista contiene estadísticas de entrada/salida (E/S) para todos los índices del grupo de particiones de base de datos. Por ejemplo:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_statio_all_indexes WHERE
 relname like'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      |
 indexrelname             | idx_blks_read | idx_blks_hit
---------------+-----------------+------------+-------------------
+------------------------------------+ --------------+--------------
```

```
             3 | shard           | public     | customers_ts00002 |
 customers_ts00002_customer_name_idx |             1 |            0
             3 | shard           | public     | customers_ts00001 |
 customers_ts00001_customer_name_idx |             1 |            0
             4 | shard           | public     | customers_ts00003 |
 customers_ts00003_customer_name_idx |             1 |            0
             4 | shard           | public     | customers_ts00004 |
 customers_ts00004_customer_name_idx |             1 |            0
             5 | shard           | public     | customers_ts00005 |
 customers_ts00005_customer_name_idx |             1 |            0
             5 | shard           | public     | customers_ts00006 |
 customers_ts00006_customer_name_idx |             1 |            0
             6 | shard           | public     | customers_ts00007 |
 customers_ts00007_customer_name_idx |             1 |            0
             6 | shard           | public     | customers_ts00008 |
 customers_ts00008_customer_name_idx |             1 |            0
(8 rows)
```

limitless_statio_all_tables

Esta vista contiene estadísticas de entrada/salida (E/S) para todas las tablas del grupo de particiones de base de datos. Por ejemplo:

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    schemaname,
    relname,
    heap_blks_read,
    heap_blks_hit
FROM
    rds_aurora.limitless_statio_all_tables
WHERE
    relname LIKE 'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      | heap_blks_read |
 heap_blks_hit
---------------+-----------------+------------+-------------------+----------------
+---------------
             3 | shard           | public     | customers_ts00002 |            305 |
      57780
             3 | shard           | public     | customers_ts00001 |            300 |
      56972
```

```
             4 | shard           | public     | customers_ts00004 |            302 |
      57291
             4 | shard           | public     | customers_ts00003 |            302 |
      57178
             5 | shard           | public     | customers_ts00006 |            300 |
      56932
             5 | shard           | public     | customers_ts00005 |            302 |
      57386
             6 | shard           | public     | customers_ts00008 |            300 |
      56881
             6 | shard           | public     | customers_ts00007 |            304 |
      57635
(8 rows)
```

limitless_tables

Esta vista contiene información sobre las tablas en Base de datos ilimitada de Aurora PostgreSQL.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | placeholder | active       | placeholder |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     | HASH
 (order_id)
         2 |     18579 | public      | customer    | active       | sharded     | HASH
 (cust_id)
(4 rows)
```

limitless_table_collocations

Esta vista contiene información sobre las tablas particionadas colocadas.

En el ejemplo siguiente, las tablas `orders` y `customers` están colocadas y las tablas `users` y `followers` están colocadas. Las tablas colocadas tienen el mismo `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;
```

```
 collocation_id | schema_name | table_name
----------------+-------------+------------
              2 | public      | orders
              2 | public      | customers
              5 | public      | users
              5 | public      | followers
(4 rows)
```

limitless_table_collocation_distributions

Esta vista muestra la distribución de claves de cada colocación.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
              2 |             6 | -9223372036854775808 | -4611686018427387904
              2 |             5 | -4611686018427387904 |                    0
              2 |             4 |                    0 |  4611686018427387904
              2 |             3 |  4611686018427387904 |  9223372036854775807
              5 |             6 | -9223372036854775808 | -4611686018427387904
              5 |             5 | -4611686018427387904 |                    0
              5 |             4 |                    0 |  4611686018427387904
              5 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Eventos de espera de Base de datos ilimitada de Aurora PostgreSQL

Un evento de espera en Aurora PostgreSQL indica que una sesión está esperando un recurso, como la entrada/salida (E/S) y los bloqueos. Los eventos de espera son útiles para averiguar por qué las sesiones están esperando recursos e identificar los cuellos de botella. Para obtener más información, consulte Eventos de espera de Aurora PostgreSQL.

Base de datos ilimitada de Aurora PostgreSQL tiene sus propios eventos de espera relacionados con enrutadores y particiones. Muchos de ellos son para enrutadores que esperan las particiones para completar tareas. Los eventos de espera de las particiones contienen detalles sobre las tareas que se están realizando.

Temas

- Consulta de eventos de espera
- Eventos de espera de Base de datos ilimitada

Consulta de eventos de espera

Puede utilizar la vista limitless_stat_activity para consultar eventos de espera, tal como se muestra en el ejemplo siguiente.

```
SELECT wait_event FROM rds_aurora.limitless_stat_activity WHERE
 wait_event_type='AuroraLimitless';

      wait_event
----------------------
 RemoteStatementSetup
 RemoteStatementSetup
(2 rows)
```

También puede usar la función `aurora_stat_system_waits` para mostrar el número de esperas y el tiempo total dedicado a cada evento de espera, como se muestra en el siguiente ejemplo.

```
postgres_limitless=> SELECT type_name,event_name,waits,wait_time
    FROM aurora_stat_system_waits()
    NATURAL JOIN aurora_stat_wait_event()
    NATURAL JOIN aurora_stat_wait_type()
    WHERE type_name='AuroraLimitless'
    ORDER BY wait_time DESC;
```

```
    type_name    |       event_name          |  waits  |  wait_time
-----------------+---------------------------+---------+-------------
 AuroraLimitless | RemoteStatementSetup      |    7518 | 75236507897
 AuroraLimitless | RemoteStatementExecution  |      40 |      132986
 AuroraLimitless | Connect                   |       5 |        1453
(3 rows)
```

Eventos de espera de Base de datos ilimitada

Los siguientes eventos de espera afectan a Base de datos ilimitada de Aurora PostgreSQL. Puede supervisar estos eventos de espera para identificar los cuellos de botella en el procesamiento de Base de datos ilimitada de Aurora PostgreSQL.

Temas

- Evento de espera IO:TwophaseFilePoolWrite
- Evento de espera IO:TwophaseFilePoolRead
- Evento de espera AuroraLimitless:Connect
- Evento de espera AuroraLimitless:AsyncConnect
- Evento de espera AuroraLimitless:RemoteStatementSetup
- Evento de espera AuroraLimitless:RemoteDDLExecution
- Evento de espera AuroraLimitless:RemoteStatementExecution
- Evento de espera AuroraLimitless:FetchRemoteResults
- Evento de espera AuroraLimitless:AsyncGetInitialResponse
- Evento de espera AuroraLimitless:AsyncGetNextResponse
- Evento de espera AuroraLimitless:AbortedCommandCleanup
- Evento de espera AuroraLimitless:DistributedCommitPrepare
- Evento de espera AuroraLimitless:DistributedCommit
- Evento de espera AuroraLimitless:DistributedCommitPrepareThrottle
- Evento de espera AuroraLimitless:PreparedTransactionResolution
- Evento de espera AuroraLimitless:SendPreparedTransactionOutcome
- Evento de espera AuroraLimitless:CommitClockBarrier
- Evento de espera AuroraLimitless:SnapshotClockBarrier

- Evento de espera AuroraLimitless:ReaderSnapshotClockBarrier
- Evento de espera AuroraLimitless:GatherDistributedDeadlockGraph
- Evento de espera AuroraLimitless:DistributedDeadlockDetection
- Evento de espera AuroraLimitless:DistributedDeadlockAbort
- Evento de espera AuroraLimitless:GatherRemoteStats
- Evento de espera AuroraLimitless:GlobalSequenceRefresh
- Evento de espera AuroraLimitless:GlobalVacuumTimeExchange
- Evento de espera AuroraLimitless:DistributedTransactionMonitorGather
- Evento de espera AuroraLimitlessActivity:AdminTaskSchedulerMain
- Evento de espera AuroraLimitlessActivity:AdminTaskExecutorMain
- Evento de espera AuroraLimitlessActivity:AdminTaskMonitorMain
- Evento de espera AuroraLimitlessActivity:DatabaseCleanupMonitorMain
- Evento de espera AuroraLimitlessActivity:TopologyCleanupMonitorMain
- Evento de espera AuroraLimitlessActivity:ToplogyChangeMonitorMain
- Evento de espera AuroraLimitlessActivity:DistributedTransactionMonitorMain
- Evento de espera AuroraLimitlessActivity:GlobalVacuumMonitorMain

Evento de espera IO:TwophaseFilePoolWrite

A la espera de una escritura de un archivo de estado de dos fases dentro del grupo de archivos de estado de dos fases. Este es un evento específico de Aurora.

Causas

Los procesos que ejecutan un comando `PREPARED TRANSACTION`, incluidos los participantes en una transacción distribuida de Base de datos ilimitada, deben conservar el estado de la transacción en un archivo de dos fases. Aurora utiliza un grupo de archivos para mejorar el rendimiento de esta operación.

Acción

Se trata de una operación de E/S de escritura sincrónica y, por lo tanto, una latencia alta en este evento tiene causas similares a `IO:XactSync` y se puede investigar de la misma manera. Si utiliza

Base de datos ilimitada, es posible que tenga que reducir el número de transacciones distribuidas que se ejecutan.

Evento de espera IO:TwophaseFilePoolRead

A la espera de una lectura de un archivo de estado de dos fases dentro del grupo de archivos de estado de dos fases.

Causas

Es posible que los procesos que ejecutan un comando `COMMIT PREPARED` en una transacción preparada previamente, incluidos los participantes en una transacción distribuida de Base de datos ilimitada, tengan que leer el estado de la transacción mantenido previamente en un archivo de dos fases. Aurora utiliza un grupo de archivos para mejorar el rendimiento de esta operación.

Acción

Esta es una operación de E/S de lectura. Por lo tanto, una latencia alta en este evento tiene causas similares a `IO:DataFileRead` y se puede investigar de la misma manera. Si utiliza Base de datos ilimitada, es posible que tenga que reducir el número de transacciones distribuidas que se ejecutan.

Evento de espera AuroraLimitless:Connect

El proceso espera a que se establezca una conexión con otro nodo del clúster.

Causas

Se establecen conexiones entre los procesos y los nodos remotos para ejecutar consultas, distribuir transacciones y ejecutar DDL.

Acción

Reduzca la cantidad de conexiones simultáneas al clúster o ajuste el uso de consultas entre particiones.

Evento de espera AuroraLimitless:AsyncConnect

Este evento es similar a `Connect`, pero representa un proceso que espera a que se establezcan conexiones paralelas a un conjunto de nodos.

Causas

El establecimiento de una conexión paralela se realiza normalmente al ejecutar sentencias DDL.

Acción

Reduzca el número de sentencias DDL o combine varios DDL en la misma sesión para mejorar la reutilización de las conexiones.

Evento de espera AuroraLimitless:RemoteStatementSetup

El proceso está esperando a que se configure la ejecución remota de consultas, como abrir o cerrar el cursor o crear una instrucción preparada.

Causas

Este evento de espera aumenta con el número de escaneos en tablas particionadas en las que la instrucción no se ha podido optimizar en una sola partición.

Acción

Optimice las consultas para reducir el número de operaciones de escaneo o aumentar la elegibilidad para la optimización de una sola partición.

Evento de espera AuroraLimitless:RemoteDDLExecution

El proceso está esperando a que se complete un comando DDL remoto.

Causas

Al ejecutar un comando DDL en un grupo de particiones de base de datos, debe distribuirse a otros nodos de enrutadores y particiones antes de confirmar la operación. Algunas operaciones de DDL pueden ejecutarse durante mucho tiempo, ya que los datos deben adaptarse a los cambios de esquema.

Acción

Identifique los comandos DDL de ejecución prolongada para poder optimizarlos.

Evento de espera AuroraLimitless:RemoteStatementExecution

Un proceso está esperando a que se complete un comando remoto.

Causas

Se está ejecutando un comando SQL en un nodo remoto. Este evento aparecerá con frecuencia en las comunicaciones internas, por ejemplo, en `auto_analyze` y en las comprobaciones de latidos del corazón.

Acción

Identifique los comandos de ejecución prolongada con la vista limitless_stat_statements. En muchos casos, esto es algo esperado, especialmente para los trabajadores en segundo plano o los procesos internos. No es necesario realizar ninguna acción.

Evento de espera AuroraLimitless:FetchRemoteResults

Un proceso está esperando para recuperar filas de un nodo remoto.

Causas

Este evento de espera puede aumentar al obtener un gran número de filas de una tabla remota, como una tabla particionada o de referencia.

Acción

Identifique las consultas `SELECT` no optimizadas con la vista `limitless_stat_statements`. Optimice las consultas para recuperar solo los datos necesarios. También puede adaptar el parámetro `rds_aurora.limitless_maximum_adaptive_fetch_size`.

Evento de espera AuroraLimitless:AsyncGetInitialResponse

El proceso espera una respuesta inicial cuando se utiliza el modo de canalización en la ejecución de la consulta.

Causas

Esto suele ocurrir durante la ejecución de enrutador a partición en el caso de consultas con una ubicación de datos de una sola partición. Todo esto forma parte de la ejecución normal.

Acción

No hay que hacer nada más.

Evento de espera AuroraLimitless:AsyncGetNextResponse

El proceso espera una respuesta adicional cuando se utiliza el modo de canalización en la ejecución de la consulta.

Causas

Esto suele ocurrir durante la ejecución de enrutador a partición en el caso de consultas con una ubicación de datos de una sola partición. Todo esto forma parte de la ejecución normal.

Acción

No hay que hacer nada más.

Evento de espera AuroraLimitless:AbortedCommandCleanup

El proceso está esperando el resultado de una consulta de limpieza remota. Las consultas de limpieza se envían a los nodos de partición para devolverlos al estado adecuado cuando finaliza una transacción distribuida.

Causas

La limpieza de transacciones se realiza cuando una transacción se cancela porque se ha detectado un error o porque un usuario ha emitido un comando ABORT explícito o ha cancelado la consulta en ejecución.

Acción

Investigue la causa de la cancelación de la transacción.

Evento de espera AuroraLimitless:DistributedCommitPrepare

El proceso confirma una transacción distribuida y espera a que todos los participantes confirmen el comando prepare.

Causas

Las transacciones que modifican varios nodos deben realizar una confirmación distribuida. Una espera prolongada en `DistributedCommitPrepare` podría deberse a las largas esperas del evento `IO:TwophaseFilePoolWrite` en los nodos participantes.

Acción

Reduzca la cantidad de transacciones que modifican los datos en varios nodos. Investigue los eventos `IO:TwophaseFilePoolWrite` en otros nodos del clúster.

Evento de espera AuroraLimitless:DistributedCommit

El proceso confirma una transacción distribuida y espera a que todos los participantes principales confirmen el comando commit.

Causas

Las transacciones que modifican varios nodos deben realizar una confirmación distribuida. Una espera prolongada en `DistributedCommit` podría deberse a las largas esperas del evento `IO:XactSync` en el participante principal.

Acción

Reduzca la cantidad de transacciones que modifican los datos en varios nodos. Investigue los eventos `IO:XactSync` en otros nodos del clúster.

Evento de espera AuroraLimitless:DistributedCommitPrepareThrottle

El proceso intenta preparar una transacción distribuida y tiene limitaciones debido a las transacciones preparadas existentes.

Causas

Las transacciones que modifican varios nodos deben realizar una confirmación distribuida. Los participantes en estas transacciones deben realizar una operación de preparación como parte del protocolo de confirmación. Aurora limita el número de preparaciones simultáneas y, si se supera este límite, el proceso esperará en el evento `DistributedCommitPrepareThrottle`.

Acción

Reduzca la cantidad de transacciones que modifican los datos en varios nodos. Investigue los eventos `IO:TwophaseFilePoolWrite`, ya que el aumento del tiempo transcurrido en esos eventos podría provocar la acumulación de transacciones preparadas existentes y, por lo tanto, limitar los nuevos intentos de preparación.

Evento de espera AuroraLimitless:PreparedTransactionResolution

El proceso ha detectado una tupla modificada por una transacción distribuida que tiene el estado preparado. El proceso debe determinar si la transacción distribuida será visible en su instantánea.

Causas

Las transacciones que modifican varios nodos deben realizar una confirmación distribuida que incluye una fase de preparación. Un número elevado de transacciones distribuidas o un aumento de la latencia en las confirmaciones distribuidas pueden provocar que otros procesos se topen con el evento de espera `PreparedTransactionResolution`.

Acción

Reduzca la cantidad de transacciones que modifican los datos en varios nodos. Investigue los eventos relacionados con las confirmaciones distribuidas, ya que si se prolonga el tiempo transcurrido en esos eventos, podría aumentar la latencia en la ruta de confirmación de las transacciones distribuidas. Se recomienda que investigue las cargas de red y de la CPU.

Evento de espera AuroraLimitless:SendPreparedTransactionOutcome

El proceso se ejecuta en un nodo que coordina una transacción distribuida y otro proceso ha preguntado por el estado de esa transacción, o bien el proceso ha confirmado una transacción distribuida y está enviando el resultado a los participantes.

Causas

Los procesos en los que se produzca el evento de espera `PreparedTransactionResolution` consultarán al coordinador de transacciones. La respuesta del coordinador de transacciones será SendPreparedTransactionOutcome.

Acción

Reduzca la cantidad de transacciones que modifican los datos en varios nodos. Investigue los eventos relacionados con las confirmaciones distribuidas, además de los eventos `IO:TwophaseFilePoolWrite` y `IO:TwophaseFilePoolRead`, pues esos eventos podrían aumentar la latencia en la ruta de confirmación de las transacciones distribuidas. Se recomienda que investigue las cargas de red y de la CPU.

Evento de espera AuroraLimitless:CommitClockBarrier

El proceso está confirmando una transacción y debe esperar para garantizar que el tiempo de confirmación asignado ya haya pasado en todos los nodos del clúster.

Causas

La saturación de la CPU o de la red podría provocar un aumento de la desviación del reloj, lo que provocaría que se perdiera tiempo en este evento de espera.

Acción

Investigue la saturación de la CPU o de la red en su clúster.

Evento de espera AuroraLimitless:SnapshotClockBarrier

El proceso ha recibido un tiempo de la instantánea de otro nodo con un reloj que marca una hora futura y está esperando a que su propio reloj alcance esa hora.

Causas

Por lo general, esto ocurre después de que el proceso haya recibido los resultados de una función que se ha colocado en una partición y cuando el reloj se desvía entre los nodos. La saturación de la CPU o de la red podría provocar un aumento de la desviación del reloj, lo que provocaría que se perdiera tiempo en este evento de espera.

Acción

Investigue la saturación de la CPU o de la red en su clúster.

Evento de espera AuroraLimitless:ReaderSnapshotClockBarrier

Este evento se produce en los nodos de lectura. El proceso espera a que el nodo de lectura reproduzca el flujo de escritura para que se apliquen todas las escrituras que se realizaron antes de la instantánea del proceso.

Causas

Un aumento del retraso en la réplica de Aurora puede provocar un aumento del tiempo de espera en este evento.

Acción

Investigue el retraso de la réplica de Aurora.

Evento de espera AuroraLimitless:GatherDistributedDeadlockGraph

El proceso consiste en comunicarse con otros nodos para recopilar gráficos de bloqueo como parte de la detección de interbloqueos distribuidos.

Causas

Cuando un proceso está esperando un bloqueo, realizará una comprobación de interbloqueo distribuido después de esperar más de `rds_aurora.limitless_distributed_deadlock_timeout`.

Acción

Investigue las causas de la contención de bloqueos en su aplicación y considere la posibilidad de ajustar `rds_aurora.limitless_distributed_deadlock_timeout`.

Evento de espera AuroraLimitless:DistributedDeadlockDetection

El proceso consiste en comunicarse con otros nodos para detectar un interbloqueo distribuido.

Causas

Cuando un proceso está esperando un bloqueo, realizará una comprobación de interbloqueo distribuido después de esperar más de `rds_aurora.limitless_distributed_deadlock_timeout`.

Acción

Investigue las causas de la contención de bloqueos en su aplicación y considere la posibilidad de ajustar `rds_aurora.limitless_distributed_deadlock_timeout`.

Evento de espera AuroraLimitless:DistributedDeadlockAbort

El proceso consiste en comunicarse con otro nodo para abortar una sesión elegida como víctima de un interbloqueo distribuido.

Causas

Los patrones de la aplicación están provocando interbloqueos distribuidos.

Acción

Investigue los patrones de la aplicación que están provocando interbloqueos distribuidos.

Evento de espera AuroraLimitless:GatherRemoteStats

El proceso consiste en recopilar estadísticas de otros nodos del clúster.

Causas

La supervisión de las consultas y vistas de actividad, como `limitless_stat_activity`, recuperará estadísticas de otros nodos.

Acción

No hay que hacer nada más.

Evento de espera AuroraLimitless:GlobalSequenceRefresh

El proceso está generando un nuevo valor de secuencia y debe solicitar un nuevo fragmento de la secuencia global.

Causas

Una alta tasa de generación de valores de secuencia puede causar retrasos en este evento si `rds_aurora.limitless_sequence_chunk_size` no es suficiente.

Acción

Esto es normal. Si ve un tiempo excesivo en este evento, considere la posibilidad de adaptar `rds_aurora.limitless_sequence_chunk_size`. Consulte la documentación sobre secuencias en Base de datos ilimitada.

Evento de espera AuroraLimitless:GlobalVacuumTimeExchange

El proceso consiste en intercambiar datos de instantáneas para respaldar el vaciado.

Causas

Los nodos de Base de datos ilimitada intercambian los datos de tiempo de las instantáneas activas más antiguas con otros nodos para calcular el tiempo límite correcto para la ejecución de vaciado.

Acción

No hay que hacer nada más.

Evento de espera AuroraLimitless:DistributedTransactionMonitorGather

El proceso consiste en recopilar metadatos de transacciones de otros nodos para facilitar la limpieza de transacciones distribuidas.

Causas

Los nodos de Base de datos ilimitada intercambian los metadatos de las transacciones con otros nodos para determinar cuándo se puede purgar el estado de las transacciones distribuidas.

Acción

No hay que hacer nada más.

Evento de espera AuroraLimitlessActivity:AdminTaskSchedulerMain

Esperando en el bucle principal del proceso del programador de tareas.

Evento de espera AuroraLimitlessActivity:AdminTaskExecutorMain

Esperando en el bucle principal del proceso del ejecutor de tareas.

Evento de espera AuroraLimitlessActivity:AdminTaskMonitorMain

Esperando en el bucle principal del proceso de supervisión de tareas.

Evento de espera AuroraLimitlessActivity:DatabaseCleanupMonitorMain

Esperando en el bucle principal del proceso de supervisión de limpieza de la base de datos.

Evento de espera AuroraLimitlessActivity:TopologyCleanupMonitorMain

Esperando en el bucle principal del proceso de supervisión de limpieza de la topología.

Evento de espera AuroraLimitlessActivity:ToplogyChangeMonitorMain

Esperando en el bucle principal del proceso de supervisión de cambio de la topología.

Evento de espera AuroraLimitlessActivity:DistributedTransactionMonitorMain

Esperando en el ciclo principal del proceso de supervisión de transacciones distribuidas.

Evento de espera AuroraLimitlessActivity:GlobalVacuumMonitorMain

Esperando en el bucle principal del proceso de supervisión de vaciado.

Copia de seguridad y restauración de Base de datos ilimitada de Aurora PostgreSQL

Puede realizar una copia de seguridad y restaurar un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL.

Contenido

- Copia de seguridad de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL.
  - Creación de una instantánea de clúster de base de datos
- Restauración de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL
  - Restauración de clúster de base de datos a partir de una instantánea de base de datos
  - Restauración de un clúster de base de datos con la recuperación en un momento dado
- No se admiten las utilidades de copia de seguridad y restauración de PostgreSQL

Copia de seguridad de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL.

La copia de seguridad de un clúster de base de datos con Base de datos ilimitada de Aurora PostgreSQL presenta similitudes y diferencias en cuanto a la funcionalidad en comparación con la copia de seguridad de un clúster de base de datos de Aurora estándar.

- Al realizar una instantánea manual de un clúster de base de datos de Aurora que utiliza Base de datos ilimitada, la instantánea incluye datos del grupo de particiones de base de datos.
- Las copias de seguridad continuas incluyen datos del grupo de particiones de base de datos.
- Las instantáneas diarias automatizadas incluyen datos del grupo de particiones de base de datos.
- Puede copiar instantáneas de clúster de base de datos. Para obtener más información, consulte Copia de una instantánea de clúster de base de datos.
- Puede compartir instantáneas de clúster de base de datos. Para obtener más información, consulte Compartir una instantánea de clúster de base de datos.
- No puede utilizar las utilidades `pg_dump` o `pg_dumpall` para hacer copias de seguridad de las bases de datos del grupo de particiones de base de datos.

- Base de datos ilimitada de Aurora PostgreSQL permite realizar instantáneas definitivas al eliminar clústeres de bases de datos.
- Base de datos ilimitada de Aurora PostgreSQL no admite conservar copias de seguridad automáticas al eliminar clústeres de bases de datos.

Creación de una instantánea de clúster de base de datos

Cree una instantánea de un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL del mismo modo que para un clúster de base de datos de Aurora estándar, tal como se muestra en el siguiente ejemplo de AWS CLI:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier my-db-cluster \
    --db-cluster-snapshot-identifier my-db-cluster-snapshot
```

Para obtener información acerca de las copias de seguridad del clúster de base de datos, consulte Información general de copias de seguridad y restauración de un clúster de base de datos Aurora.

Restauración de un clúster de base de datos que utiliza Base de datos ilimitada de Aurora PostgreSQL

La restauración de un clúster de base de datos con Base de datos ilimitada de Aurora PostgreSQL presenta similitudes y diferencias en cuanto a la funcionalidad en comparación con la restauración de un clúster de base de datos de Aurora estándar.

- Puede restaurar un clúster de base de datos de Base de datos ilimitada únicamente desde un clúster de base de datos de origen que utilice una versión de motor de base de datos compatible con la base de datos ilimitada, como `16.4-limitless`.
- Cuando se restaura un clúster de base de datos desde una instantánea manual de un clúster de base de datos que usa Base de datos ilimitada, se restaura todo el almacenamiento del clúster de base de datos. Esto incluye el almacenamiento del grupo de particiones de base de datos.

  Debe crear un grupo de particiones de base de datos para acceder al almacenamiento de su Base de datos ilimitada.
- Puede restaurar un clúster de base de datos a un momento dado dentro del período de retención con la recuperación en un momento dado (PITR). El clúster de base de datos restaurado incluye el almacenamiento del grupo de particiones de base de datos.

  Debe crear un grupo de particiones de base de datos para acceder al almacenamiento de su Base de datos ilimitada.
- Los clústers de base datos de Base de datos ilimitada de Aurora PostgreSQL eliminados no admiten la PITR.
- Cuando se restaura un clúster de base de datos desde una instantánea diaria automática, también se restaura el almacenamiento del grupo de particiones de base de datos.
- Al restaurar un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL, debe habilitar Monitorización mejorada e Información de rendimiento. Asegúrese de incluir el ID de la clave de KMS de Información de rendimiento.

Tras restaurar un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL, asegúrese de comprobar su funcionalidad ejecutando consultas.

Restauración de clúster de base de datos a partir de una instantánea de base de datos

Los siguientes ejemplos de la AWS CLI muestran cómo restaurar un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL a partir de una instantánea de clúster de base de datos.

Debe usar la versión del motor de base de datos `16.4-limitless`.

Restauración de un clúster de base de datos de Base de datos ilimitada desde una instantánea de clúster de base de datos

1. Restaure el clúster de base de datos:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-db-cluster \
    --snapshot-identifier my-db-cluster-snapshot \
    --engine aurora-postgresql \
    --engine-version 16.4-limitless \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Cree el grupo de particiones de base de datos:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Para obtener más información, consulte Adición de un grupo de particiones de base de datos a un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL existente.

Para obtener más información acerca de cómo restaurar clústers de base de datos de Aurora a partir de instantáneas de clúster de base de datos, consulte Restauración de una instantánea de clúster de base de datos.

Restauración de un clúster de base de datos con la recuperación en un momento dado

Los siguientes ejemplos de la AWS CLI muestran cómo restaurar un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL mediante la recuperación en un momento dado (PITR).

Restauración de un clúster de base de datos de Base de datos ilimitada mediante la PITR

1. Restaure el clúster de base de datos:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-db-cluster \
    --db-cluster-identifier my-new-db-cluster \
    --use-latest-restorable-time \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Cree el grupo de particiones de base de datos:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Para obtener más información, consulte Adición de un grupo de particiones de base de datos a un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL existente.

Para obtener más información acerca de PITR, consulte Restauración de un clúster de base de dato a un momento indicado.

No se admiten las utilidades de copia de seguridad y restauración de PostgreSQL

Las siguientes utilidades de PostgreSQL no son compatibles ni con el clúster de base de datos principal ni con el grupo de particiones de base de datos:

- `pg_dump`
- `pg_dumpall`
- `pg_restore`

Si bien es posible que pueda utilizarlas con archivos binarios de código abierto o con métodos alternativos, si lo hace, podría arrojar resultados incoherentes.

Actualización de Base de datos ilimitada de Amazon Aurora PostgreSQL

Lo siguiente afecta a la actualización de Base de datos ilimitada de Aurora PostgreSQL:

- Se admiten actualizaciones de versiones secundarias.
- Se admite la aplicación de parches a Base de datos ilimitada de Aurora PostgreSQL. Los parches aparecen como pendientes de mantenimiento y se deben aplicar durante el período de mantenimiento.

Actualización de los clústeres de base de datos que utiliza Base de datos ilimitada de Amazon Aurora PostgreSQL.

Para actualizar un clúster de base de datos, debe modificarlo y elegir una nueva versión del motor de base de datos. Puede utilizar la Consola de administración de AWS o la AWS CLI.

Consola

Actualización de su clúster de base de datos de Base de datos ilimitada

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Acceda a la página Databases (Bases de datos).
3. Seleccione el clúster de base de datos de Base de datos ilimitada que desea actualizar.
4. Elija Modificar.

   Aparece la página Modificar el clúster de base de datos.
5. Para Versión del motor de base de datos, elija una versión de motor de base de datos nueva, por ejemplo, Aurora PostgreSQL con base de datos ilimitada (compatible con PostgreSQL 16.6).
6. Elija Continuar.
7. En la página de resumen, seleccione si desea aplicar los cambios inmediatamente o durante el siguiente período de mantenimiento programado y, a continuación, elija Modificar clúster.

CLI

Use el comando modify-db-cluster de la AWS CLI, tal como se muestra en el ejemplo siguiente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-sv2-cluster \
    --engine-version 16.6-limitless \
    --apply-immediately
```

Referencia sobre Base de datos ilimitada de Aurora PostgreSQL

Proporcionamos los siguientes temas de referencia para Base de datos ilimitada de Aurora PostgreSQL.

Temas

- Comandos SQL de lenguaje de definición de datos (DDL) compatibles y no compatibles
- Limitaciones del DDL y otra información para Base de datos ilimitada de Aurora PostgreSQL
- Comandos de lenguaje de manipulación de datos (DDL) y SQL de procesamiento de consultas compatibles y no compatibles
- Limitaciones del DML y otra información para Base de datos ilimitada de Aurora PostgreSQL
- Variables de Base de datos ilimitada de Aurora PostgreSQL
- Parámetros de clúster en Base de datos ilimitada de Aurora PostgreSQL

Comandos SQL de lenguaje de definición de datos (DDL) compatibles y no compatibles

En la siguiente tabla se enumeran los comandos de DDL compatibles y no compatibles con Base de datos ilimitada de Aurora PostgreSQL, además de referencias a las limitaciones y otra información.

Comando	¿Se admite?	Limitaciones o más información
ALTER AGGREGATE	No	No aplicable
ALTER COLLATION	Sí	Ninguno
ALTER CONVERSION	Sí	Ninguno
ALTER DATABASE	No	No aplicable
ALTER DEFAULT PRIVILEGES	No	No aplicable
ALTER DOMAIN	No	No aplicable
ALTER EVENT TRIGGER	No	No aplicable

Comando	¿Se admite?	Limitaciones o más información
ALTER EXTENSION	Sí	Extensiones
ALTER FOREIGN DATA WRAPPER	No	No aplicable
ALTER FOREIGN TABLE	No	No aplicable
ALTER FUNCTION	Sí	Funciones
ALTER GROUP	Sí	Ninguno
ALTER INDEX	Sí	Ninguno
ALTER LANGUAGE	No	No aplicable
ALTER LARGE OBJECT	No	No aplicable
ALTER MATERIALIZED VIEW	No	No aplicable
ALTER OPERATOR	Sí	Ninguno
ALTER OPERATOR CLASS	Sí	Ninguno
ALTER OPERATOR FAMILY	Sí	Ninguno
ALTER POLICY	No	No aplicable
ALTER PROCEDURE	Sí	Ninguno
ALTER PUBLICATION	No	No aplicable
ALTER ROLE	Sí	Ninguno
ALTER ROUTINE	No	No aplicable
ALTER RULE	No	No aplicable
ALTER SCHEMA	Sí	Ninguno

Comando	¿Se admite?	Limitaciones o más información
ALTER SEQUENCE	Sí	Secuencias
ALTER SERVER	No	No aplicable
ALTER STATISTICS	No	No aplicable
ALTER SUBSCRIPTION	No	No aplicable
ALTER SYSTEM	No	No aplicable
ALTER TABLE	Sí	ALTER TABLE
ALTER TABLESPACE	No	No aplicable
ALTER TEXT SEARCH CONFIGURATION	No	No aplicable
ALTER TEXT SEARCH DICTIONARY	No	No aplicable
ALTER TEXT SEARCH PARSER	No	No aplicable
ALTER TEXT SEARCH TEMPLATE	No	No aplicable
ALTER TRIGGER	No	No aplicable
ALTER TYPE	Sí	Ninguno
ALTER USER	Sí	Ninguno
ALTER USER MAPPING	No	No aplicable
ALTER VIEW	Sí	Ninguno
COMMENT	No	No aplicable

Comando	¿Se admite?	Limitaciones o más información
CREATE ACCESS METHOD	No	No aplicable
CREATE AGGREGATE	No	No aplicable
CREATE CAST	Sí	Ninguno
CREATE COLLATION	Sí	Ninguno
CREATE CONVERSION	Sí	Ninguno
CREATE DATABASE	Sí	CREATE DATABASE
CREATE DOMAIN	No	No aplicable
CREATE EVENT TRIGGER	No	No aplicable
CREATE EXTENSION	Sí	Extensiones
CREATE FOREIGN DATA WRAPPER	No	No aplicable
CREATE FOREIGN TABLE	No	No aplicable
CREATE FUNCTION	Sí	Funciones
CREATE GROUP	Sí	Ninguno
CREATE INDEX	Sí	CREATE INDEX
CREATE LANGUAGE	No	No aplicable
CREATE MATERIALIZED VIEW	No	No aplicable
CREATE OPERATOR	Sí	Ninguno
CREATE OPERATOR CLASS	Sí	Ninguno

Comando	¿Se admite?	Limitaciones o más información
CREATE OPERATOR FAMILY	Sí	Ninguno
CREATE POLICY	Sí	Ninguno
CREATE PROCEDURE	Sí	Ninguno
CREATE PUBLICATION	No	No aplicable
CREAR ROL	Sí	Ninguno
CREATE RULE	No	No aplicable
CREATE SCHEMA	Sí	CREATE SCHEMA
CREATE SEQUENCE	Sí	Secuencias
CREATE SERVER	No	No aplicable
CREATE STATISTICS	No	No aplicable
CREATE SUBSCRIPTION	No	No aplicable
CREATE TABLE	Sí	CREATE TABLE
CREATE TABLE AS	Sí	CREATE TABLE AS
CREATE TABLESPACE	No	No aplicable
CREATE TEMPORARY TABLE	No	No aplicable
CREATE TEMPORARY TABLE AS	No	No aplicable
CREATE TEXT SEARCH CONFIGURATION	No	No aplicable

Comando	¿Se admite?	Limitaciones o más información
CREATE TEXT SEARCH DICTIONARY	No	No aplicable
CREATE TEXT SEARCH PARSER	No	No aplicable
CREATE TEXT SEARCH TEMPLATE	No	No aplicable
CREATE TRANSFORM	No	No aplicable
CREATE TRIGGER	No	No aplicable
CREATE TYPE	Sí	Ninguno
CREAR USUARIO	Sí	Ninguno
CREATE USER MAPPING	No	No aplicable
CREATE VIEW	Sí	Ninguno
DROP ACCESS METHOD	No	No aplicable
DROP AGGREGATE	Sí	Ninguno
DROP CAST	Sí	Ninguno
DROP COLLATION	Sí	Ninguno
DROP CONVERSION	Sí	Ninguno
DROP DATABASE	Sí	DROP DATABASE
DROP DOMAIN	No	No aplicable
DROP EVENT TRIGGER	No	No aplicable
DROP EXTENSION	Sí	Extensiones

Comando	¿Se admite?	Limitaciones o más información
DROP FOREIGN DATA WRAPPER	No	No aplicable
DROP FOREIGN TABLE	No	No aplicable
DROP FUNCTION	Sí	Funciones
DROP GROUP	Sí	Ninguno
DROP INDEX	Sí	Ninguno
DROP LANGUAGE	No	No aplicable
DROP MATERIALIZED VIEW	No	No aplicable
DROP OPERATOR	Sí	Ninguno
DROP OPERATOR CLASS	Sí	Ninguno
DROP OPERATOR FAMILY	Sí	Ninguno
DROP OWNED	No	No aplicable
DROP POLICY	No	No aplicable
DROP PROCEDURE	Sí	Ninguno
DROP PUBLICATION	No	No aplicable
DROP ROLE	Sí	Ninguno
DROP ROUTINE	No	No aplicable
DROP RULE	No	No aplicable
DROP SCHEMA	Sí	Ninguno
DROP SEQUENCE	Sí	Ninguno

Comando	¿Se admite?	Limitaciones o más información
DROP SERVER	No	No aplicable
DROP STATISTICS	No	No aplicable
DROP SUBSCRIPTION	No	Ninguno
DROP TABLE	Sí	Ninguno
DROP TABLESPACE	No	No aplicable
DROP TEXT SEARCH CONFIGURATION	No	No aplicable
DROP TEXT SEARCH DICTIONARY	No	No aplicable
DROP TEXT SEARCH PARSER	No	No aplicable
DROP TEXT SEARCH TEMPLATE	No	No aplicable
DROP TRANSFORM	No	No aplicable
DROP TRIGGER	No	No aplicable
DROP TYPE	Sí	Ninguno
DROP USER	Sí	Ninguno
DROP USER MAPPING	No	No aplicable
DROP VIEW	Sí	Ninguno
GRANT	Sí	Ninguno
REASSIGN OWNED	No	No aplicable

Comando	¿Se admite?	Limitaciones o más información
REVOKE	Sí	Ninguno
SECURITY LABEL	No	No aplicable
SELECT INTO	Sí	SELECT INTO
SET	Sí	Ninguno
SET CONSTRAINTS	No	No aplicable
SET ROLE	Sí	Ninguno
SET SESSION AUTHORIZATION	Sí	Ninguno
SET TRANSACTION	Sí	Ninguno
TRUNCATE	Sí	Ninguno

Limitaciones del DDL y otra información para Base de datos ilimitada de Aurora PostgreSQL

En los siguientes temas se describen las limitaciones o se proporciona más información sobre los comandos DDL SQL en Base de datos ilimitada de Aurora PostgreSQL.

Temas

- ALTER TABLE
- CREATE DATABASE
- CREATE INDEX
- CREATE SCHEMA
- CREATE TABLE
- CREATE TABLE AS
- DROP DATABASE
- SELECT INTO
- Restricciones
- Valores predeterminados
- Extensiones
- Claves externas
- Funciones
- Secuencias

ALTER TABLE

El comando `ALTER TABLE` es compatible de forma generalizada con Base de datos ilimitada de Aurora PostgreSQL. Para obtener más información, consulte ALTER ROLE en la documentación de PostgreSQL.

Limitaciones

`ALTER TABLE` tiene las siguientes limitaciones en cuanto a las opciones compatibles.

Eliminación de columnas

- En las tablas particionadas, no se pueden eliminar las columnas que forman parte de la clave de partición.
- En las tablas de referencia, no se pueden eliminar las columnas de clave principal.

Cambio del tipo de datos de una columna

- La expresión `USING` no se admite.
- En las tablas particionadas, no puede eliminar las columnas que forman parte de la clave de partición.

Agregación o eliminación de una restricción

Para obtener más información sobre lo que no se admite, consulte Restricciones.

Cambio del valor predeterminado de una columna

Se admiten los valores predeterminados. Para obtener más información, consulte Valores predeterminados.

Opciones no admitidas

Algunas opciones no son compatibles porque dependen de características no compatibles, como los desencadenadores.

Las siguientes opciones de nivel de tabla para `ALTER TABLE` no son compatibles:

- `ALL IN TABLESPACE`
- `ATTACH PARTITION`
- `DETACH PARTITION`
- Indicador `ONLY`

- `RENAME CONSTRAINT`

Las siguientes opciones de nivel de columna para `ALTER TABLE` no son compatibles:

- ADD GENERATED
- DROP EXPRESSION [IF EXISTS]
- DROP IDENTITY [IF EXISTS]
- RESET
- RESTART
- SET
- SET COMPRESSION
- SET STATISTICS

CREATE DATABASE

En Base de datos ilimitada de Aurora PostgreSQL, solo se admiten bases de datos ilimitadas.

Mientras se ejecuta `CREATE DATABASE`, las bases de datos creadas correctamente en uno o más nodos pueden fallar en otros nodos, ya que la creación de bases de datos no es una operación transaccional. En este caso, los objetos de la base de datos creados correctamente se eliminan automáticamente de todos los nodos dentro de un período de tiempo predeterminado para mantener la coherencia en el grupo de particiones de base de datos. Durante este tiempo, si se vuelve a crear una base de datos con el mismo nombre, se puede producir un error que indique que la base de datos ya existe.

Las siguientes opciones son compatibles:

- Intercalación:

```
CREATE DATABASE name WITH
    [LOCALE = locale]
    [LC_COLLATE = lc_collate]
    [LC_CTYPE = lc_ctype]
    [ICU_LOCALE = icu_locale]
    [ICU_RULES = icu_rules]
    [LOCALE_PROVIDER = locale_provider]
    [COLLATION_VERSION = collation_version];
```

- `CREATE DATABASE WITH OWNER`:

```
CREATE DATABASE name WITH OWNER = user_name;
```

Las siguientes opciones no son compatibles:

- `CREATE DATABASE WITH TABLESPACE`:

```
CREATE DATABASE name WITH TABLESPACE = tablespace_name;
```

- `CREATE DATABASE WITH TEMPLATE`:

```
CREATE DATABASE name WITH TEMPLATE = template;
```

CREATE INDEX

Se admite `CREATE INDEX CONCURRENTLY` en las tablas particionadas:

```
CREATE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

Se admite `CREATE UNIQUE INDEX` para todos los tipos de tabla:

```
CREATE UNIQUE INDEX index_name ON table_name(column_name);
```

No se admite `CREATE UNIQUE INDEX CONCURRENTLY`:

```
CREATE UNIQUE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

Para obtener más información, consulte UNIQUE. Para obtener información general sobre la creación de índices, consulte CREATE INDEX en la documentación de PostgreSQL.

Visualización de índices

> No todos los índices son visibles en los enrutadores cuando se utiliza `\d table_name` o comandos similares. En su lugar, utilice la vista `pg_catalog.pg_indexes` para obtener índices, tal y como se muestra en el siguiente ejemplo.

```
SET rds_aurora.limitless_create_table_mode='sharded';
```

```
SET rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items (id int PRIMARY KEY, val int);
CREATE INDEX items_my_index on items (id, val);

postgres_limitless=> SELECT * FROM pg_catalog.pg_indexes WHERE tablename='items';

 schemaname | tablename |   indexname    | tablespace |
   indexdef
------------+-----------+----------------+------------
+-------------------------------------------------------------------------
 public     | items     | items_my_index |            | CREATE INDEX items_my_index
 ON ONLY public.items USING btree (id, val)
 public     | items     | items_pkey     |            | CREATE UNIQUE INDEX
 items_pkey ON ONLY public.items USING btree (id)
(2 rows)
```

CREATE SCHEMA

No se admite `CREATE SCHEMA` con un elemento de esquema:

```
CREATE SCHEMA my_schema CREATE TABLE (column_name INT);
```

Este comando genera un error similar al siguiente:

```
ERROR: CREATE SCHEMA with schema elements is not supported
```

CREATE TABLE

No se admiten relaciones en las instrucciones `CREATE TABLE`, como:

```
CREATE TABLE orders (orderid int, customerId int, orderDate date) WITH
 (autovacuum_enabled = false);
```

No se admiten las columnas `IDENTITY`, como:

```
CREATE TABLE orders (orderid INT GENERATED ALWAYS AS IDENTITY);
```

La base de datos ilimitada de Aurora PostgreSQL admite hasta 54 caracteres para los nombres de las tablas particionadas.

CREATE TABLE AS

Para crear una tabla con `CREATE TABLE AS`, debe usar la variable `rds_aurora.limitless_create_table_mode`. Para las tablas particionadas, también debe usar la variable `rds_aurora.limitless_create_table_shard_key`. Para obtener más información, consulte Creación de tablas ilimitadas con variables.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

CREATE TABLE ctas_table AS SELECT 1 a;

-- "source" is the source table whose columns and data types are used to create the new
 "ctas_table2" table.
CREATE TABLE ctas_table2 AS SELECT a,b FROM source;
```

No se puede utilizar `CREATE TABLE AS` para crear tablas de referencia, ya que requieren restricciones de clave principal. `CREATE TABLE AS` no propaga las claves principales a las tablas nuevas.

Para obtener más información, consulte CREATE TABLE AS en la documentación de PostgreSQL.

DROP DATABASE

Puede eliminar las bases de datos que haya creado.

El comando `DROP DATABASE` se ejecuta de forma asíncrona en segundo plano. Mientras se está ejecutando, recibirá un error si intenta crear una nueva base de datos con el mismo nombre.

SELECT INTO

`SELECT INTO` es funcionalmente similar a CREATE TABLE AS. Debe usar la variable `rds_aurora.limitless_create_table_mode`. Para las tablas particionadas, también debe usar la variable `rds_aurora.limitless_create_table_shard_key`. Para obtener más información, consulte Creación de tablas ilimitadas con variables.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';
```

```
-- "source" is the source table whose columns and data types are used to create the new
 "destination" table.
SELECT * INTO destination FROM source;
```

Actualmente, la operación `SELECT INTO` se realiza a través del enrutador, no directamente a través de las particiones. Por lo tanto, el rendimiento puede ser lento.

Para obtener más información, consulte SELECT INTO en la documentación de PostgreSQL.

Restricciones

Las siguientes limitaciones se aplican a las restricciones en Base de datos ilimitada de Aurora PostgreSQL.

CHECK

Se admiten restricciones simples que implican operadores de comparación con literales. No se admiten expresiones ni restricciones más complejas que requieren validaciones de funciones, tal como se muestra en los siguientes ejemplos.

```
CREATE TABLE my_table (
    id  INT CHECK (id > 0)                                      -- supported
  , val INT CHECK (val > 0 AND val < 1000)                      -- supported
  , tag TEXT CHECK (length(tag) > 0)                            -- not supported:
 throws "Expression inside CHECK constraint is not supported"
  , op_date TIMESTAMP WITH TIME ZONE CHECK (op_date <= now())  -- not supported:
 throws "Expression inside CHECK constraint is not supported"
);
```

También puede asignar nombres explícitos a las restricciones, tal como se muestra en el siguiente ejemplo.

```
CREATE TABLE my_table (
    id  INT CONSTRAINT positive_id  CHECK (id > 0)
  , val INT CONSTRAINT val_in_range CHECK (val > 0 AND val < 1000)
);
```

Puede utilizar la sintaxis de restricciones de nivel de tabla con la restricción `CHECK`, tal como se muestra en el siguiente ejemplo.

```
CREATE TABLE my_table (
    id INT CONSTRAINT positive_id  CHECK (id > 0)
  , min_val INT CONSTRAINT min_val_in_range CHECK (min_val > 0 AND min_val < 1000)
  , max_val INT
  , CONSTRAINT max_val_in_range CHECK (max_val > 0 AND max_val < 1000 AND max_val >
 min_val)
);
```

EXCLUDE

Las restricciones de exclusión no se admiten en Base de datos ilimitada de Aurora PostgreSQL.

FOREIGN KEY

Para obtener más información, consulte Claves externas.

NOT NULL

Las restricciones `NOT NULL` se admiten sin restricciones.

PRIMARY KEY

La clave principal implica restricciones únicas y, por lo tanto, las mismas restricciones a las restricciones únicas se aplican a la clave principal. Esto significa:

- Que si una tabla se convierte en tabla particionada, la clave de partición debe ser un subconjunto de la clave principal. Es decir, la clave principal contiene todas las columnas de la clave de partición.
- Si una tabla se convierte en una tabla de referencia, debe tener una clave principal.

En los siguientes ejemplos se ilustra el uso de claves principales.

```
-- Create a standard table.
CREATE TABLE public.my_table (
    item_id INT
  , location_code INT
  , val INT
  , comment text
);

-- Change the table to a sharded table using the 'item_id' and 'location_code'
 columns as shard keys.
CALL rds_aurora.limitless_alter_table_type_sharded('public.my_table',
 ARRAY['item_id', 'location_code']);
```

Aquí se intenta añadir una clave principal que no contiene una clave de partición:

```
-- Add column 'item_id' as the primary key.
-- Invalid because the primary key doesnt include all columns from the shard key:
-- 'location_code' is part of the shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR

-- add column "val" as primary key
-- Invalid because primary key does not include all columns from shard key:
--  item_id and location_code iare part of shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR
```

Aquí se intenta añadir una clave principal que contiene una clave de partición:

```
-- Add the 'item_id' and 'location_code' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code); -- OK

-- Add the 'item_id', 'location_code', and 'val' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code, val); -- OK
```

Cambie una tabla estándar por una tabla de referencia.

```
-- Create a standard table.
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);

-- Convert the table to a reference table.
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcode');
```

Para obtener más información sobre la creación de tablas particionadas y de referencia, consulte Creación de tablas de Base de datos ilimitada de Aurora PostgreSQL.

UNIQUE

En las tablas particionadas, la clave única debe contener la clave de partición, es decir, la clave de partición debe ser un subconjunto de la clave única. Esto se comprueba al cambiar el tipo de tabla a particionada. En las tablas de referencia no hay ninguna restricción.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT UNIQUE
);
```

Se admiten restricciones `UNIQUE` en el nivel de tabla, tal como se muestra en el siguiente ejemplo.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT
  , CONSTRAINT zipcode_and_email UNIQUE (zipcode, email)
```

```
);
```

El siguiente ejemplo muestra el uso conjunto de una clave principal y una clave única. Ambas claves deben incluir la clave de partición.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"p_id"}';
CREATE TABLE t1 (
p_id BIGINT NOT NULL,
c_id BIGINT NOT NULL,
PRIMARY KEY (p_id),
UNIQUE (p_id, c_id)
);
```

Para obtener más información, consulte Constraints en la documentación de PostgreSQL.

Valores predeterminados

Base de datos ilimitada de Aurora PostgreSQL admite expresiones con valores predeterminados.

En el siguiente ejemplo se muestra el uso de los valores predeterminados.

```
CREATE TABLE t (
    a INT DEFAULT 5,
    b TEXT DEFAULT 'NAN',
    c NUMERIC
);

CALL rds_aurora.limitless_alter_table_type_sharded('t', ARRAY['a']);
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c
---+-----+---
 5 | NAN |
(1 row)
```

Se admiten expresiones, como las del siguiente ejemplo.

```
CREATE TABLE t1 (a NUMERIC DEFAULT random());
```

En el siguiente ejemplo, se agrega una nueva columna que es NOT NULL y tiene un valor predeterminado.

```
ALTER TABLE t ADD COLUMN d BOOLEAN NOT NULL DEFAULT FALSE;
SELECT * FROM t;

 a |  b  | c | d
---+-----+---+---
 5 | NAN |   | f
(1 row)
```

En el siguiente ejemplo, se modifica una columna existente con un valor predeterminado.

```
ALTER TABLE t ALTER COLUMN c SET DEFAULT 0.0;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
(2 rows)
```

En el siguiente ejemplo, se elimina un valor predeterminado.

```
ALTER TABLE t ALTER COLUMN a DROP DEFAULT;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
   | NAN | 0.0 | f
(3 rows)
```

Para obtener más información, consulte Default values en la documentación de PostgreSQL.

Extensiones

Las siguientes extensiones de PostgreSQL son compatibles con Base de datos ilimitada de Aurora PostgreSQL:

- `aurora_limitless_fdw`: esta extensión viene preinstalada. No se puede eliminar.
- `aws_s3`: esta extensión funciona en Base de datos ilimitada de Aurora PostgreSQL de forma similar a Aurora PostgreSQL.

  Puede importar datos de un bucket de Amazon S3 a un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL o exportar datos desde un clúster de base de datos de Base de datos ilimitada de Aurora PostgreSQL a un bucket de Amazon S3. Para obtener más información, consulte Importación de datos de Amazon S3 en un clúster de base de datos Aurora PostgreSQL y Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3.
- `btree_gin`
- `citext`
- `ip4r`
- `pg_buffercache`: esta extensión se comporta de forma diferente en Base de datos ilimitada de Aurora PostgreSQL y en la comunidad PostgreSQL. Para obtener más información, consulte Diferencias de pg_buffercache en Base de datos ilimitada de Aurora PostgreSQL.
- `pg_stat_statements`
- `pg_trgm`
- `pgcrypto`
- `pgstattuple`: esta extensión se comporta de forma diferente en Base de datos ilimitada de Aurora PostgreSQL y en la comunidad PostgreSQL. Para obtener más información, consulte Diferencias de pgstattuple en Base de datos ilimitada de Aurora PostgreSQL.
- `pgvector`
- `plpgsql`: esta extensión viene preinstalada, pero puede eliminarla.
- `PostGIS`: no se admiten transacciones largas ni funciones de administración de tablas. No se admite la modificación de la tabla de referencia espacial.
- `unaccent`
- `uuid`

La mayoría de extensiones de PostgreSQL no son compatibles con Base de datos ilimitada de Aurora PostgreSQL. Sin embargo, puede seguir utilizando el parámetro de configuración shared_preload_libraries (SPL) para cargar extensiones en el clúster de base de datos principal de Aurora PostgreSQL. También se cargan en Base de datos ilimitada de Aurora PostgreSQL, pero es posible que no funcionen correctamente.

Por ejemplo, puede cargar la extensión `pg_hint_plan`, pero cargarla no garantiza que se utilicen las sugerencias incluidas en los comentarios de las consultas.

> **Note**
>
> No se pueden modificar los objetos asociados a la extensión pg_stat_statements. Para obtener más información sobre la instalación de `pg_stat_statements`, consulte limitless_stat_statements.

Puede utilizar las funciones `pg_available_extensions` y `pg_available_extension_versions` para buscar las extensiones compatibles con Base de datos ilimitada de Aurora PostgreSQL.

Los siguientes DDL admiten las extensiones:

CREATE EXTENSION

Puede crear extensiones, como en PostgreSQL.

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
    [ WITH ] [ SCHEMA schema_name ]
             [ VERSION version ]
             [ CASCADE ]
```

Para obtener más información, consulte CREATE EXTENSION en la documentación de PostgreSQL.

ALTER EXTENSION

Se admiten los siguientes DDL:

```
ALTER EXTENSION name UPDATE [ TO new_version ]
```

```
ALTER EXTENSION name SET SCHEMA new_schema
```

Para obtener más información, consulte ALTER EXTENSION en la documentación de PostgreSQL.

DROP EXTENSION

Puede eliminar extensiones, como en PostgreSQL.

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Para obtener más información, consulte DROP EXTENSION en la documentación de PostgreSQL.

Los siguientes DDL no admiten extensiones:

ALTER EXTENSION

No se pueden agregar ni eliminar objetos miembros de extensiones.

```
ALTER EXTENSION name ADD member_object

ALTER EXTENSION name DROP member_object
```

Diferencias de pg_buffercache en Base de datos ilimitada de Aurora PostgreSQL

En Base de datos ilimitada de Aurora PostgreSQL, al instalar la extensión pg_buffercache y usar la vista `pg_buffercache`, recibirá información relacionada con el búfer únicamente del nodo al que está conectado actualmente, es decir, el enrutador. Del mismo modo, al utilizar la función `pg_buffercache_summary` o `pg_buffercache_usage_counts` se proporciona información únicamente del nodo conectado.

Puede tener varios nodos y es posible que necesite acceder a la información del búfer desde cualquier nodo para diagnosticar los problemas de forma eficaz. Por lo tanto, Base de datos ilimitada ofrece las siguientes funciones:

- `rds_aurora.limitless_pg_buffercache(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_summary(subcluster_id)`

- `rds_aurora.limitless_pg_buffercache_usage_counts(subcluster_id)`

Al introducir el ID de subclúster de cualquier nodo, ya sea un enrutador o una partición, puede acceder fácilmente a la información del búfer específica de ese nodo. Estas funciones están disponibles directamente al instalar la extensión `pg_buffercache` en la base de datos ilimitada.

> **Note**
>
> Base de datos ilimitada de Aurora PostgreSQL admite estas funciones en la versión 1.4 y posteriores de la extensión `pg_buffercache`.

Las columnas que se muestran en la vista `limitless_pg_buffercache` difieren ligeramente de las de la vista `pg_buffercache`:

- `bufferid`: permanece sin cambios por parte de `pg_buffercache`.
- `relname`: en lugar de mostrar el número de nodo del archivo como en `pg_buffercache`, `limitless_pg_buffercache` presenta el `relname` asociado si está disponible en la base de datos actual o en los catálogos del sistema particionado; de lo contrario es `NULL`.
- `parent_relname`: esta nueva columna, que no está presente en `pg_buffercache`, muestra la columna principal `relname` si el valor de la columna `relname` representa una tabla particionada (en el caso de las tablas particionadas). De lo contrario, se muestra `NULL`.
- `spcname`: en lugar de mostrar el identificador del objeto del espacio de tabla (OID) como en `pg_buffercache`, `limitless_pg_buffercache` muestra el nombre del espacio de tabla.
- `datname`: en lugar de mostrar el OID de la base de datos como en `pg_buffercache`, `limitless_pg_buffercache` muestra el nombre de la base de datos.
- `relforknumber`: permanece sin cambios por parte de `pg_buffercache`.
- `relblocknumber`: permanece sin cambios por parte de `pg_buffercache`.
- `isdirty`: permanece sin cambios por parte de `pg_buffercache`.
- `usagecount`: permanece sin cambios por parte de `pg_buffercache`.
- `pinning_backends`: permanece sin cambios por parte de `pg_buffercache`.

Las columnas de las vistas `limitless_pg_buffercache_summary` y `limitless_pg_buffercache_usage_counts` son las mismas que las de las vistas normales `pg_buffercache_summary` y `pg_buffercache_usage_counts`, respectivamente.

Al utilizar estas funciones, puede acceder a información detallada de la memoria caché del búfer en todos los nodos de su entorno de Base de datos ilimitada, lo que facilita un diagnóstico y una administración más eficaces de sus sistemas de bases de datos.

Diferencias de pgstattuple en Base de datos ilimitada de Aurora PostgreSQL

En Aurora PostgreSQL, la extensión pgstattuple actualmente no admite tablas externas, tablas particionadas ni índices particionados. En Base de datos ilimitada de Aurora PostgreSQL, los objetos creados por los usuarios suelen estar entre estos tipos no compatibles. Si bien hay tablas e índices normales (por ejemplo, tablas de catálogos y sus índices), la mayoría de los objetos residen en nodos externos, lo que los convierte en objetos extraños para el enrutador.

Reconocemos la importancia de esta extensión para obtener estadísticas en el nivel de tupla, lo cual es esencial para tareas como eliminar la sobrecarga y recopilar información de diagnóstico. Por lo tanto, Base de datos ilimitada de Aurora PostgreSQL admite la extensión `pgstattuple` en bases de datos ilimitadas.

Base de datos ilimitada de Aurora PostgreSQL incluye las siguientes funciones en el esquema `rds_aurora`:

Funciones estadísticas en el nivel de tupla

`rds_aurora.limitless_pgstattuple(relation_name)`

- Objetivo: extraer estadísticas en el nivel de tupla para tablas estándar y sus índices
- Entrada: `relation_name` (texto) es el nombre de la relación
- Salida: columnas coherentes con las devueltas por la función `pgstattuple` en Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple(relation_name, subcluster_id)`

- Objetivo: extraer estadísticas en el nivel de tupla para las tablas de referencia, las tablas particionadas, las tablas de catálogo y sus índices
- Input:
  - `relation_name` (texto): es el nombre de la relación
  - `subcluster_id` (texto): es el ID del subclúster del nodo donde se van a extraer las estadísticas
- Salida:
  - En el caso de las tablas de referencia y de catálogo (incluidos sus índices), las columnas son coherentes con las de Aurora PostgreSQL.

- En el caso de las tablas particionadas, las estadísticas representan únicamente la partición de la tabla particionada que reside en el subclúster especificado.

Funciones de estadísticas de índices

`rds_aurora.limitless_pgstatindex(`*`relation_name`*`)`

- Objetivo: extraer estadísticas para los índices del árbol B en tablas estándar
- Entrada: `relation_name` (texto) es el nombre del índice del árbol B
- Salida: se devuelven todas las columnas excepto `root_block_no`. Las columnas devueltas son coherentes con la función `pgstatindex` en Aurora PostgreSQL

`rds_aurora.limitless_pgstatindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objetivo: extraer estadísticas para los índices del árbol B en las tablas de referencia, las tablas particionadas y las tablas de catálogo
- Input:
  - `relation_name` (texto): es el nombre del índice del árbol B
  - `subcluster_id` (texto): es el ID del subclúster del nodo donde se van a extraer las estadísticas
- Salida:
  - Para los índices de las tablas de referencia y de catálogo, se devuelven todas las columnas (excepto `root_block_no`). Las columnas devueltas son coherentes con Aurora PostgreSQL.
  - En el caso de las tablas particionadas, las estadísticas representan únicamente la partición del índice de la tabla particionada que reside en el subclúster especificado. La columna `tree_level` muestra el promedio de todos los segmentos de la tabla del subclúster solicitado.

`rds_aurora.limitless_pgstatginindex(`*`relation_name`*`)`

- Objetivo: extraer estadísticas de índices invertidos generalizados (GIN) en tablas estándar
- Entrada: `relation_name` (texto) es el nombre del GIN
- Salida: columnas coherentes con las devueltas por la función `pgstatginindex` en Aurora PostgreSQL

`rds_aurora.limitless_pgstatginindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objetivo: extraer estadísticas para los índices GIN en las tablas de referencia, las tablas particionadas y las tablas de catálogo

- Input:
  - `relation_name` (texto): es el nombre del índice
  - `subcluster_id` (texto): es el ID del subclúster del nodo donde se van a extraer las estadísticas
- Salida:
  - En el caso de los índices GIN de las tablas de referencia y de catálogo, las columnas son coherentes con las de Aurora PostgreSQL.
  - En el caso de las tablas particionadas, las estadísticas representan únicamente la partición del índice de la tabla particionada que reside en el subclúster especificado.

`rds_aurora.limitless_pgstathashindex(relation_name)`

- Objetivo: extraer estadísticas para los índices hash en tablas estándar
- Entrada: `relation_name` (texto) es el nombre del índice hash
- Salida: columnas coherentes con las devueltas por la función `pgstathashindex` en Aurora PostgreSQL

`rds_aurora.limitless_pgstathashindex(relation_name, subcluster_id)`

- Objetivo: extraer estadísticas para los índices hash en las tablas de referencia, las tablas particionadas y las tablas de catálogo
- Input:
  - `relation_name` (texto): es el nombre del índice
  - `subcluster_id` (texto): es el ID del subclúster del nodo donde se van a extraer las estadísticas
- Salida:
  - Para los índices hash de las tablas de referencia y de catálogo, las columnas son coherentes con Aurora PostgreSQL.
  - En el caso de las tablas particionadas, las estadísticas representan únicamente la partición del índice de la tabla particionada que reside en el subclúster especificado.

Funciones de recuento de páginas

`rds_aurora.limitless_pg_relpages(relation_name)`

- Objetivo: extraer el recuento de páginas de las tablas estándar y sus índices
- Entrada: `relation_name` (texto) es el nombre de la relación
- Salida: recuento de páginas de la relación especificada

`rds_aurora.limitless_pg_relpages(relation_name, subcluster_id)`

- Objetivo: extraer el recuento de páginas de las tablas de referencia, las tablas particionadas y las tablas de catálogo (incluidos sus índices)
- Input:
  - `relation_name` (texto): es el nombre de la relación
  - `subcluster_id` (texto): es el ID del subclúster del nodo de donde se va a extraer el recuento de páginas
- Salida: en el caso de las tablas particionadas, el recuento de páginas es la suma de las páginas de todos los segmentos de la tabla del subclúster especificado.

Funciones estadísticas aproximadas en el nivel de tupla

`rds_aurora.limitless_pgstattuple_approx(relation_name)`

- Objetivo: extraer estadísticas aproximadas en el nivel de tupla para las tablas estándar y sus índices
- Entrada: `relation_name` (texto) es el nombre de la relación
- Salida: columnas coherentes con las devueltas por la función pgstattuple_approx en Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple_approx(relation_name, subcluster_id)`

- Objetivo: extraer estadísticas aproximadas en el nivel de tupla para las tablas de referencia, las tablas particionadas y las tablas de catálogo (con sus índices)
- Input:
  - `relation_name` (texto): es el nombre de la relación
  - `subcluster_id` (texto): es el ID del subclúster del nodo donde se van a extraer las estadísticas
- Salida:
  - En el caso de las tablas de referencia y de catálogo (incluidos sus índices), las columnas son coherentes con las de Aurora PostgreSQL.
  - En el caso de las tablas particionadas, las estadísticas representan únicamente la partición de la tabla particionada que reside en el subclúster especificado.

> **Note**
>
> Actualmente, Base de datos ilimitada de Aurora PostgreSQL no admite la extensión `pgstattuple` en vistas materializadas, tablas TOAST o tablas temporales.
> En Base de datos ilimitada de Aurora PostgreSQL, debe proporcionar la entrada como texto, aunque Aurora PostgreSQL admite otros formatos.

Claves externas

Se admiten las restricciones de clave externa (`FOREIGN KEY`) con algunas limitaciones:

- Se admite `CREATE TABLE` con `FOREIGN KEY` solo en tablas estándar. Para crear una tabla particionada o de referencia con `FOREIGN KEY`, primero debe crear la tabla sin una restricción de clave externa. A continuación, modifíquela con la siguiente instrucción:

  ```
  ALTER TABLE ADD CONSTRAINT;
  ```

- No se admite la conversión de una tabla estándar en una tabla particionada o de referencia cuando la tabla tiene una restricción de clave externa. Elimine la restricción y agréguela después de la conversión.
- Se aplican las siguientes limitaciones a los tipos de tabla para restricciones de clave externa:
  - Una tabla estándar puede tener una restricción de clave externa a otra tabla estándar.
  - Una tabla particionada puede tener una restricción de clave externa si las tablas principal y secundaria están colocadas y la clave externa es un superconjunto de la clave de partición.
  - Una tabla particionada puede tener una restricción de clave externa a una tabla de referencia.
  - Una tabla particionada puede tener una restricción de clave externa a otra tabla de referencia.

Temas

- Opciones de clave externa
- Ejemplos

Opciones de clave externa

Algunas opciones de DDL admiten claves externas en Base de datos ilimitada de Aurora PostgreSQL. En la siguiente tabla se enumeran las opciones compatibles y no compatibles entre las tablas de Base de datos ilimitada de Aurora PostgreSQL.

Opción de DDL	De referencia a de referencia	De particionada a particionada (colocada)	De particionada a de referencia	De estándar a estándar
`DEFERRABLE`	Sí	Sí	Sí	Sí

Opción de DDL	De referencia a de referencia	De particionada a particionada (colocada)	De particionada a de referencia	De estándar a estándar
`INITIALLY DEFERRED`	Sí	Sí	Sí	Sí
`INITIALLY IMMEDIATE`	Sí	Sí	Sí	Sí
`MATCH FULL`	Sí	Sí	Sí	Sí
`MATCH PARTIAL`	No	No	No	No
`MATCH SIMPLE`	Sí	Sí	Sí	Sí
`NOT DEFERRABLE`	Sí	Sí	Sí	Sí
`NOT VALID`	Sí	No	No	Sí
`ON DELETE CASCADE`	Sí	Sí	Sí	Sí
`ON DELETE NO ACTION`	Sí	Sí	Sí	Sí
`ON DELETE RESTRICT`	Sí	Sí	Sí	Sí
`ON DELETE SET DEFAULT`	No	No	No	No
`ON DELETE SET NULL`	Sí	No	No	Sí
`ON UPDATE CASCADE`	No	No	No	Sí

Opción de DDL	De referencia a de referencia	De particionada a particionada (colocada)	De particionada a de referencia	De estándar a estándar
`ON UPDATE NO ACTION`	Sí	Sí	Sí	Sí
`ON UPDATE RESTRICT`	Sí	Sí	Sí	Sí
`ON UPDATE SET DEFAULT`	No	No	No	No
`ON UPDATE SET NULL`	Sí	No	No	Sí

Ejemplos

- De estándar a estándar:

```
set rds_aurora.limitless_create_table_mode='standard';

CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products (product_no),
    quantity integer
);

SELECT constraint_name, table_name, constraint_type
FROM information_schema.table_constraints WHERE constraint_type='FOREIGN KEY';

 constraint_name         | table_name  | constraint_type
-------------------------+-------------+-----------------
 orders_product_no_fkey  | orders      | FOREIGN KEY
```

```
(1 row)
```

- De particionada a particionada (colocada):

```
set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"product_no"}';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
set rds_aurora.limitless_create_table_collocate_with='products';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- De particionada a de referencia:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- De referencia a de referencia:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

Funciones

Base de datos ilimitada de Aurora PostgreSQL admite funciones.

Se admiten los siguientes DDL para las funciones:

CREATE FUNCTION

Puede crear funciones, como en Aurora PostgreSQL, menos cambiar su volatilidad al sustituirlas.

Para obtener más información, consulte CREATE FUNCTION en la documentación de PostgreSQL.

ALTER FUNCTION

Puede modificar funciones, como en Aurora PostgreSQL, menos modificar su volatilidad.

Para obtener más información, consulte ALTER FUNCTION en la documentación de PostgreSQL.

DROP FUNCTION

Puede eliminar funciones, como en Aurora PostgreSQL.

```
DROP FUNCTION [ IF EXISTS ] name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ]
 [, ...]
    [ CASCADE | RESTRICT ]
```

Para obtener más información, consulte DROP FUNCTION en la documentación de PostgreSQL.

Temas

- Distribución de funciones
- Volatilidad de las funciones

Distribución de funciones

Cuando todas las instrucciones de una función están dirigidas a una única partición, resulta beneficioso colocar toda la función en la partición de destino. Luego, el resultado se propaga de nuevo al enrutador, en lugar de desentrañar la función en el propio enrutador. La función de compresión de funciones y procedimientos almacenados resulta útil para los clientes que desean

ejecutar su función o procedimiento almacenado más cerca del origen de datos, es decir, de la partición.

Para distribuir una función, primero debe crear la función y, a continuación, llamar al procedimiento `rds_aurora.limitless_distribute_function` para distribuirla. Esta función usa la siguiente sintaxis:

```
SELECT rds_aurora.limitless_distribute_function('function_prototype',
 ARRAY['shard_key'], 'collocating_table');
```

La función toma los siguientes parámetros:

- *`function_prototype`*: es la función que se va a distribuir. Mencione solo los argumentos de entrada y ningún argumento de salida.

  Si alguno de los argumentos está definido como parámetro `OUT`, no incluya su tipo en los argumentos de `function_prototype`.
- `ARRAY['`*`shard_key`*`']`: es la lista de argumentos de la función que se identifica como la clave de partición de la función.
- *`collocating_table`*: es la tabla particionada que contiene el rango de datos de la partición de destino.

Para identificar la partición en la que se debe presionar esta función para ejecutarla, el sistema toma el argumento `ARRAY['`*`shard_key`*`']`, lo convierte en un hash y busca la partición de *`collocating_table`* que aloja el rango que contiene este valor de hash.

Restricciones

Al distribuir una función o un procedimiento, solo se ocupa de los datos que están limitados por el rango de claves de partición de dicha partición. En los casos en que la función o el procedimiento intenten acceder a los datos de una partición diferente, los resultados devueltos por la función o el procedimiento distribuidos serán diferentes de los que no estén distribuidos.

Por ejemplo, se crea una función que contiene consultas que tocarán varias particiones, pero después se llama al procedimiento `rds_aurora.limitless_distribute_function` para distribuirla. Al invocar esta función proporcionando argumentos para una clave de partición, es probable que los resultados de su ejecución estén limitados por los valores presentes en esa partición. Estos resultados son distintos de los que se obtienen sin distribuir la función.

Ejemplos

Considere la función `func` en la que tenemos la tabla particionada `customers` con la clave de partición `customer_id`.

```
postgres_limitless=> CREATE OR REPLACE FUNCTION func(c_id integer, sc integer)
 RETURNS int
    language SQL
    volatile
    AS $$
    UPDATE customers SET score = sc WHERE customer_id = c_id RETURNING score;
    $$;
```

Ahora distribuimos esta función:

```
SELECT rds_aurora.limitless_distribute_function('func(integer, integer)',
 ARRAY['c_id'], 'customers');
```

A continuación se muestra un ejemplo de planes de consulta.

```
EXPLAIN(costs false, verbose true) SELECT func(27+1,10);

                    QUERY PLAN
 ---------------------------------------------------
  Foreign Scan
    Output: (func((27 + 1), 10))
    Remote SQL:  SELECT func((27 + 1), 10) AS func
  Single Shard Optimized
 (4 rows)
```

```
EXPLAIN(costs false, verbose true)
 SELECT * FROM customers,func(customer_id, score) WHERE customer_id=10 AND score=27;

                          QUERY PLAN
 ------------------------------------------------------------------------
  Foreign Scan
    Output: customer_id, name, score, func
    Remote SQL:  SELECT customers.customer_id,
      customers.name,
      customers.score,
      func.func
```

```
     FROM public.customers,
      LATERAL func(customers.customer_id, customers.score) func(func)
    WHERE ((customers.customer_id = 10) AND (customers.score = 27))
  Single Shard Optimized
 (10 rows)
```

El siguiente ejemplo muestra un procedimiento con los parámetros `IN` y `OUT` como argumentos.

```
CREATE OR REPLACE FUNCTION get_data(OUT id INTEGER, IN arg_id INT)
    AS $$
    BEGIN
        SELECT customer_id,
        INTO id
        FROM customer
        WHERE customer_id = arg_id;
    END;
    $$ LANGUAGE plpgsql;
```

El siguiente ejemplo distribuye el procedimiento utilizando únicamente parámetros `IN`.

```
EXPLAIN(costs false, verbose true) SELECT * FROM get_data(1);

             QUERY PLAN
 -----------------------------------
  Foreign Scan
    Output: id
    Remote SQL:  SELECT customer_id
     FROM get_data(1) get_data(id)
  Single Shard Optimized
 (6 rows)
```

Volatilidad de las funciones

Para determinar si una función es inmutable, estable o volátil, puede comprobar el valor `provolatile` en la vista pg_proc. El valor `provolatile` indica si el resultado de la función depende únicamente de sus argumentos de entrada o si se ve afectado por factores externos.

Puede ser uno de los siguientes:

- `i`: funciones inmutables, que siempre ofrecen el mismo resultado para las mismas entradas
- `s`: funciones estables, cuyos resultados (para entradas fijas) no cambian en un escaneo

- v: funciones volátiles, cuyos resultados pueden cambiar en cualquier momento. También puede utilizar v para funciones con efectos secundarios, de manera que las llamadas que se realicen no se puedan optimizar.

Los siguientes ejemplos muestran funciones volátiles.

```
SELECT proname, provolatile FROM pg_proc WHERE proname='pg_sleep';

 proname  | provolatile
----------+-------------
 pg_sleep | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='uuid_generate_v4';

     proname      | provolatile
------------------+-------------
 uuid_generate_v4 | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='nextval';

 proname | provolatile
---------+-------------
 nextval | v
(1 row)
```

Base de datos ilimitada de Aurora PostgreSQL no admite cambiar la volatilidad de una función existente. Esto se aplica a ambos comandos `ALTER FUNCTION` y `CREATE OR REPLACE FUNCTION`, tal como se muestra en los siguientes ejemplos.

```
-- Create an immutable function
CREATE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$IMMUTABLE;

-- Altering the volatility throws an error
ALTER FUNCTION immutable_func1 STABLE;
```

```
-- Replacing the function with altered volatility throws an error
CREATE OR REPLACE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$VOLATILE;
```

Se recomienda asignar las volatilidades correctas a las funciones. Por ejemplo, si la función utiliza `SELECT` de varias tablas o hace referencia a objetos de bases de datos, no la defina como `IMMUTABLE`. Si el contenido de la tabla cambia alguna vez, se rompe la inmutabilidad.

Aurora PostgreSQL permite `SELECT` dentro de funciones inmutables, pero los resultados pueden ser incorrectos. Base de datos ilimitada de Aurora PostgreSQL puede devolver errores y resultados incorrectos. Para obtener más información, consulte Function volatility categories en la documentación de PostgreSQL.

Secuencias

Las secuencias con nombre son objetos de base de datos que generan números únicos en orden ascendente o descendente. `CREATE SEQUENCE` crea un nuevo generador de números de secuencia. Se garantiza que los valores de secuencia son únicos.

Al crear una secuencia con nombre en Base de datos ilimitada de Aurora PostgreSQL, se crea un objeto de secuencia distribuida. Luego, Base de datos ilimitada de Aurora PostgreSQL distribuye fragmentos de valores de secuencia que no se superponen entre todos los enrutadores de transacciones distribuidas (enrutadores). Los fragmentos se representan como objetos de secuencia local en los enrutadores; por lo tanto, las operaciones de secuencia, como `nextval` y `currval`, se ejecutan localmente. Los enrutadores funcionan de forma independiente y solicitan nuevos fragmentos de la secuencia distribuida cuando es necesario.

Para obtener más información sobre las secuencias, consulte CREATE SEQUENCE en la documentación de PostgreSQL.

Temas

- Solicitud de un nuevo fragmento
- Limitaciones
- Opciones no admitidas
- Ejemplos
- Vistas de secuencias
- Solución de problemas con secuencias

Solicitud de un nuevo fragmento

Los tamaños de los fragmentos asignados a los enrutadores se configuran con el parámetro `rds_aurora.limitless_sequence_chunk_size`. El valor predeterminado es `250000`. Inicialmente, cada enrutador posee dos fragmentos: uno activo y otro reservado. Los fragmentos activos se utilizan para configurar los objetos de secuencia locales (configuración de `minvalue` y `maxvalue`) y los fragmentos reservados se almacenan en una tabla de catálogo interna. Cuando un fragmento activo alcanza el valor mínimo o máximo, se sustituye por el fragmento reservado. Para ello, `ALTER SEQUENCE` se utiliza internamente, es decir, que `AccessExclusiveLock` se adquiere.

Los trabajadores en segundo plano se ejecutan cada diez segundos en los nodos del enrutador para escanear las secuencias en busca de fragmentos reservados usados. Si se encuentra un fragmento

usado, el trabajador solicita un nuevo fragmento de la secuencia distribuida. Asegúrese de establecer el tamaño del fragmento lo suficientemente grande como para que los trabajadores en segundo plano tengan tiempo suficiente para solicitar otros nuevos. Las solicitudes remotas nunca se realizan en el contexto de las sesiones de usuario, lo que significa que no se puede solicitar una nueva secuencia directamente.

Limitaciones

Las siguientes limitaciones se aplican a las secuencias en Base de datos ilimitada de Aurora PostgreSQL:

- El catálogo `pg_sequence`, la función `pg_sequences` y la instrucción `SELECT * FROM` *`sequence_name`* muestran solo el estado de la secuencia local, no el estado distribuido.
- Se garantiza que los valores de secuencia son únicos y monótonos dentro de una sesión. Sin embargo, pueden estar desordenados al ejecutar instrucciones `nextval` en otras sesiones, si esas sesiones están conectadas a otros enrutadores.
- Asegúrese de que el tamaño de la secuencia (número de valores disponibles) sea lo suficientemente grande como para distribuirse entre todos los enrutadores. Para utilizar el parámetro `rds_aurora.limitless_sequence_chunk_size`, configure `chunk_size`. (Cada enrutador tiene dos fragmentos).
- La opción `CACHE` es compatible, pero la memoria caché debe ser inferior a `chunk_size`.

Opciones no admitidas

Las siguientes opciones no se admiten para secuencias en Base de datos ilimitada de Aurora PostgreSQL.

Funciones de manipulación de secuencias

No se admite la función `setval`. Para obtener más información, consulte Sequence Manipulation Functions en la documentación de PostgreSQL.

CREATE SEQUENCE

No se admiten las siguientes opciones.

```
CREATE [{ TEMPORARY | TEMP} | UNLOGGED] SEQUENCE
    [[ NO ] CYCLE]
```

Para obtener más información, consulte CREATE SEQUENCE en la documentación de PostgreSQL.

ALTER SEQUENCE

No se admiten las siguientes opciones.

```
ALTER SEQUENCE
    [[ NO ] CYCLE]
```

Para obtener más información, consulte ALTER SEQUENCE en la documentación de PostgreSQL.

ALTER TABLE

El comando `ALTER TABLE` no admite secuencias.

Ejemplos

CREATE/DROP SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
       1
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16960 |       20 |        1 |            1 |  10000 |      1 |        1 | f
(1 row)

% connect to another router
postgres_limitless=> SELECT nextval('s');
```

```
 nextval
---------
   10001
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16959 |       20 |    10001 |            1 |  20000 |  10001 |        1 | f
(1 row)

postgres_limitless=> DROP SEQUENCE s;
DROP SEQUENCE
```

ALTER SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> ALTER SEQUENCE s RESTART 500;
ALTER SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
     500
(1 row)

postgres_limitless=> SELECT currval('s');

 currval
---------
     500
(1 row)
```

Funciones de manipulación de secuencias

```
postgres=# CREATE TABLE t(a bigint primary key, b bigint);
CREATE TABLE
```

```
postgres=# CREATE SEQUENCE s minvalue 0 START 0;
CREATE SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));
INSERT 0 1

postgres=# SELECT * FROM t;

 a | b
---+---
 0 | 0
 1 | 1
(2 rows)

postgres=# ALTER SEQUENCE s RESTART 10000;
ALTER SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# SELECT * FROM t;

   a   |   b
-------+-------
     0 |     0
     1 |     1
 10000 | 10000
(3 rows)
```

Vistas de secuencias

Base de datos ilimitada de Aurora PostgreSQL ofrece las siguientes vistas para las secuencias.

rds_aurora.limitless_distributed_sequence

Esta vista muestra el estado y la configuración de una secuencia distribuida. Las columnas `minvalue`, `maxvalue`, `start`, `inc` y `cache` tienen el mismo significado en la vista pg_sequences y muestran las opciones con las que se ha creado la secuencia. La columna `lastval` muestra el último valor asignado o reservado en un objeto de secuencia distribuida. Esto no significa que el valor ya se haya utilizado, ya que los enrutadores mantienen los fragmentos de secuencia de forma local.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_distributed_sequence WHERE
 sequence_name='test_serial_b_seq';

 schema_name |   sequence_name   | lastval | minvalue |  maxvalue  | start | inc |
 cache
-------------+-------------------+---------+----------+------------+-------+-----
+-------
 public      | test_serial_b_seq | 1250000 |        1 | 2147483647 |     1 |   1 |
   1
(1 row)
```

rds_aurora.limitless_sequence_metadata

Esta vista muestra los metadatos de secuencia distribuida y agrega los metadatos de secuencia de los nodos del clúster. Utiliza las siguientes columnas:

- `subcluster_id`: es el ID del nodo del clúster que posee un fragmento.
- Fragmento activo: es un fragmento de una secuencia que se está utilizando (`active_minvalue`, `active_maxvalue`).
- Fragmento reservado: es el fragmento local que se utilizará a continuación (`reserved_minvalue`, `reserved_maxvalue`).
- `local_last_value`: es el último valor observado de una secuencia local.
- `chunk_size`: es el tamaño de un fragmento, tal como se configuró al crearlo.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_sequence_metadata WHERE
 sequence_name='test_serial_b_seq' order by subcluster_id;
```

```
 subcluster_id |   sequence_name   | schema_name | active_minvalue | active_maxvalue
 | reserved_minvalue | reserved_maxvalue | chunk_size | chunk_state |
 local_last_value
---------------+-------------------+-------------+-----------------
+-----------------+-------------------+-------------------+------------
+-------------+------------------
 1             | test_serial_b_seq | public      |          500001 |          750000
 |           1000001 |           1250000 |     250000 |           1 |
 550010
 2             | test_serial_b_seq | public      |          250001 |          500000
 |            750001 |           1000000 |     250000 |           1 |

(2 rows)
```

Solución de problemas con secuencias

Pueden producirse los siguientes problemas con las secuencias.

El tamaño del fragmento no es lo suficientemente grande

Si el tamaño del fragmento no es lo suficientemente grande y la tasa de transacciones es alta, es posible que los trabajadores en segundo plano no tengan tiempo suficiente para solicitar nuevos fragmentos antes de que se agoten los fragmentos activos. Esto puede provocar contención y eventos de espera como `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager`, `Lockrelation` y `LWlock:bufferscontent`.

Aumente el valor del parámetro `rds_aurora.limitless_sequence_chunk_size`.

La caché de secuencias está configurada a un nivel demasiado alto

En PostgreSQL, el almacenamiento en caché de secuencias se produce en el nivel de sesión. Cada sesión asigna valores de secuencia sucesivos durante un acceso al objeto de secuencia y aumenta los `last_value` del objeto de secuencia en consecuencia. Luego, los siguientes usos de `nextval` en esa sesión simplemente devuelven los valores preasignados sin tocar el objeto de secuencia.

Los números asignados pero no utilizados en una sesión se pierden al finalizar la sesión, lo que genera huecos en la secuencia. Esto puede consumir el sequence_chunk rápidamente y provocar contención y eventos de espera como `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager`, `Lockrelation` y `LWlock:bufferscontent`.

Reduzca la configuración de la caché de secuencias.

La siguiente figura muestra los eventos de espera causados por problemas de secuencia.

Comandos de lenguaje de manipulación de datos (DDL) y SQL de procesamiento de consultas compatibles y no compatibles

En la siguiente tabla se enumeran los comandos DML compatibles y no compatibles con Base de datos ilimitada de Aurora PostgreSQL, además de referencias a las limitaciones y otra información.

Comando	¿Se admite?	Limitaciones o más información
ABORT	Sí	Ninguno
ANALYZE	Sí	ANALYZE
BEGIN	Sí	Ninguno
CALL	Sí	Ninguno
CHECKPOINT	Sí	Ninguno
CLOSE	Sí	Ninguno
CLUSTER	Sí	CLUSTER
COMMIT	Sí	Ninguno
COMMIT PREPARED	No	No aplicable
COPY	Sí	Ninguno
DEALLOCATE	Sí	Ninguno
DECLARE	Sí	Ninguno
DELETE	Sí	Ninguno
DISCARD	Sí	Ninguno
DO	Sí	Ninguno
END	Sí	Ninguno

Comando	¿Se admite?	Limitaciones o más información
EXECUTE	Sí	Ninguno
EXPLAIN	Sí	EXPLAIN
FETCH	Sí	Ninguno
IMPORT FOREIGN SCHEMA	No	No aplicable
INSERT	Sí	INSERT
LISTEN	No	No aplicable
LOCK	Sí	Ninguno
MERGE	No	No aplicable
MOVE	Sí	Ninguno
NOTIFY	No	No aplicable
OPEN	Sí	Ninguno
PREPARE	Sí	Ninguno
PREPARE TRANSACTION	No	No aplicable
REFRESH MATERIALIZED VIEW	No	No aplicable
REINDEX	No	No aplicable
RELEASE SAVEPOINT	Sí	Ninguno
ROLLBACK	Sí	Ninguno
ROLLBACK PREPARED	No	No aplicable
ROLLBACK TO SAVEPOINT	Sí	Ninguno

Comando	¿Se admite?	Limitaciones o más información
SAVEPOINT	Sí	Ninguno
SELECT	Sí	Ninguno
SELECT INTO	Sí	Ninguno
SHOW	Sí	Ninguno
START TRANSACTION	Sí	Ninguno
UNLISTEN	No	Ninguno
UPDATE	Sí	UPDATE
UPDATE … WHERE CURRENT OF	No	No aplicable
VACUUM	Sí	VACUUM
VALUES	Sí	Ninguno

Limitaciones del DML y otra información para Base de datos ilimitada de Aurora PostgreSQL

En los siguientes temas se describen las limitaciones o se proporciona más información sobre los comandos SQL de procesamiento de consultas y DML en Base de datos ilimitada de Aurora PostgreSQL.

Temas

- ANALYZE
- CLUSTER
- EXPLAIN
- INSERT
- UPDATE
- VACUUM

ANALYZE

El comando `ANALYZE` recopila estadísticas sobre el contenido de las tablas de la base de datos. Posteriormente, el planificador de consultas utiliza estas estadísticas para ayudar a determinar los planes de ejecución más eficaces para las consultas. Para obtener más información, consulte ANALYZE en la documentación de PostgreSQL.

En Base de datos ilimitada de Aurora PostgreSQL, el comando `ANALYZE` recopila estadísticas de tablas en todos los enrutadores y particiones cuando se ejecuta.

Para evitar el cálculo de estadísticas de todos los enrutadores durante las ejecuciones de `ANALYZE`, las estadísticas de la tabla se calculan en uno de los enrutadores y, a continuación, se copian en los enrutadores homólogos.

CLUSTER

El comando `CLUSTER` reordena físicamente una tabla en función de un índice. El índice ya debe estar definido en la tabla. En Base de datos ilimitada de Aurora PostgreSQL, la agrupación es local en la parte del índice que está presente en cada partición.

Para obtener más información, consulte CLUSTER en la documentación de PostgreSQL.

EXPLAIN

Utilice el siguiente parámetro para configurar la salida del comando `EXPLAIN`:

- `rds_aurora.limitless_explain_options`: indica lo que se debe incluir en la salida de `EXPLAIN`. El valor predeterminado es `single_shard_optimization` y muestra si los planes están optimizados para una sola partición, aunque los planes para particiones no están incluidos.

En este ejemplo, la salida de `EXPLAIN` no muestra los planes de las particiones.

```
postgres_limitless=> EXPLAIN SELECT * FROM employees where id =25;

                      QUERY PLAN
------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
 Single Shard Optimized
(2 rows)
```

Ahora configuraremos `rds_aurora.limitless_explain_options` para que incluya `shard_plans` y `single_shard_optimization`. Podemos ver los planes de ejecución de las instrucciones tanto en los enrutadores como en las particiones. Además, desactivamos el parámetro `enable_seqscan` para exigir que el escaneo de índices se utilice en la capa de particiones.

```
postgres_limitless=> SET rds_aurora.limitless_explain_options = shard_plans,
 single_shard_optimization;
SET

postgres_limitless=> SET enable_seqscan = OFF;
SET

postgres_limitless=> EXPLAIN SELECT * FROM employees WHERE id = 25;

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Remote Plans from Shard postgres_s4:
         Index Scan using employees_ts00287_id_idx on employees_ts00287
 employees_fs00003  (cost=0.14..8.16 rows=1 width=15)
           Index Cond: (id = 25)
 Single Shard Optimized
(5 rows)
```

Para obtener más información acerca del comando EXPLAIN, consulte EXPLAIN en la documentación de PostgreSQL.

INSERT

Base de datos ilimitada de Aurora PostgreSQL admite la mayoría de comandos INSERT.

PostgreSQL no tiene un comando UPSERT explícito, pero admite las instrucciones INSERT ... ON CONFLICT.

INSERT ... ON CONFLICT no se admite si la acción conflictiva tiene una subconsulta o una función mutable:

```
-- RANDOM is a mutable function.
INSERT INTO sharded_table VALUES (1, 100) ON CONFLICT (id) DO UPDATE SET other_id =
 RANDOM();

ERROR: Aurora Limitless Tables doesn't support pushdown-unsafe functions with DO UPDATE
 clauses.
```

Para obtener más información acerca del comando INSERT, consulte INSERT en la documentación de PostgreSQL.

UPDATE

La actualización de la clave de partición es incompatible. Por ejemplo, tiene una tabla particionada llamada customers, con una clave de partición customer_id. Las siguientes instrucciones DML provocan errores:

```
postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_id =1;
ERROR:  Shard key column update is not supported

postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_name='abc';
ERROR:  Shard key column update is not supported
```

Para actualizar una clave de partición, primero hay que DELETE la fila con la clave de partición y, a continuación, INSERT una nueva fila con el valor de la clave de partición actualizada.

Para obtener más información acerca del comando UPDATE, consulte Updating data en la documentación de PostgreSQL.

VACUUM

Puede vaciar tanto en las tablas particionadas como en las tablas de referencia. Base de datos ilimitada de Aurora PostgreSQL admite por completo las siguientes funciones `VACUUM`:

- VACUUM
- ANALYZE
- DISABLE_PAGE_SKIPPING
- FREEZE
- FULL
- INDEX_CLEANUP
- PARALLEL
- PROCESS_TOAST
- TRUNCATE
- VERBOSE

`VACUUM` en Base de datos ilimitada de Aurora PostgreSQL tiene las siguientes limitaciones:

- No se admite la extensión pg_visibility_map.
- No se admite la comprobación de índices no utilizados con la vista pg_stat_all_indexes.
- No se implementan las vistas consolidadas de pg_stat_user_indexes, pg_class y pg_stats.

Para obtener más información acerca del comando `VACUUM`, consulte VACUUM en la documentación de PostgreSQL. Para obtener más información sobre cómo funciona el vacío en Base de datos ilimitada de Aurora PostgreSQL, consulte Recuperación de espacio de almacenamiento mediante el vaciado.

Variables de Base de datos ilimitada de Aurora PostgreSQL

Puede usar las siguientes variables para configurar Base de datos ilimitada de Aurora PostgreSQL.

rds_aurora.limitless_active_shard_key

Esta variable establece una única clave de partición al consultar la base de datos, lo que hace que todas las consultas `SELECT` y DML se asocien a la clave de partición como predicado constante. Para obtener más información, consulte Definición de una clave de partición activa.

rds_aurora.limitless_create_table_collocate_with

Defina esta variable de sesión con un nombre de tabla específico para colocar las tablas recién creadas en esa tabla. Para obtener más información, consulte Creación de tablas ilimitadas con variables.

rds_aurora.limitless_create_table_mode

Establece el modo de creación de la tabla. Para obtener más información, consulte Creación de tablas ilimitadas con variables.

rds_aurora.limitless_create_table_shard_key

Defina esta variable en una matriz de nombres de columnas para utilizarlos como claves de partición. Para obtener más información, consulte Creación de tablas ilimitadas con variables.

rds_aurora.limitless_explain_options

Indica lo que se debe incluir en la salida de `EXPLAIN`. Para obtener más información, consulte EXPLAIN.

Parámetros de clúster en Base de datos ilimitada de Aurora PostgreSQL

Puede usar los siguientes parámetros de clúster de base de datos para configurar Base de datos ilimitada de Aurora PostgreSQL.

rds_aurora.limitless_adaptive_fetch_size

Mejora la captura previa por lotes. Si se establece en `true`, este parámetro permite un tamaño de captura autoajustable (adaptable) para la captura previa. Si se establece en `false`, el tamaño de captura es constante.

rds_aurora.limitless_auto_scale_options

Establece las opciones disponibles para añadir enrutadores o dividir particiones en un grupo de particiones de base de datos. Este valor puede ser `add_router`, `split_shard` o ambos.

Para obtener más información, consulte Adición de un enrutador a un grupo de partición de base de datos y División de una partición en un grupo de particiones de base de datos.

rds_aurora.limitless_distributed_deadlock_timeout

Es el periodo de tiempo, en milisegundos, durante el que se espera un bloqueo antes de verificar si hay una condición de interbloqueo distribuido. El valor predeterminado es de `1000` (un segundo).

Para obtener más información, consulte Interbloqueos distribuidos en Base de datos ilimitada de Aurora PostgreSQL.

rds_aurora.limitless_enable_auto_scale

Permite añadir enrutadores y dividir las particiones en un grupo de particiones de base de datos.

Para obtener más información, consulte Adición de un enrutador a un grupo de partición de base de datos y División de una partición en un grupo de particiones de base de datos.

rds_aurora.limitless_finalize_split_shard_mode

Determina cómo finalizan las divisiones de particiones iniciadas por el sistema. Para obtener más información, consulte División de una partición en un grupo de particiones de base de datos.

rds_aurora.limitless_maximum_adaptive_fetch_size

Establece el límite superior del tamaño de captura adaptable. El rango es de `1` a `INT_MAX`. El valor predeterminado es `1000`.

Uso de una base de datos global de Amazon Aurora

Con la característica de base de datos global de Amazon Aurora, puede configurar varios clústeres de bases de datos Aurora que abarquen varias Regiones de AWS. Aurora sincroniza automáticamente todos los cambios realizados en el clúster de base de datos principal con uno o más clústeres secundarios. Una base de datos global de Aurora tiene un clúster de base de datos principal en una región y hasta 10 clústeres de base de datos secundarios en diferentes regiones. Esta configuración multirregional proporciona una recuperación rápida de cualquier interrupción que pueda afectar a toda una Región de AWS. Tener una copia completa de todos los datos en varias ubicaciones geográficas también permite operaciones de lectura de baja latencia para las aplicaciones que se conectan desde ubicaciones muy distantes de todo el mundo.

Temas

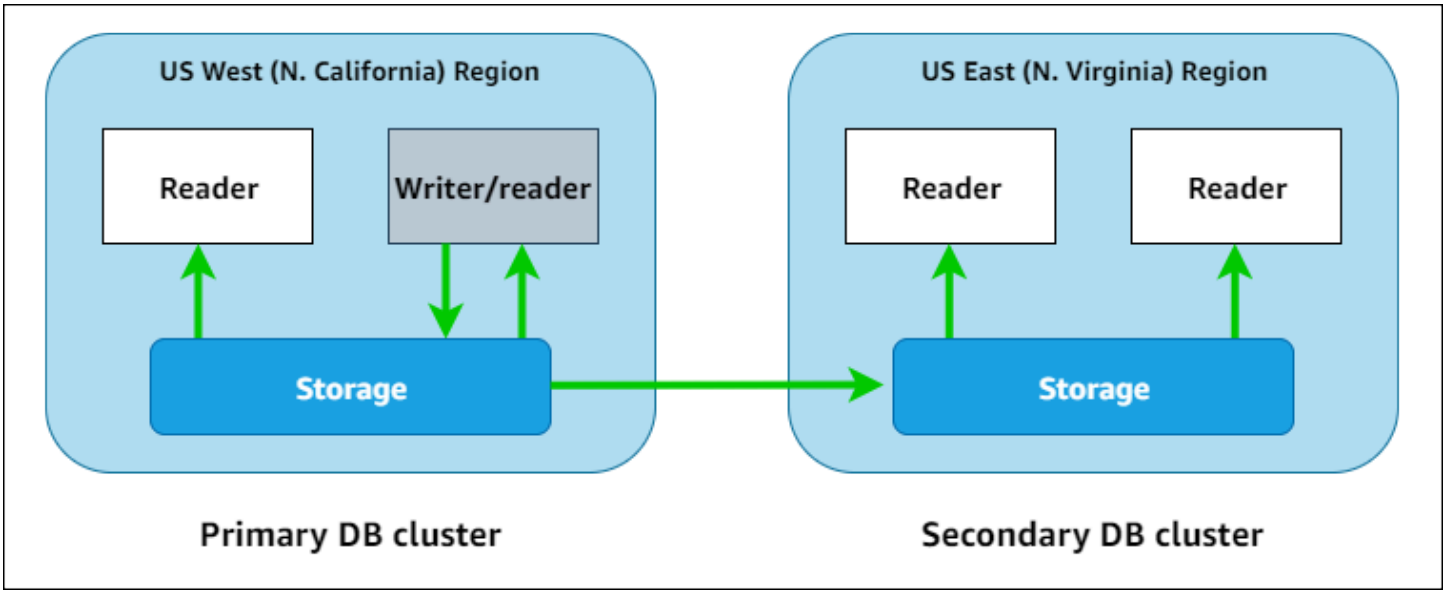
- Información general sobre la base de datos global de Amazon Aurora
- Ventajas de la base de datos global de Amazon Aurora
- Disponibilidad en regiones y versiones
- Limitaciones de la base de datos global de Amazon Aurora
- Introducción a la base de datos global de Amazon Aurora
- Administración de una base de datos global de Amazon Aurora
- Conexión a la base de datos global de Amazon Aurora
- Uso del reenvío de escritura en una base de datos Amazon Aurora global
- Uso de la transición o la conmutación por error en la base de datos global de Amazon Aurora
- Monitoreo de una base de datos global de Amazon Aurora
- Uso de las implementaciones azul/verde para base de datos global de Amazon Aurora
- Uso de las bases de datos globales de Amazon Aurora con otros servicios de AWS
- Actualización de una base de datos global de Amazon Aurora

Información general sobre la base de datos global de Amazon Aurora

Por medio de la característica de base de datos global de Amazon Aurora, puede ejecutar sus aplicaciones distribuidas globalmente a través de una única base de datos de Aurora que abarca múltiples Regiones de AWS.

Una base de datos global de Aurora consta de una Región de AWS principal donde se escriben los datos, y hasta 10 Regiones de AWS secundarias de solo lectura. Emita operaciones de escritura en el clúster de base de datos principal en la Región de AWS principal. La forma más cómoda de hacerlo es conectarse al punto de conexión del escritor de la base de datos global Aurora, que siempre apunta al clúster de base de datos principal, incluso después de una transición o una conmutación por error a otra Región de AWS. Después de cualquier operación de escritura, Aurora replica los datos en las Regiones de AWS secundarias mediante una infraestructura dedicada, con una latencia normalmente inferior a un segundo.

En el siguiente diagrama, puede encontrar un ejemplo de base de datos global de Aurora que abarca dos Regiones de AWS.



Puede escalar verticalmente el clúster secundario independientemente agregando una o varias Aurora instancias del lector de Aurora para servir a cargas de trabajo de solo lectura. Puede utilizar Aurora Serverless v2 para las instancias del lector para lograr un escalado aún más granular y flexible.

Solo el clúster primario realiza operaciones de escritura. Los clientes que realizan operaciones de escritura se conectan al punto de conexión del escritor de la base de datos global de Aurora, que siempre apunta a la instancia de base de datos del clúster primario. Como se muestra en el diagrama, Aurora utiliza el volumen de almacenamiento de clúster y no el motor de base de datos para una replicación rápida y de baja sobrecarga. Para obtener más información, consulte Información general del almacenamiento de Amazon Aurora.

La base de datos global de Aurora está diseñada para aplicaciones con una huella mundial. Los clústeres de base de datos secundarios de solo lectura en varias Regiones de AWS ayudan a optimizar las operaciones de lectura más cercanas a los usuarios de la aplicación. Con la característica del reenvío de escritura, también puede configurar su base de datos global para que los clústeres secundarios envíen solicitudes de escritura al principal. Para obtener más información, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

La base de datos global de Aurora admite dos operaciones diferentes para cambiar la región del clúster de base de datos principal, según el caso: transición de base de datos global de Aurora y conmutación por error de base de datos global.

- En el caso de procedimientos operativos planificados, como la rotación regional, utilice el mecanismo de transición (antes denominad “conmutación por error planificada administrada”). Con esta característica, puede reubicar el clúster principal de una base de datos global de Aurora en buen estado a una de sus regiones secundarias sin necesidad de perder datos. Para obtener más información, consulte Ejecución de transiciones para bases de datos globales de Amazon Aurora.
- Para recuperar la base de datos global de Aurora después de una interrupción en la región principal, utilice el mecanismo de conmutación por error. Con esta característica, realiza una conmutación por error del clúster de base de datos principal a otra región (conmutación por error entre regiones). Para obtener más información, consulte Ejecución de la conmutación por error administrada para bases de datos globales de Aurora.

Ventajas de la base de datos global de Amazon Aurora

Mediante el uso de la base de datos global de Aurora, puede obtener las siguientes ventajas:

- Lecturas globales con latencia local: si tiene oficinas en todo el mundo, puede utilizar la base de datos global de Aurora para mantener actualizadas sus principales orígenes de información en la Región de AWS principal. Las oficinas en sus otras regiones pueden acceder a la información en su propia región, con latencia local.

- Clústeres de base de datos de Aurora secundarios y escalables: puede escalar los clústeres secundarios; para ello, agregue más instancias de solo lectura a una Región de AWS secundaria. El clúster secundario es de solo lectura, por lo que puede admitir hasta 16 instancias de base de datos de Aurora de solo lectura en lugar del límite habitual de 15 para un solo clúster de Aurora.
- Replicación rápida de clústeres de base de datos de Aurora primarios a secundarios: la replicación realizada por una base de datos global de Aurora tiene poco impacto en el rendimiento del clúster de base de datos principal. Los recursos de las instancias de bases de datos están totalmente dedicados a servir a las cargas de trabajo de lectura y escritura de la aplicación.
- Recuperación de interrupciones en toda la región: los clústeres secundarios le permiten hacer que una base de datos global de Aurora esté disponible en una Región de AWS principal nueva más rápido (RTO más bajo) y con menos pérdida de datos (RPO más bajo) que las soluciones de replicación tradicionales.

Disponibilidad en regiones y versiones

La disponibilidad y el soporte de la característica varía según las versiones específicas de cada motor de base de datos de Aurora y entre Regiones de AWS. Para obtener más información acerca de la versión y la disponibilidad de las regiones con la base de datos global de Aurora, consulte Regiones y motores de base de datos admitidos para bases de datos globales Aurora.

Limitaciones de la base de datos global de Amazon Aurora

Las limitaciones siguientes se aplican actualmente a la base de datos global de Aurora:

- La base de datos global de Aurora está disponibles en ciertas Regiones de AWS y para versiones específicas de Aurora MySQL y Aurora PostgreSQL. Para obtener más información, consulte Regiones y motores de base de datos admitidos para bases de datos globales Aurora.
- La base de datos global de Aurora tienen requisitos de configuración concretos para las clases de instancias de base de datos compatibles de Aurora, la cantidad máxima de Regiones de AWS, etc. Para obtener más información, consulte Requisitos de configuración de una base de datos Amazon Aurora global.
- Para la compatibilidad de Aurora MySQL con MySQL 5.7, las transiciones de base de datos global de Aurora requieren la versión 2.09.1 o una superior.
- Solo puede realizar transiciones o conmutaciones por error administradas entre regiones en la base de datos global de Aurora si los clústeres de base de datos principal y secundario tienen las mismas versiones principal, secundaria y de nivel de parche del motor. Según el motor y las

versiones del motor, es posible que los niveles de parche deban ser idénticos o diferentes. Para obtener una lista de los motores y las versiones de motores que permiten estas operaciones entre clústeres principales y secundarios con diferentes niveles de parches, consulte Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones. Si las versiones del motor requieren niveles de parches idénticos, puede realizar la conmutación por error manualmente por medio de los pasos que se indican en Ejecución de la conmutación por error manual para bases de datos globales de Aurora.

- La base de datos global de Aurora actualmente no admite las siguientes características de Aurora:
  - Aurora Serverless v1
  - Búsqueda de datos anteriores en Aurora.
- Para conocer las limitaciones del uso de la característica RDS Proxy con la base de datos global, consulte Limitaciones de RDS Proxy con bases de datos globales.
- La actualización automática de versiones secundarias no se aplica a clústeres de Aurora MySQL y Aurora PostgreSQL que formen parte de una base de datos global. Tenga en cuenta que puede especificar esta configuración para una instancia de base de datos que forme parte de un clúster de base de datos global, pero la configuración no tendrá ningún efecto.
- La base de datos global de Aurora actualmente no admite Aurora Auto Scaling para clústeres de bases de datos secundarios.
- Para utilizar las secuencias de actividades de base de datos (DAS) en la base de datos global de Aurora que ejecuta Aurora MySQL 5.7, la versión del motor debe ser la 2.08 o superior. Para obtener más información sobre DAS, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.
- Las limitaciones siguientes se aplican actualmente a la actualización de la base de datos global de Aurora:
  - No puede aplicar un grupo de parámetros personalizado al clúster de base de datos global mientras realiza una actualización importante de la versión de esa base de datos global de Aurora. Se crean grupos de parámetros personalizados en cada región del clúster global y se aplican manualmente a los clústeres regionales después de la actualización.
  - Con una base de datos global de Aurora basada en Aurora MySQL, no se puede realizar una actualización local desde la versión 2 a la versión 3 de Aurora MySQL si el parámetro `lower_case_table_names` está activado. Para obtener más información sobre los métodos que puede utilizar, consulte Actualizaciones de la versión principal.
  - Con la base de datos global de Aurora, no se puede realizar una actualización de versión importante del motor de base de datos de Aurora PostgreSQL si la característica Objetivo de

punto de recuperación (RPO) está habilitada. Para obtener información sobre la característica RPO, consulte Administración de RPO para bases de datos globales basadas en Aurora PostgreSQL–.

- Con una base de datos global de Aurora, no se puede realizar una actualización de versión secundaria de la versión de Aurora MySQL 3.01 o 3.02 a la versión 3.03 o una posterior mediante el proceso estándar. Para obtener más información sobre el proceso que se debe usar, consulte Actualización de Aurora MySQL mediante la modificación de la versión del motor.

Para obtener información sobre cómo actualizar la base de datos global de Aurora, consulte Actualización de una base de datos global de Amazon Aurora.

- No puede detener ni iniciar por separado los clústeres de base de datos de Aurora en su base de datos global. Para obtener más información, consulte Detención e inicio de un clúster de bases de datos de Amazon Aurora.
- Las instancias de base de datos del lector de Aurora conectadas al clúster de base de datos de Aurora secundario pueden reiniciarse en determinadas circunstancias. Si la instancia de base de datos del escritor de la Región de AWS principal se reinicia o se conmuta por error, las instancias de base de datos del lector también se reinician en las regiones secundarias. El clúster secundario no estará disponible hasta que todas las instancias de base de datos del lector vuelvan a estar sincronizadas con la instancia del escritor del clúster de base de datos principal. El comportamiento del clúster principal cuando se reinicia o se produce una conmutación por error es el mismo que en un clúster de base de datos único y no global. Para obtener más información, consulte Replicación con Amazon Aurora.

  Asegúrese de comprender los impactos en la base de datos global de antes de realizar cambios en el clúster de base de datos principal. Para obtener más información, consulte Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada.
- Actualmente, la base de datos global de Aurora no admite el estado `inaccessible-encryption-credentials-recoverable` en el que Amazon Aurora pierde el acceso a la clave AWS KMS del clúster de base de datos. En estos casos, el clúster de base de datos cifrado entra en el estado `inaccessible-encryption-credentials` de terminal. Para obtener más información sobre estos estados, consulte Ver el estado del clúster de base de datos.
- Secrets Manager no admite Aurora Global Database. Al agregar una región a una base de datos global, primero debe desactivar la integración de Secrets Manager para la instancia de base de datos.
- Los clústeres de base de datos basados en Aurora PostgreSQL que se ejecutan en la base de datos global de Aurora tienen las siguientes limitaciones:

- La administración de caché de clúster no es compatible con los clústeres de base de datos secundarios de Aurora PostgreSQL que forman parte de bases de datos globales de Aurora.
- Si el clúster de base de datos principal de la base de datos global se basa en una réplica de una instancia de Amazon RDS PostgreSQL, no puede crear un clúster secundario. No intente crear un secundario a partir de ese clúster mediante Consola de administración de AWS, la AWS CLI, o la operación `CreateDBCluster` API. Los intentos para hacerlo se agotan y no se crea el clúster secundario.

Se recomienda crear clústeres de base de datos secundarios para las bases de datos globales utilizando la misma versión del motor de base de datos de Aurora que el primario. Para obtener más información, consulte Creación de una base de datos global de Amazon Aurora.

Introducción a la base de datos global de Amazon Aurora

Para comenzar a utilizar la base de datos global de Aurora, decida primero qué motor de base de datos de Aurora desea usar y en qué Regiones de AWS. Solo las versiones específicas de los motores de base de datos de Aurora MySQL y Aurora PostgreSQL en ciertas Regiones de AWS son compatibles con la base de datos global de Aurora. Para ver una lista completa, consulte Regiones y motores de base de datos admitidos para bases de datos globales Aurora.

Puede crear una base de datos de Aurora global de una de las siguientes maneras:

- Cree una nueva base de datos global de Aurora con nuevos clústeres de base de datos de Aurora e instancias de base de datos de Aurora: puede hacerlo siguiendo los pasos descritos en Creación de una base de datos global de Amazon Aurora. Después de crear el clúster principal de base de datos de Aurora, agregue la Región de AWS secundaria siguiendo los pasos que se detallan en Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.
- Use un clúster de base de datos de Aurora existente que admita la característica de base de datos global de Aurora y agréguele una Región de AWS: solo puede hacerlo si su clúster de base de datos de Aurora existente usa una versión del motor de base de datos que sea compatible globalmente.

  Verifique si puede elegir Add regiom (Agregar región) para Action (Acción) en la Consola de administración de AWS cuando se selecciona el clúster de la base de datos de Aurora. Si puede, puede utilizar ese clúster de base de datos de Aurora para el clúster global Aurora. Para obtener más información, consulte Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Antes de crear una base de datos global de Aurora, le recomendamos que comprenda todos los requisitos de configuración.

Temas

- Requisitos de configuración de una base de datos Amazon Aurora global
- Creación de una base de datos global de Amazon Aurora
- Incorporación de una Región de AWS a una base de datos global de Amazon Aurora
- Creación de un clúster de base de datos de Aurora sin pantalla en una región secundaria
- Creación de una base de datos global de Amazon Aurora a partir de una instantánea de Aurora o Amazon RDS

Requisitos de configuración de una base de datos Amazon Aurora global

Antes de empezar a trabajar con la base de datos global, determine cuáles serán los nombres de los clústeres y las instancias, las regiones de AWS, el número de instancias y las clases de instancias que piensa utilizar. Sus opciones deben coincidir con los siguientes requisitos de configuración.

Una base de datos global de Aurora abarca al menos dos Regiones de AWS. La Región de AWS principal admite un clúster de base de datos de Aurora que tiene una instancia de base de datos de Aurora de escritor. Una Región de AWS secundaria ejecuta un clúster de base de datos de Aurora de solo lectura compuesto por réplicas de Aurora. Se requiere al menos una Región de AWS secundaria, pero una base de datos global de Aurora puede tener hasta 10 Regiones de AWS secundarias. La tabla muestra el máximo de clústeres de Aurora base de datos, instancias de base de datos Aurora y réplicas Aurora permitidos en una base de datos Aurora global.

Descripción	Región de AWS principal	Regiones de AWS secundaria
Clústeres de base de datos de Aurora	1	10 (máximo)
Instancias de escritor	1	0
Instancias de solo lectura (réplicas de Aurora), por clúster de Aurora base de datos	15 (máx)	16 (total)

Descripción	Región de AWS principal	Regiones de AWS secundaria
Instancias de solo lectura (máximo permitido, dado el número real de Regiones secundarias)	15 - s	s = total de Regiones de AWS secundarias

Los clústeres de base de datos Aurora que componen una base de datos Aurora global tienen los siguientes requisitos específicos:

- Requisitos de clase de instancia de base de datos – Una base de datos Aurora global requiere clases de instancia de base de datos optimizadas para aplicaciones con uso intensivo de memoria Para obtener información sobre las clases de instancias de base de datos optimizadas para la memoria, consulte Tipos de clase de instancia de base de datos. Recomendamos utilizar una clase de instancia db.r5 o superior.
- Requisitos de Región de AWS: una base de datos global de Aurora necesita un clúster de base de datos de Aurora principal en una Región de AWS y al menos un clúster de base de datos de Aurora secundario en una región diferente. Puede crear hasta 10 clústeres de base de datos de Aurora secundarios (de solo lectura) y cada uno debe estar en una región diferente. En otras palabras, no hay dos clústeres de base de datos de Aurora en una base de datos global de Aurora en la misma Región de AWS.

  Para obtener información sobre las combinaciones del motor de base de datos de Aurora, la versión del motor y la Región de AWS que puede utilizar con la base de datos global de Aurora, consulte Regiones y motores de base de datos admitidos para bases de datos globales Aurora.
- Requisitos de nombre: los nombres que elija para cada uno de los clústeres de base de datos de Aurora deben ser únicos, en todas las Regiones de AWS. No puede usar el mismo nombre para diferentes clústeres de base de datos Aurora aunque estén en diferentes regiones.
- Requisitos de capacidad para Aurora Serverless v2: para una base de datos global con Aurora Serverless v2, la capacidad mínima recomendada para el clúster de base de datos de la Región de AWS principal es de 8 ACU.

Antes de poder seguir los procedimientos de esta sección, necesita una Cuenta de AWS. Complete las tareas de configuración para trabajar con Amazon Aurora. Para obtener más información, consulte Configuración del entorno para Amazon Aurora. También debe completar otros pasos

preliminares para crear cualquier clúster de bases de datos de Aurora. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

Cuando lo tenga todo preparado para configurar la base de datos global, consulte Creación de una base de datos global de Amazon Aurora para saber cuál es el procedimiento adecuado a fin de crear todos los recursos necesarios. También puede seguir el procedimiento Incorporación de una Región de AWS a una base de datos global de Amazon Aurora para crear una base de datos global con un clúster de Aurora existente como clúster principal.

Creación de una base de datos global de Amazon Aurora

Puede crear una base de datos global de Aurora, y los recursos relacionados, con la Consola de administración de AWS, la AWS CLI o la API de RDS mediante los pasos que se indican a continuación.

> **Note**
>
> Si tiene un clúster de base de datos de Aurora existente que ejecuta un motor de base de datos de Aurora compatible en el ámbito global, puede usar una versión simplificada de este procedimiento. En ese caso, puede añadir otra Región de AWS al clúster de base de datos existente a fin de crear la base de datos global de Aurora. Para ello, consulte Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Consola

Los pasos para crear una base de datos global de Aurora comienzan iniciando sesión en una Región de AWS que admita la característica de base de datos global de Aurora. Para ver una lista completa, consulte Regiones y motores de base de datos admitidos para bases de datos globales Aurora.

Uno de los pasos siguientes es elegir una nube virtual privada (VPC) basada en Amazon VPC para su clúster de base de datos de Aurora. Para utilizar su propia VPC, recomendamos que la cree de antemano para que esté disponible para que pueda elegir. Al mismo tiempo, cree subredes relacionadas y, según sea necesario, un grupo de subredes y un grupo de seguridad. Para aprender a hacerlo, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

Para obtener información general acerca de cómo crear un clúster de base de datos Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora.

Para crear una base de datos global de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Create database (Crear base de datos). En la página Create database (Crear base de datos), haga lo siguiente:
   - Elija Standard Create (Creación estándar) para el método de creación de la base de datos. No elija la opción Easy Create (Creación sencilla).
   - En `Engine type`, en la sección Opciones de motor, elija el tipo de motor aplicable, Aurora (compatible con MySQL) o Aurora (compatible con PostgreSQL).
3. Para seguir creando la base de datos global de Aurora, utilice los pasos de los procedimientos que se detallan a continuación.

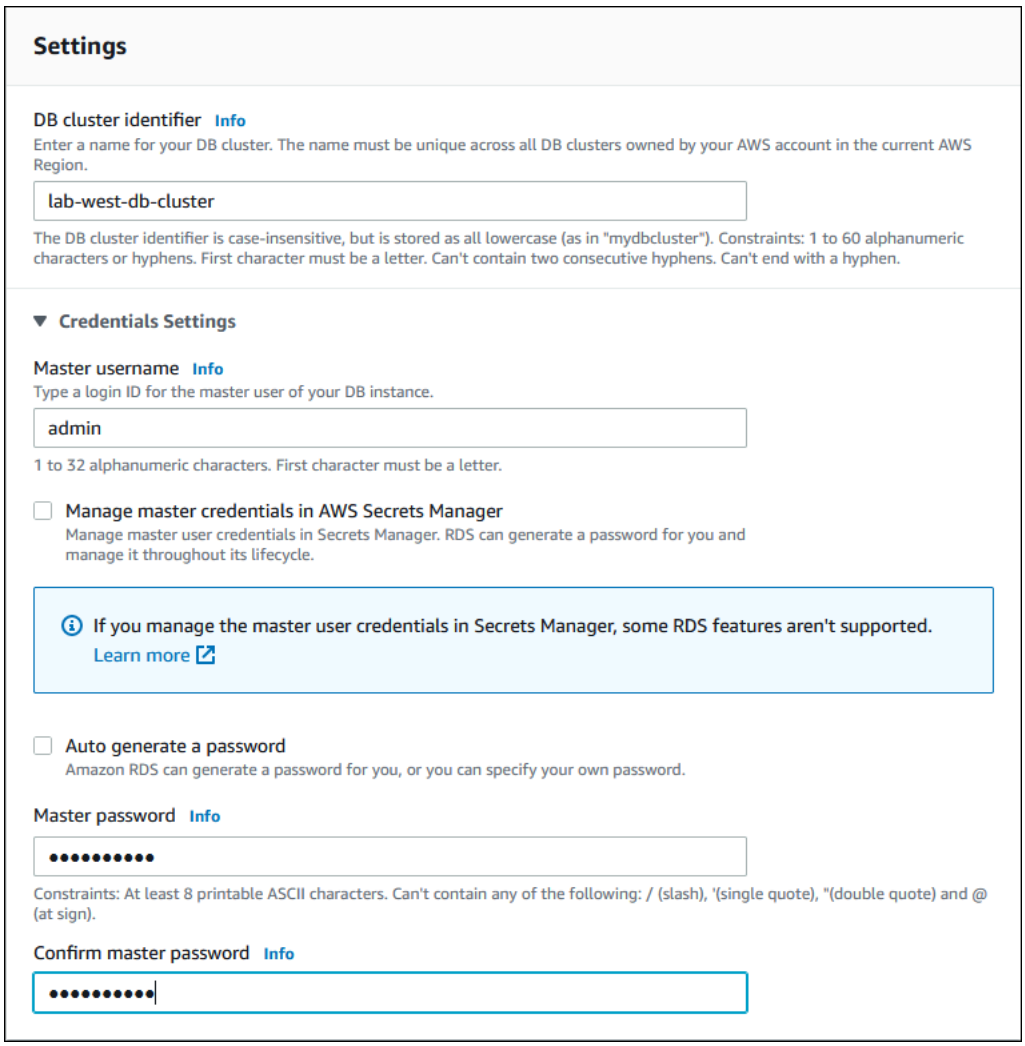
Creación de una base de datos global con Aurora MySQL

Los siguientes pasos se aplican a todas las versiones de Aurora MySQL.

Para crear una base de datos global de Aurora utilizando Aurora MySQL

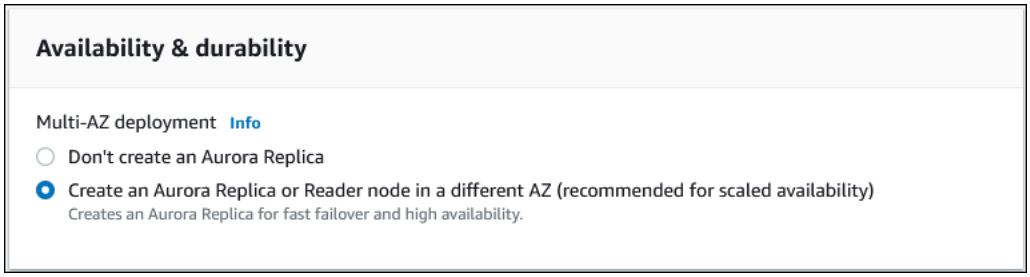
Complete la página Create database (Crear base de datos).

1. En Engine options (Opciones del motor), elija lo siguiente:
   - En Engine version (Versión del motor), elija la versión de Aurora MySQL que desea utilizar para la base de datos global de Aurora.
2. Para Templates (Plantillas), elija Production (Producción). O bien, puede elegir Dev/Test si es apropiado para su caso de uso. No utilice Dev/Test en entornos de producción.
3. En Settings (Configuración), haga lo siguiente:
   a. Introduzca un nombre significativo para el identificador de clúster de base de datos. Cuando termine de crear la base de datos Aurora global, este nombre identifica el clúster de base de datos principal.
   b. Introduzca su propia contraseña para la cuenta de usuario `admin` de la instancia de base de datos, o deje que Aurora genere una para usted. Si elige generar automáticamente una contraseña, obtendrá la opción de copiar la contraseña.



4. Para clase de instancia de base de datos, elija `db.r5.large` u otra clase de instancia de base de datos optimizada para memoria. Recomendamos utilizar una clase de instancia db.r5 o superior.

5. Para disponibilidad y durabilidad, recomendamos que se elija Aurora que cree una réplica Aurora en una zona de disponibilidad (AZ, por sus siglas en inglés) diferente para usted. Si no crea una réplica Aurora ahora, tendrá que hacerlo más tarde.



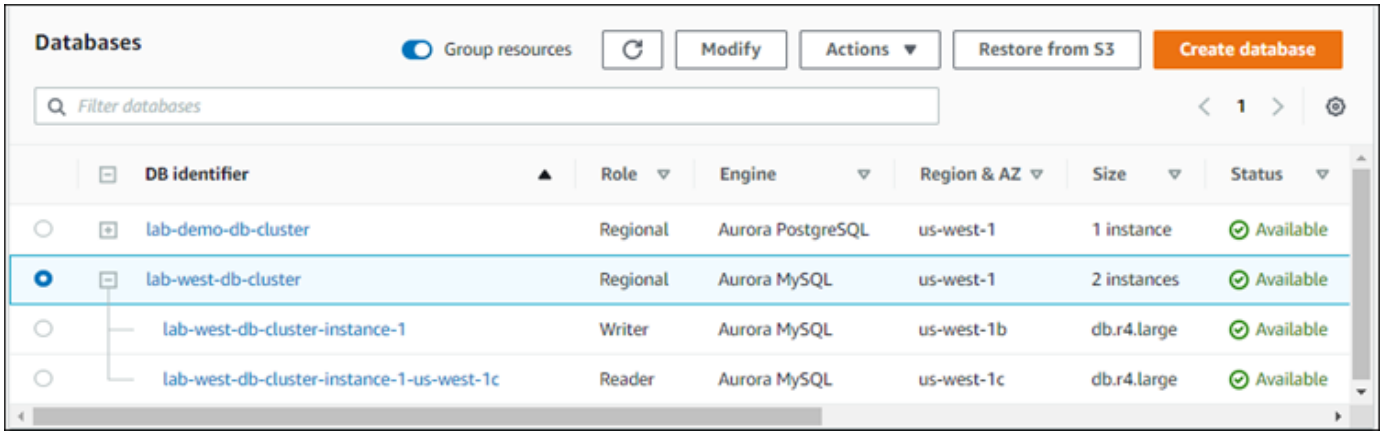
6. En Conectivity (Conectividad), elija la nube virtual privada (VPC) basada en Amazon VPC que defina el entorno de red virtual para esta instancia de base de datos. Puede elegir los valores predeterminados para simplificar esta tarea.
7. Complete la configuración de Database authentication (Autenticación de base de datos). Para simplificar el proceso, puede elegir Password authentication (Autenticación de contraseña) ahora y configurar AWS Identity and Access Management (IAM) más adelante.
8. En Additional configuration (Configuración adicional), haga lo siguiente:

   a. Introduzca un nombre para Initial database name (Nombre de la base de datos inicial) para crear la instancia Aurora de base de datos principal para este clúster. Este es el nodo de escritor del clúster de base de datos Aurora principal.

      Deje los valores predeterminados seleccionados para el grupo de parámetros de clúster de base de datos y el grupo de parámetros de base de datos, a menos que tenga sus propios grupos de parámetros personalizados que desee utilizar.

   b. Desmarque la casilla de verificación Enable backtrack (Habilitar búsquedas de datos anteriores) si está seleccionada. Las bases de datos globales de Aurora no admiten la búsqueda de datos anteriores. De lo contrario, puede aceptar las demás opciones predeterminadas para Additional configuration (Configuración adicional).
9. Elija Create database (Crear base de datos).

   Aurora puede tardar varios minutos en completar el proceso de creación de la instancia de base de datos Aurora, su réplica Aurora y el clúster de base de datos Aurora. Puede saber cuándo el clúster de la base de datos de Aurora está listo para usar como clúster de base de datos principal en una base de datos global de Aurora por su estado. Cuando eso es así, su estado y el del nodo de escritura y réplica está Disponible, como se muestra a continuación.



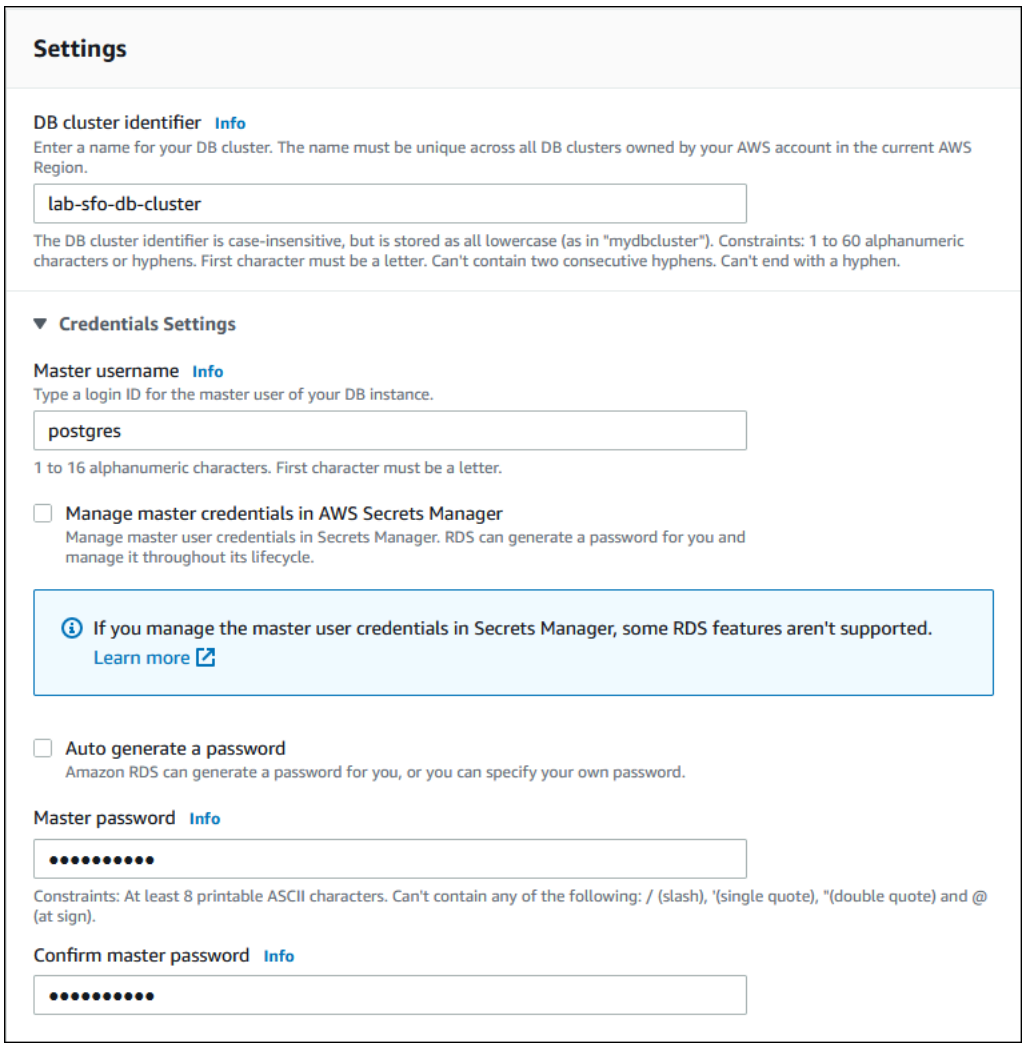
Cuando el clúster de base de datos principal esté disponible, cree la base de datos global de Aurora agregándole un clúster secundario. Para ello, siga los pasos que se indican en Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Creación de una base de datos global con Aurora PostgreSQL

Para crear una base de datos global Aurora utilizando Aurora PostgreSQL

Complete la página Create database (Crear base de datos).

1. En Engine options (Opciones del motor), elija lo siguiente:
   - En Engine version (Versión del motor), elija la versión de Aurora PostgreSQL que desea utilizar para la base de datos global de Aurora.
2. Para Templates (Plantillas), elija Production (Producción). O bien, puede elegir Dev/Test si es apropiado. No utilice Dev/Test en entornos de producción.
3. En Settings (Configuración), haga lo siguiente:
   a. Introduzca un nombre significativo para el identificador de clúster de base de datos. Cuando termine de crear la base de datos Aurora global, este nombre identifica el clúster de base de datos principal.
   b. Introduzca su propia contraseña para la cuenta de administrador predeterminada del clúster de base de datos, o haga que Aurora genere una para usted. Si elige Auto generate a password (Generar automáticamente una contraseña), obtendrá la opción de copiar la contraseña.



4. Para clase de instancia de base de datos, elija `db.r5.large` u otra clase de instancia de base de datos optimizada para memoria. Recomendamos utilizar una clase de instancia db.r5 o superior.
5. Para Availability & durability (Disponibilidad y durabilidad), le recomendamos que elija que Aurora cree una réplica Aurora en una zona de disponibilidad diferente para usted. Si no crea una réplica Aurora ahora, tendrá que hacerlo más tarde.
6. En Conectivity (Conectividad), elija la nube virtual privada (VPC) basada en Amazon VPC que defina el entorno de red virtual para esta instancia de base de datos. Puede elegir los valores predeterminados para simplificar esta tarea.
7. (Opcional) Complete la configuración de Database authentication (Autenticación de base de datos). La autenticación con contraseña siempre está habilitada. Para simplificar el proceso, puede omitir esta sección y configurar IAM o la autenticación de contraseña y Kerberos más adelante.

8. En Additional configuration (Configuración adicional), haga lo siguiente:

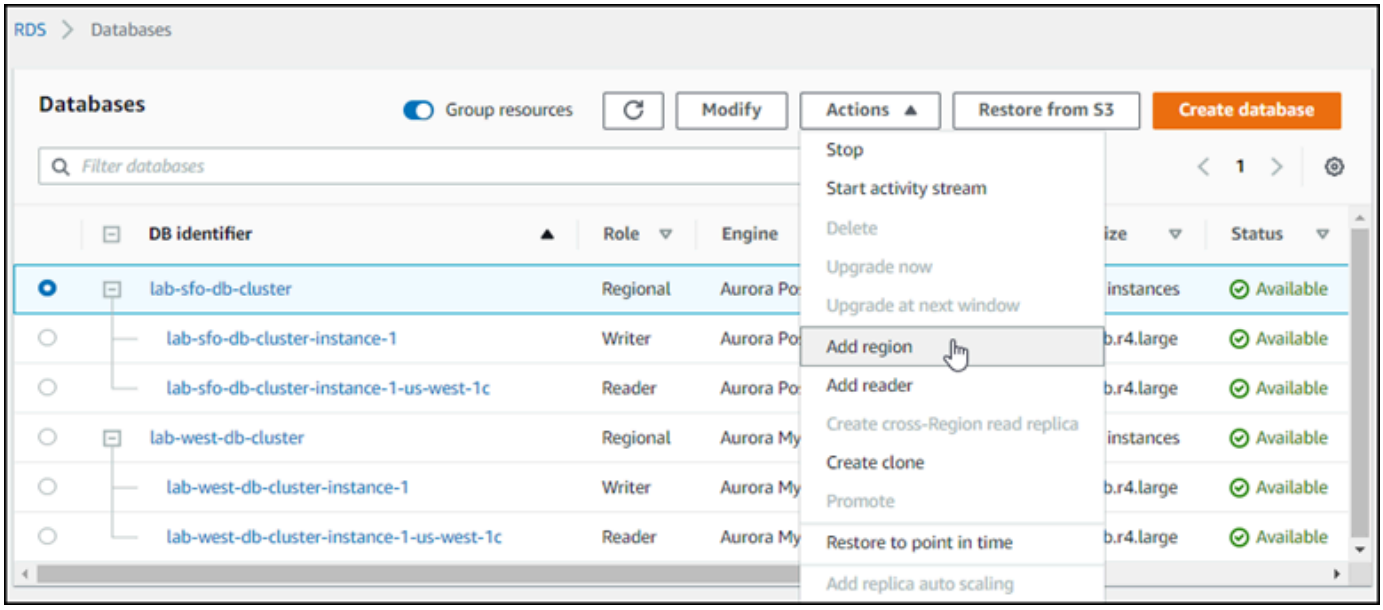
   a. Introduzca un nombre para Initial database name (Nombre de la base de datos inicial) para crear la instancia Aurora de base de datos principal para este clúster. Este es el nodo de escritor del clúster de base de datos Aurora principal.

      Deje los valores predeterminados seleccionados para el grupo de parámetros de clúster de base de datos y el grupo de parámetros de base de datos, a menos que tenga sus propios grupos de parámetros personalizados que desee utilizar.

   b. Acepte todas las demás opciones predeterminadas para Additional configuration (Configuración adicional), como Encryption (Cifrado), Log exports (Exportaciones de registros), etc.

9. Elija Create database (Crear base de datos).

   Aurora puede tardar varios minutos en completar el proceso de creación de la instancia de base de datos Aurora, su réplica Aurora y el clúster de base de datos Aurora. Cuando el clúster está listo para su uso, el clúster de base de datos de Aurora y sus nodos de escritor y réplica muestran el estado Available (Disponible). Esto se convierte en el clúster de base de datos principal de la base de datos Aurora global, después de agregar un secundario.



Cuando se crea el clúster de base de datos principal y está disponible, puede crear uno o más clústeres secundarios siguiendo los pasos de Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

AWS CLI

Los comandos AWS CLI de los procedimientos siguientes realizan las siguientes tareas:

1. Cree una base de datos global de Aurora, asígnele un nombre y especifique el tipo de motor de base de datos de Aurora que va a utilizar.
2. Cree un clúster de base de datos Aurora para la base de datos Aurora global.
3. Cree la instancia de base de datos Aurora para el clúster. Este es el clúster principal de Aurora DB para la base de datos global.
4. Cree una segunda instancia de base de datos para clúster de base de datos Aurora. Este es un lector para completar el clúster de base de datos de Aurora.
5. Cree un segundo clúster de base de datos Aurora en otra región y, a continuación, agréguelo a la base de datos Aurora global, siguiendo los pasos descritos en Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Siga el procedimiento de su motor de base de datos de Aurora.

Creación de una base de datos global con Aurora MySQL

Para crear una base de datos global de Aurora utilizando Aurora MySQL

1. Utilice el comando `create-global-cluster` de la CLI y pase el nombre de la Región de AWS, el motor y la versión de la base de datos de Aurora.

   Para Linux, macOS o:Unix

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-mysql \
       --engine-version version # optional
   ```

   En:Windows

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-mysql ^
       --engine-version version # optional
   ```

Esto crea una base de datos Aurora global "vacía", con solo un nombre (identificador) y un motor de base de datos Aurora. La base de datos Aurora global puede tardar unos minutos en estar disponible. Antes de ir al siguiente paso, utilice el comando `describe-global-clusters` CLI para ver si está disponible.

```
aws rds describe-global-clusters --region primary_region --global-cluster-
identifier global_database_id
```

Cuando la base de datos Aurora global está disponible, puede crear su clúster de base de datos principal Aurora.

2. Para crear un clúster de base de datos Aurora principal, utilice el comando `create-db-cluster` CLI. Incluya el nombre de la base de datos global de Aurora mediante el parámetro `--global-cluster-identifier`.

   Para Linux, macOS o:Unix

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-mysql \
     --engine-version version \
     --global-cluster-identifier global_database_id
   ```

   En:Windows

   ```
   aws rds create-db-cluster ^
     --region primary_region ^
     --db-cluster-identifier primary_db_cluster_id ^
     --master-username userid ^
     --master-user-password password ^
     --engine aurora-mysql ^
     --engine-version version ^
     --global-cluster-identifier global_database_id
   ```

   Utilice el comando `describe-db-clusters` de la AWS CLI para confirmar que el clúster de base de datos de Aurora está listo. Para individuar un clúster de base de datos Aurora

específico, utilice el parámetro `--db-cluster-identifier`. O puede dejar fuera el nombre del clúster de base de datos Aurora en el comando para obtener detalles sobre todos los clústeres de base de datos Aurora en la región dada.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

Cuando se muestra la respuesta `"Status": "available"` para el clúster, está lista para su uso.

3. Cree la instancia de base de datos para el clúster de base de datos Aurora principal. Para ello, utilice el comando de CLI `create-db-instance`. Asigne al comando el nombre de su clúster de Aurora base de datos y especifique los detalles de configuración de la instancia. No necesita pasar los parámetros `--master-username` y `--master-user-password` en el comando, ya que los obtiene del clúster de base de datos Aurora.

   Para el `--db-instance-class`, puede usar solo aquellos de clases optimizadas para memoria, como `db.r5.large`. Recomendamos utilizar una clase de instancia db.r5 o superior. Para obtener información acerca de estas clases, consulte Tipos de clase de instancia de base de datos.

   Para Linux, macOS o:Unix

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-mysql \
  --engine-version version \
  --region primary_region
```

   En:Windows

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-mysql ^
  --engine-version version ^
  --region primary_region
```

Las operaciones `create-db-instance` podrían tardar un tiempo en completarse. Compruebe el estado para ver si la instancia de base de datos Aurora está disponible antes de continuar.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Cuando el comando devuelve un estado de `available`, puede crear otra instancia de base de datos de Aurora para su clúster de base de datos principal. Ésta es la instancia de lector (la réplica Aurora) para el clúster de base de datos Aurora.

4. Para crear otra instancia de base de datos de Aurora para el clúster, utilice el comando CLI `create-db-instance`:

   Para Linux, macOS o:Unix

   ```
   aws rds create-db-instance \
     --db-cluster-identifier primary_db_cluster_id \
     --db-instance-class instance_class \
     --db-instance-identifier replica_db_instance_id \
     --engine aurora-mysql
   ```

   En:Windows

   ```
   aws rds create-db-instance ^
     --db-cluster-identifier primary_db_cluster_id ^
     --db-instance-class instance_class ^
     --db-instance-identifier replica_db_instance_id ^
     --engine aurora-mysql
   ```

Cuando la instancia de base de datos está disponible, la reproducción comienza desde el nodo de escritor a la réplica. Antes de continuar, compruebe que la instancia de base de datos esté disponible con el comando `describe-db-instances` CLI.

En este punto, tiene una base de datos Aurora global con su clúster de base de datos principal Aurora que contiene una instancia de base de datos de escritor y una réplica Aurora. Ahora puede agregar un clúster de base de datos Aurora de solo lectura en una región diferente para completar la base de datos Aurora global. Para ello, siga los pasos que se indican en Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Creación de una base de datos global con Aurora PostgreSQL

Al crear objetos Aurora para una base de datos global de Aurora mediante los siguientes comandos, cada uno puede tardar unos minutos en estar disponible. Recomendamos que después de llevar a cabo cualquier comando dado, compruebe el estado del objeto Aurora específico para asegurarse de que su estado es disponible.

Para ello, utilice el comando de CLI `describe-global-clusters`.

```
aws rds describe-global-clusters --region primary_region
    --global-cluster-identifier global_database_id
```

Para crear una base de datos global Aurora utilizando Aurora PostgreSQL

1. Utilice el comando CLI `create-global-cluster`.

   Para Linux, macOS o:Unix

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-postgresql \
       --engine-version version # optional
   ```

   En:Windows

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-postgresql ^
       --engine-version version # optional
   ```

   Cuando la base de datos Aurora global está disponible, puede crear su clúster de base de datos principal Aurora.
2. Para crear un clúster de base de datos Aurora principal, utilice el comando `create-db-cluster` CLI. Incluya el nombre de la base de datos global de Aurora mediante el parámetro `--global-cluster-identifier`.

   Para Linux, macOS o:Unix

   ```
   aws rds create-db-cluster \
     --region primary_region \
   ```

```
  --db-cluster-identifier primary_db_cluster_id \
  --master-username userid \
  --master-user-password password \
  --engine aurora-postgresql \
  --engine-version version \
  --global-cluster-identifier global_database_id
```

En:Windows

```
aws rds create-db-cluster ^
  --region primary_region ^
  --db-cluster-identifier primary_db_cluster_id ^
  --master-username userid ^
  --master-user-password password ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --global-cluster-identifier global_database_id
```

Compruebe que el clúster de la base de datos Aurora esté listo. Cuando se muestra la respuesta del siguiente comando `"Status": "available"` para el clúster de Aurora base de datos, puede continuar.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

3. Cree la instancia de base de datos para el clúster de base de datos Aurora principal. Para ello, utilice el comando de CLI `create-db-instance`.

   Pase el nombre de su clúster de base de datos de Aurora con el parámetro `--db-cluster-identifier`.

   No necesita pasar los parámetros `--master-username` y `--master-user-password` en el comando, ya que los obtiene del clúster de base de datos Aurora.

   Para el `--db-instance-class`, puede usar solo aquellos de clases optimizadas para memoria, como `db.r5.large`. Recomendamos utilizar una clase de instancia db.r5 o superior. Para obtener información acerca de estas clases, consulte Tipos de clase de instancia de base de datos.

   Para Linux, macOS o:Unix

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-postgresql \
  --engine-version version \
  --region primary_region
```

En:Windows

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --region primary_region
```

4. Compruebe el estado de la instancia Aurora de base de datos antes de continuar.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Si la respuesta muestra que el estado de la instancia de base de datos de Aurora está `available`, puede crear otra instancia de base de datos de Aurora para su clúster de base de datos principal.

5. Para crear una réplica Aurora para clúster de base de datos Aurora, utilice el comando `create-db-instance` CLI.

Para Linux, macOS o:Unix

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier replica_db_instance_id \
  --engine aurora-postgresql
```

En:Windows

```
aws rds create-db-instance ^
```

```
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier replica_db_instance_id ^
  --engine aurora-postgresql
```

Cuando la instancia de base de datos está disponible, la reproducción comienza desde el nodo de escritor a la réplica. Antes de continuar, compruebe que la instancia de base de datos esté disponible con el comando `describe-db-instances` CLI.

Su base de datos Aurora global existe, pero solo tiene su región principal con un clúster de base de datos Aurora compuesto por una instancia de base de datos de escritor y una réplica Aurora. Ahora puede agregar un clúster de base de datos Aurora de solo lectura en una región diferente para completar la base de datos Aurora global. Para ello, siga los pasos que se indican en Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

API de RDS

Para crear una base de datos de Aurora global con la API de RDS, ejecute la operación CreateGlobalCluster.

Incorporación de una Región de AWS a una base de datos global de Amazon Aurora

Puede usar el siguiente procedimiento para añadir un clúster secundario adicional a una base de datos global existente. También puede crear una base de datos global a partir de un clúster de base de datos de Aurora independiente mediante este procedimiento para añadir la primera región de AWS secundaria.

Una base de datos global de Aurora necesita al menos un clúster de base de datos de Aurora secundario en una Región de AWS diferente que la del clúster principal de base de datos de Aurora. Puede adjuntar hasta 10 clústeres de base de datos secundarios a la base de datos global de Aurora. Repita el siguiente procedimiento para cada clúster de base de datos secundario nuevo. Para cada clúster de base de datos secundaria que agregue a la base de datos de Aurora global, reduzca el número de réplicas de Aurora permitidas en el clúster de base de datos principal a una.

Por ejemplo, si la base de datos global de Aurora tiene 10 regiones secundarias, el clúster de la base de datos principal solo puede tener 5 (en lugar de 15) réplicas de Aurora. Para obtener más información, consulte Requisitos de configuración de una base de datos Amazon Aurora global.

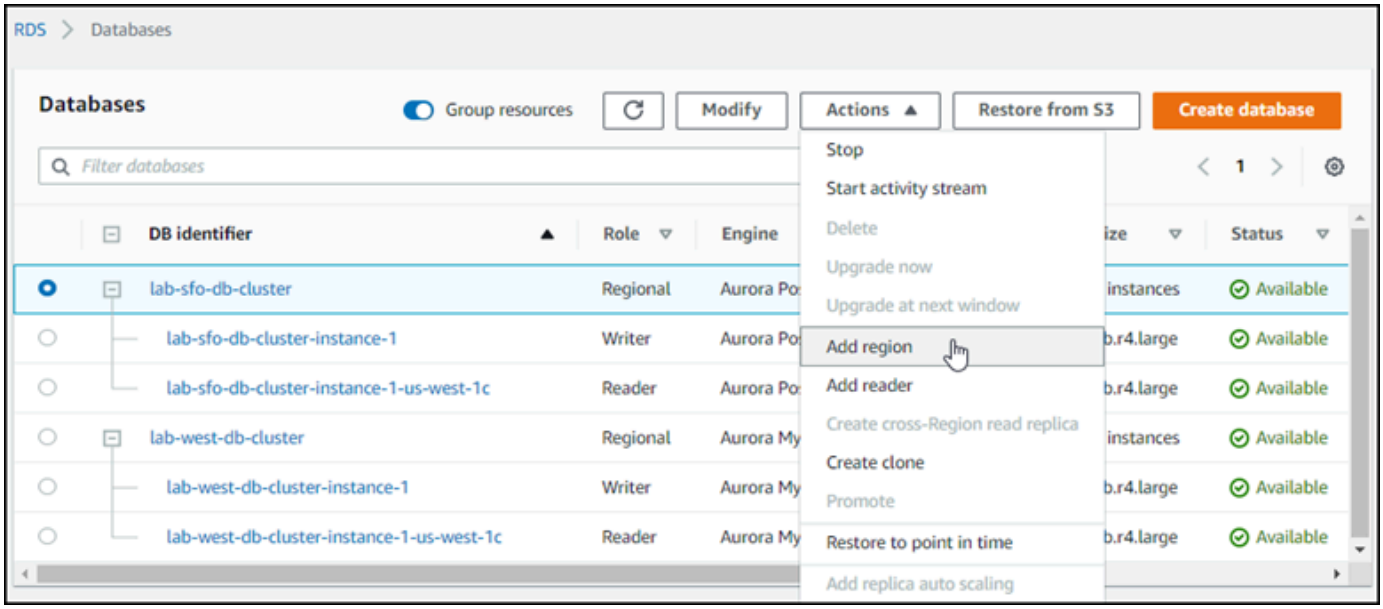
El número de réplicas de Aurora (instancias de lectura) en el clúster de base de datos principal determina el número de clústeres de base de datos secundarias que puede agregar. El número total de instancias de lectura en el clúster de base de datos principal más el número de clústeres secundarios no puede ser mayor de 15. Por ejemplo, si tiene 14 instancias de lectura en el clúster de base de datos principal y 1 clúster secundario, no puede agregar otro clúster secundario a la base de datos global.

> **Note**
>
> En el caso de la versión 3 de Aurora MySQL, cuando cree un clúster secundario, asegúrese de que el valor de `lower_case_table_names` coincida con el valor del clúster principal. Esta configuración es un parámetro de la base de datos que afecta la forma en que el servidor gestiona la distinción entre mayúsculas y minúsculas del identificador. Para obtener más información acerca de los parámetros de la base de datos, consulte Grupos de parámetros para Amazon Aurora.
>
> Se recomienda que, al crear un clúster secundario, utilice la misma versión del motor de base de datos para el principal y el secundario. Si es necesario, actualice la versión principal para que tenga la misma versión que la secundaria. Para obtener más información, consulte Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones.

Consola

Para agregar una Región de AWS a una base de datos global de Aurora

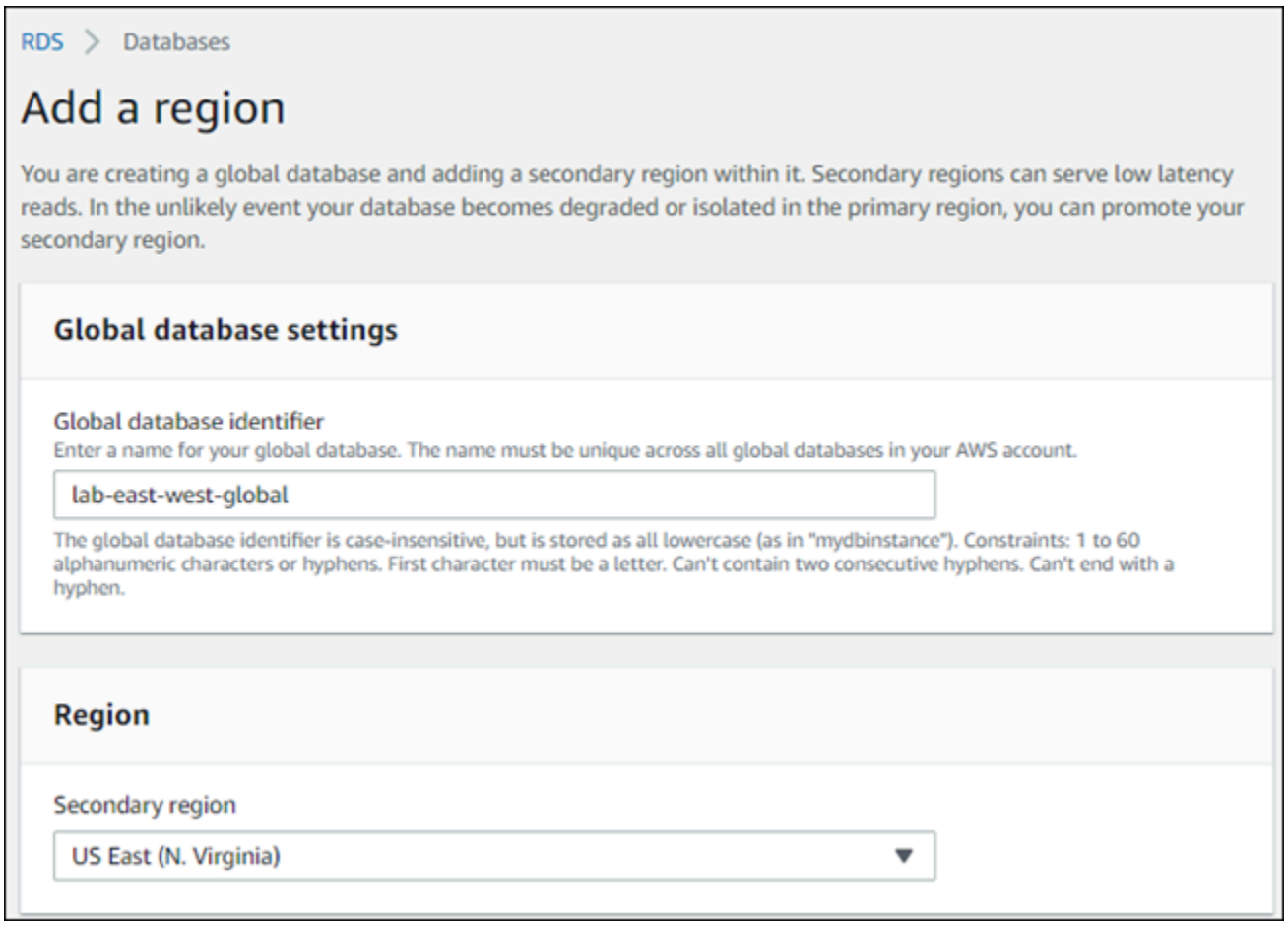
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la Consola de administración de AWS, elija Databases (Bases de datos).
3. Elija la base de datos Aurora global que necesita un clúster de base de datos Aurora secundario. Asegúrese de que el clúster de Aurora base de datos principal es `Available`.
4. En Acciones, elija Agregar región de AWS.



5. En la página Add a region (Agregar una región), seleccione la Región de AWS secundaria.

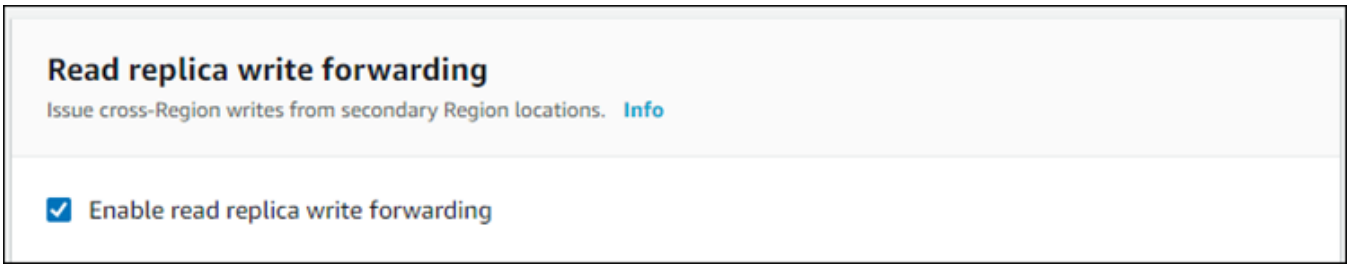
   No puede elegir una Región de AWS que ya tenga un clúster secundario de base de datos de Aurora para la misma base de datos global de Aurora. Además, no puede ser la misma región que el clúster principal de la base de datos de Aurora.

> **Note**
>
> Las bases de datos globales de Babelfish para Aurora PostgreSQL solo funcionan en regiones secundarias si los parámetros que controlan las preferencias de Babelfish están activados en esas regiones. Para obtener más información, consulte Configuración del grupo de parámetros del clúster de base de datos para Babelfish



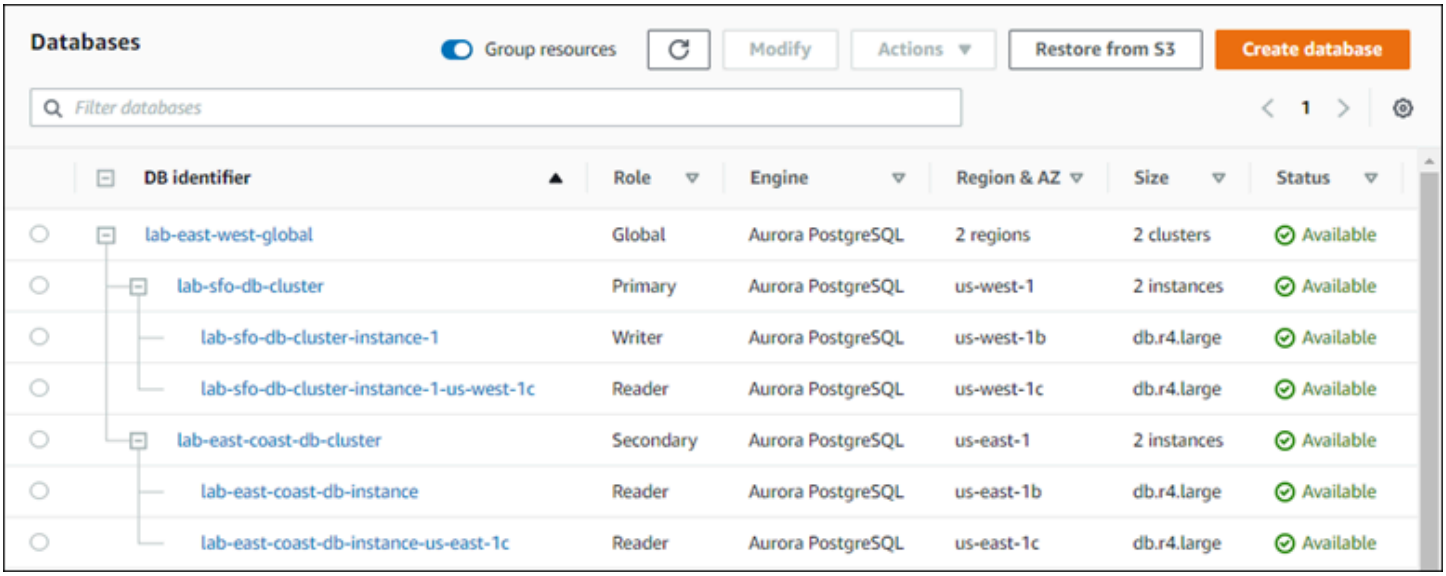
6. Complete los campos restantes para el clúster secundario de Aurora en la nueva región de AWS. Estas son las mismas opciones de configuración que para cualquier instancia de clúster de base de datos de Aurora, excepto la siguiente opción solo para bases de datos globales de Aurora basadas en Aurora MySQL–:

   - Habilitar el reenvío de escritura de réplica de lectura – Esta configuración opcional permite que los clústeres secundarios de la base de datos global de Aurora reenvíen operaciones de escritura al clúster principal. Para obtener más información, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.



7. Elija Agregar región de AWS.

Después de terminar de agregar la región a la base de datos de Aurora global, puede verla en la lista de bases de datos en la Consola de administración de AWS como se muestra en la captura de pantalla.



AWS CLI

Para agregar una Región de AWS secundaria a una base de datos global de Aurora

Para añadir un clúster secundario a la base de datos global mediante la CLI, debe disponer ya del objeto contenedor del clúster global. Si aún no ha ejecutado el comando `create-global-cluster`, consulte el procedimiento de la CLI en Creación de una base de datos global de Amazon Aurora.

1. Utilice el comando `create-db-cluster` CLI con el nombre (`--global-cluster-identifier`) de la base de datos Aurora global. Para otros parámetros, haga lo siguiente:
2. Para `--region`, elija una Región de AWS diferente a la de la región de Aurora principal.
3. Elija valores específicos para los parámetros `--engine` y `--engine-version`. Estos valores son los mismos que los del clúster principal de base de datos de Aurora de la base de datos global de Aurora.
4. Para un clúster cifrado, especifique la Región de AWS principal como `--source-region` para cifrado.

En el ejemplo siguiente se crea un nuevo clúster de base de datos Aurora y se adjunta a una base de datos Aurora global como clúster de base de datos Aurora secundario de solo lectura. En el último paso, se agrega una instancia Aurora de base de datos al nuevo clúster de Aurora base de datos.

Para Linux, macOS o Unix:

```
aws rds --region secondary_region \
  create-db-cluster \
    --db-cluster-identifier secondary_cluster_id \
    --global-cluster-identifier global_database_id \
    --engine aurora-mysql | aurora-postgresql \
    --engine-version version

aws rds --region secondary_region \
  create-db-instance \
    --db-instance-class instance_class \
    --db-cluster-identifier secondary_cluster_id \
    --db-instance-identifier db_instance_id \
    --engine aurora-mysql | aurora-postgresql
```

Para Windows:

```
aws rds --region secondary_region ^
  create-db-cluster ^
    --db-cluster-identifier secondary_cluster_id ^
    --global-cluster-identifier global_database_id_id ^
    --engine aurora-mysql | aurora-postgresql ^
    --engine-version version

aws rds --region secondary_region ^
  create-db-instance ^
    --db-instance-class instance_class ^
    --db-cluster-identifier secondary_cluster_id ^
    --db-instance-identifier db_instance_id ^
    --engine aurora-mysql | aurora-postgresql
```

API de RDS

Para agregar una nueva Región de AWS a una base de datos global de Aurora con la API de RDS, ejecute la operación CreateDBCluster. Especifique el identificador de la base de datos global existente utilizando el parámetro `GlobalClusterIdentifier`.

Creación de un clúster de base de datos de Aurora sin pantalla en una región secundaria

Aunque una base de datos global de Aurora requiere al menos un clúster de base de datos secundaria de Aurora en una Región de AWS diferente a la principal, puede utilizar una configuración headless (sin pantalla) para el clúster secundario. Un clúster secundario de base de datos de Aurora sin pantalla es uno sin una instancia de base de datos. Este tipo de configuración puede reducir los gastos para una base de datos global de Aurora. En un clúster de base de datos de Aurora, se desacoplan la informática y el almacenamiento. Sin la instancia de base de datos, no se le cobrará por la informática, solo por almacenamiento. Si está configurado correctamente, el volumen de almacenamiento de un clúster secundario sin pantalla se mantiene sincronizado con el clúster principal de la base de datos de Aurora.

Agregue el clúster secundario como lo hace normalmente al crear una base de datos global de Aurora. Si va a crear todos los clústeres de la base de datos global, siga el procedimiento descrito en Creación de una base de datos global de Amazon Aurora. Si ya tiene un clúster de base de datos para usarlo como clúster principal, siga el procedimiento descrito en Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.

Después de que el clúster de base de datos de Aurora principal comience la replicación en el secundario, se elimina la instancia de base de datos de Aurora de solo lectura del clúster de base de datos de Aurora secundario. Ahora, este clúster secundario se considera “sin pantalla” porque ya no tiene una instancia de base de datos. Aunque no haya ninguna instancia de base de datos en el clúster secundario, Aurora mantiene el volumen de almacenamiento sincronizado con el clúster de base de datos de Aurora principal.

> **Warning**
>
> Con Aurora PostgreSQL, para crear un clúster sin pantalla en una Región de AWS secundaria, use la AWS CLI o la API de ARS para agregar la Región de AWS secundaria. Omita el paso a fin de crear la instancia de base de datos del lector para el clúster secundario. Actualmente, la creación de un clúster sin pantalla no se admite en la consola de RDS. Para obtener información sobre los procedimientos de la CLI y la API que se utilizarán, consulte Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.
>
> Si su base de datos global utiliza una versión del motor de Aurora PostgreSQL anterior a 13.4, 12.8 u 11.13, crear una instancia de base de datos del lector en una región secundaria y, posteriormente, eliminarla podría provocar un problema de vacío de Aurora

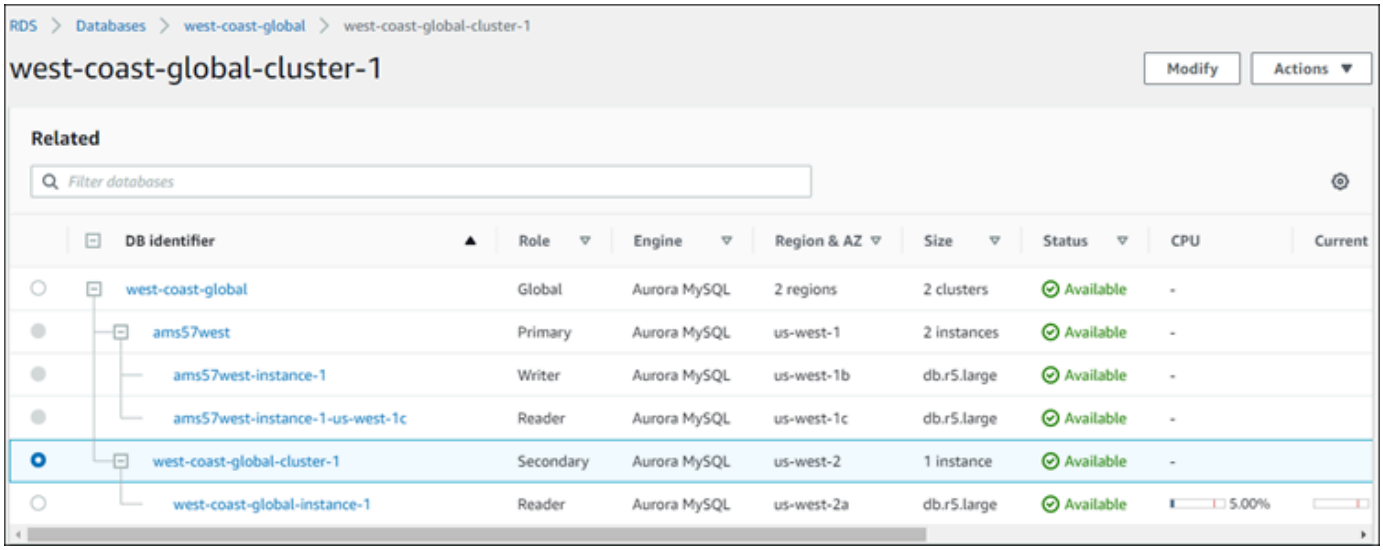
PostgreSQL en la instancia de base de datos del escritor de la región principal. Si se produce este problema, reinicie la instancia de base de datos del escritor de la región principal después de eliminar la instancia de base de datos del lector de la región secundaria.

Para agregar un clúster secundario de base de datos de Aurora sin pantalla a la base de datos global de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación de la Consola de administración de AWS, elija Databases (Bases de datos).
3. Elija la base de datos Aurora global que necesita un clúster de base de datos Aurora secundario. Asegúrese de que el clúster de Aurora base de datos principal es `Available`.
4. En Acciones, elija Agregar región de AWS.
5. En la página Add a region (Agregar una región), seleccione la Región de AWS secundaria.

   No puede elegir una Región de AWS que ya tenga un clúster secundario de base de datos de Aurora para la misma base de datos global de Aurora. Además, no puede ser la misma región que el clúster principal de la base de datos de Aurora.
6. Complete los campos restantes para el clúster secundario de Aurora en la nueva Región de AWS. Estas son las mismas opciones de configuración que para cualquier instancia de clúster de base de datos de Aurora.

   Para una base de datos global de Aurora global basada en Aurora MySQL–, omita la opción Enable read replica write forwarding (Habilitar reenvío de escritura de réplica de lectura). Esta opción no tiene ninguna función después de eliminar la instancia del lector.
7. Elija Agregar región de AWS. Después de terminar de agregar la región a la base de datos de Aurora global, puede verla en la lista de bases de datos en la Consola de administración de AWS como se muestra en la captura de pantalla.

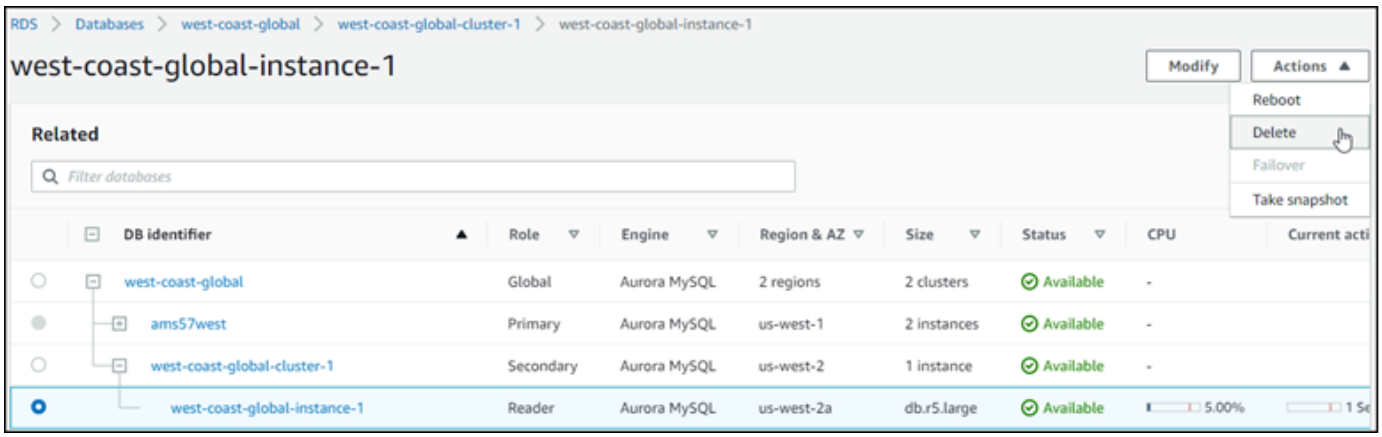


8. Verifique el estado del clúster secundario de la base de datos de Aurora y su instancia de lector antes de continuar, mediante la Consola de administración de AWS o la AWS CLI. Por ejemplo:

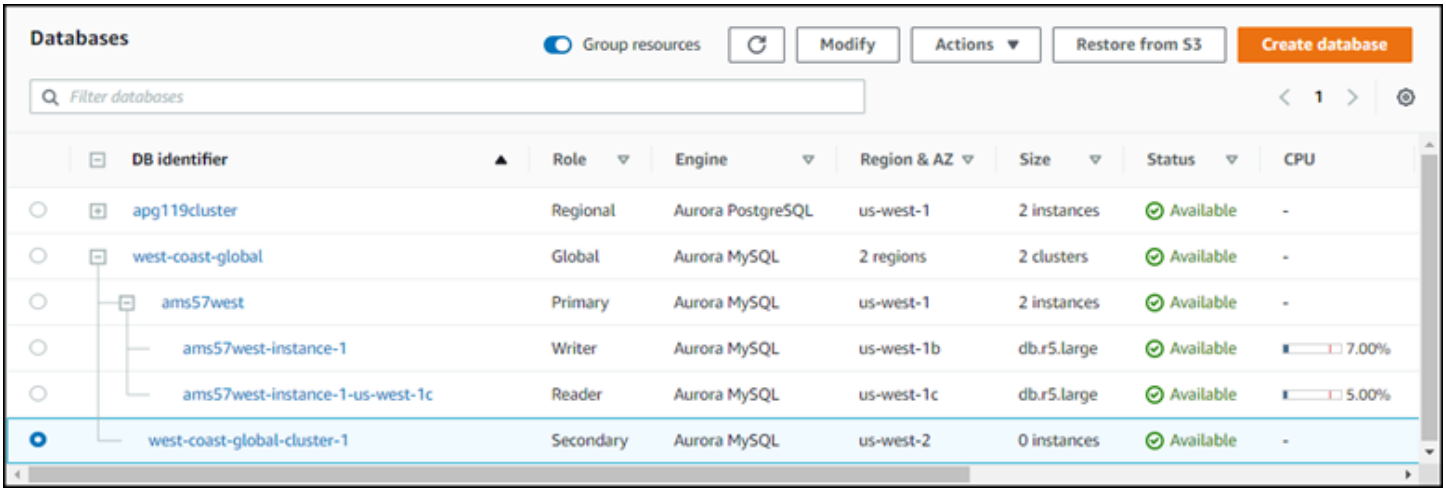
```
$ aws rds describe-db-clusters --db-cluster-identifier secondary-cluster-id --query
 '*[].[Status]' --output text
```

   El estado de un clúster secundario de la base de datos de Aurora recién agregado puede tardar varios minutos en cambiar de `creating` a `available`. Cuando el clúster de la base de datos de Aurora se encuentra disponible, puede eliminar la instancia de lector.

9. Seleccione la instancia de lector en el clúster secundario de la base de datos de Aurora y, a continuación, elija Delete (Eliminar).



Después de eliminar la instancia de lector, el clúster secundario sigue siendo parte de la base de datos global de Aurora. No tiene ninguna instancia asociada con él, como se muestra a continuación.



Puede utilizar este clúster secundario de base de datos de Aurora sin pantalla para recuperar de forma manual la base de datos global de Amazon Aurora de una interrupción no planificada en la Región de AWS principal, si se produce una interrupción de este tipo.

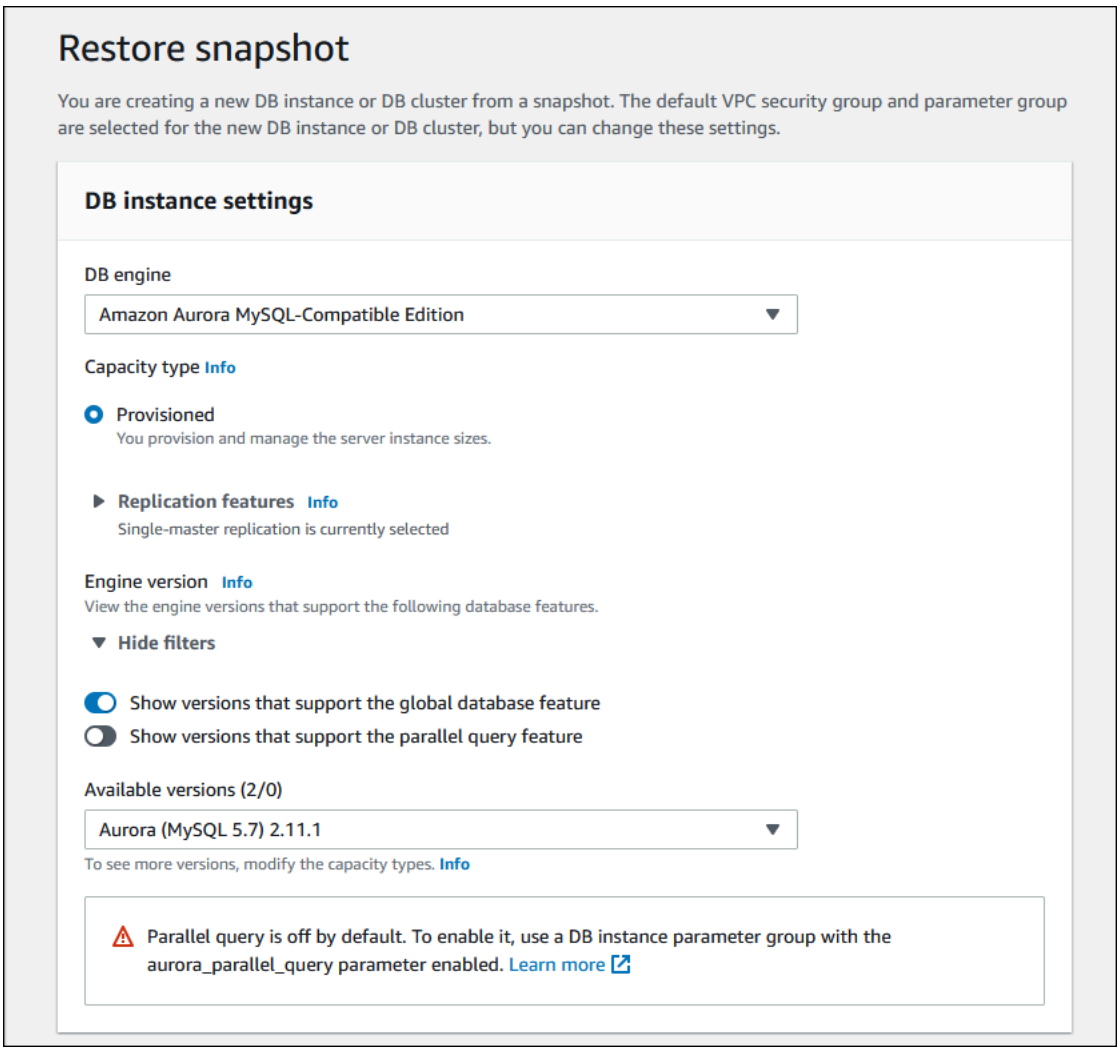
Creación de una base de datos global de Amazon Aurora a partir de una instantánea de Aurora o Amazon RDS

Puede restaurar una instantánea de un clúster de base de datos Aurora o desde una instancia de base de datos Amazon RDS para utilizarla como punto de partida de la base de datos global Aurora. Restaurar la instantánea y crear un nuevo clúster de base de datos Aurora aprovisionado al mismo tiempo. Luego, agrega otra Región de AWS al clúster de base de datos restaurado y lo convierte en una base de datos global de Aurora. Cualquier clúster de base de datos de Aurora que cree utilizando una instantánea de esta manera se convierte en el clúster principal de la base de datos global de Aurora.

La instantánea que utilice puede ser de un clúster de base de datos `provisioned` o de un clúster de `serverless` Aurora base de datos.

Durante el proceso de restauración, elija el mismo tipo de motor de base de datos que la instantánea. Por ejemplo, supongamos que desea restaurar una instantánea realizada desde un clúster de base de datos de Aurora Serverless que ejecute Aurora PostgreSQL. En este caso, se crea un clúster de base de datos de Aurora PostgreSQL utilizando el mismo motor de base de datos de Aurora y la misma versión.

El clúster de base de datos restaurado asume el rol de clúster principal para la base de datos global de Aurora cuando se añade una Región de AWS. Todos los datos contenidos en este clúster principal se replican en cualquier clúster secundario que agregue a la base de datos Aurora global.



Administración de una base de datos global de Amazon Aurora

Puede realizar la mayor parte de las operaciones de administración en los clústeres individuales que componen una base de datos global de Aurora. Cuando selecciona Group related resources (Recursos relacionados con grupos) en la página Databases (Bases de datos) de la consola, verá el clúster principal y el secundario agrupados bajo la base de datos global asociada. Para encontrar las Regiones de AWS donde se ejecutan los clústeres de una base de datos global, el motor, el identificador y la versión de la base de datos de Aurora, use la pestaña Configuration (Configuración).

Los procesos de conmutación por error de la base de datos entre regiones solo están disponibles para las bases de datos globales de Aurora, no para un solo clúster de base de datos de Aurora. Para obtener más información, consulte Uso de la transición o la conmutación por error en la base de datos global de Amazon Aurora.

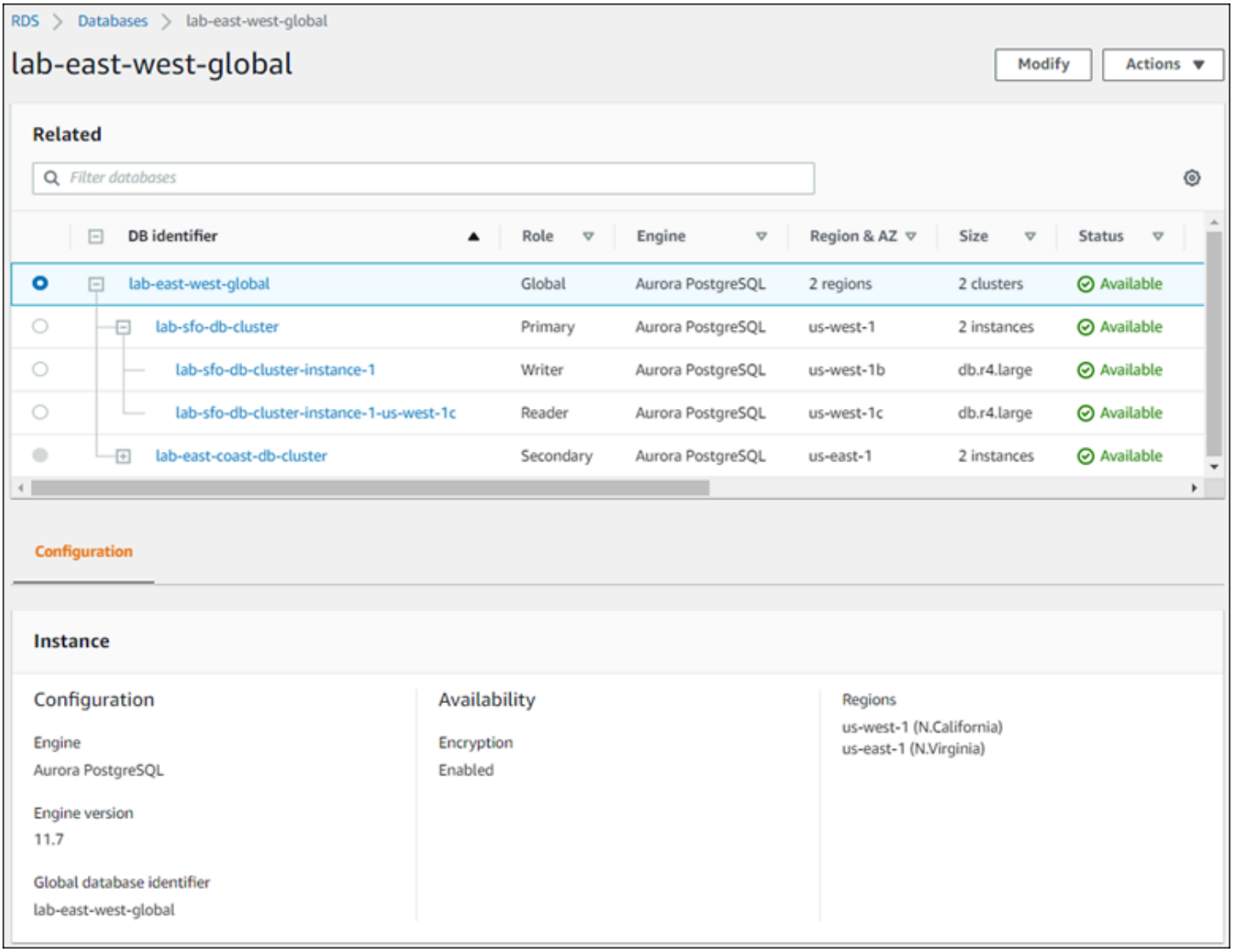
Para recuperar una base de datos global de Aurora de una interrupción no planificada en la región principal, consulte Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada.

Temas

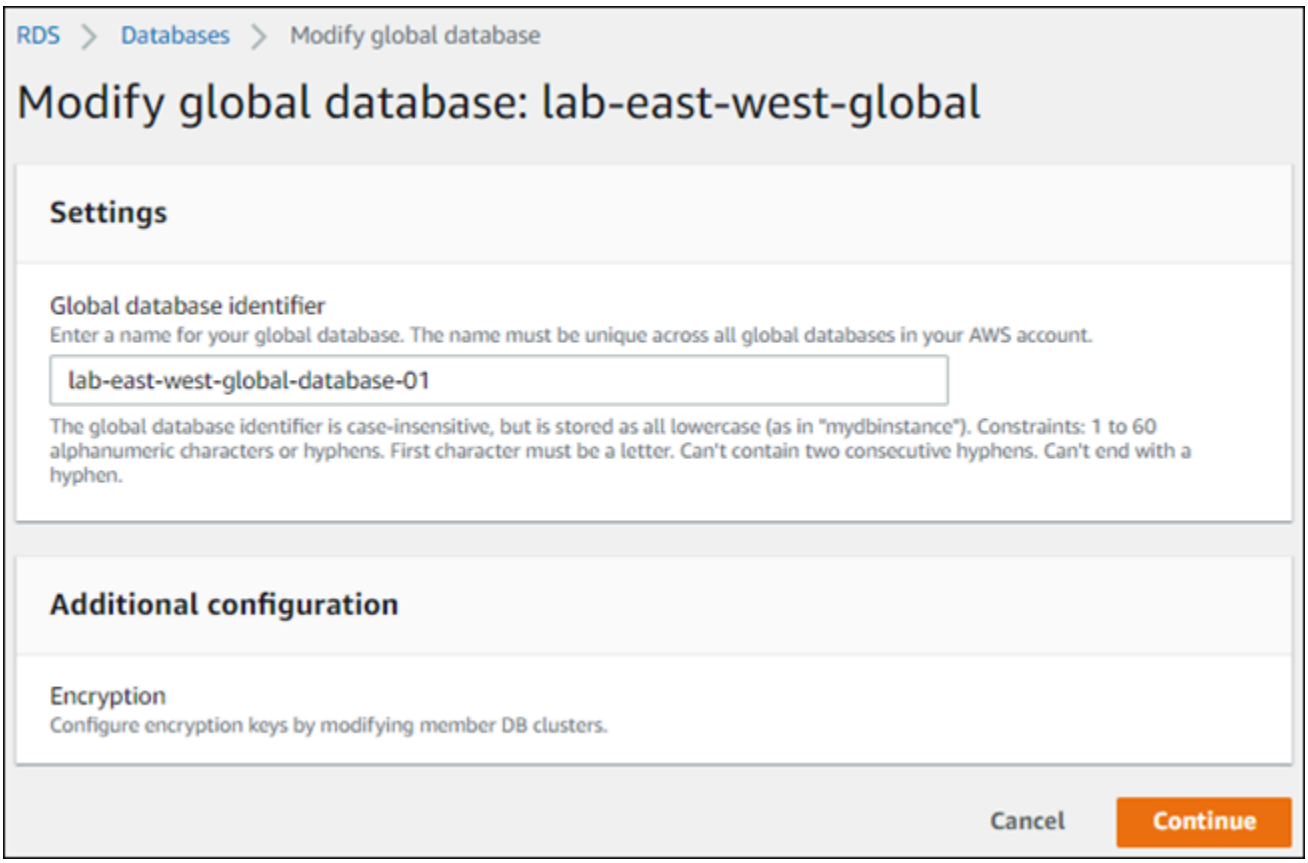
- Modificación de una base de datos global de Amazon Aurora
- Modificación de parámetros para una base de datos Aurora global
- Eliminación de un clúster de una base de datos global de Amazon Aurora
- Eliminación de una base de datos global de Amazon Aurora
- Etiquetado de recursos de la base de datos global de Amazon Aurora

Modificación de una base de datos global de Amazon Aurora

La página Databases (Bases de datos) en la Consola de administración de AWS muestra todas las bases de datos globales de Aurora, que muestran el clúster principal y los clústeres secundarios de cada una de ellas. La base de datos global de Aurora tiene sus propias opciones de configuración. Específicamente, tiene Regiones de AWS asociadas con los clústeres principales y secundarios, como se muestra en la siguiente captura de pantalla.



Al realizar cambios en la base de datos Aurora global, tiene la oportunidad de cancelar los cambios, como se muestra en la siguiente captura de pantalla.



Al seleccionar Continue (Continuar), confirma los cambios.

Modificación de parámetros para una base de datos Aurora global

Puede configurar los grupos de parámetros independientemente para cada clúster de Aurora de base de datos Aurora dentro de la base de datos global de Aurora. La mayoría de parámetros funcionan igual que para otros tipos de clústeres de Aurora. Se recomienda mantener la configuración coherente entre todos los clústeres de una base de datos global. Esto ayuda a evitar cambios de comportamiento inesperados si se promueve un clúster secundario para que sea el principal.

Por ejemplo, utilice la misma configuración de zonas horarias y conjuntos de caracteres para evitar un comportamiento incoherente si un clúster diferente asume la función del clúster principal.

Los ajustes de configuración `aurora_enable_repl_bin_log_filtering` y `aurora_enable_replica_log_compression` no tienen efecto.

Eliminación de un clúster de una base de datos global de Amazon Aurora

Puede eliminar clústeres de base de datos Aurora de la base de datos Aurora global por varias razones diferentes. Por ejemplo, es posible que desee quitar un clúster de base de datos Aurora de una base de datos Aurora global si el clúster principal se degrada o se aísla. A continuación, se convierte en un clúster de base de datos de Aurora aprovisionado independiente que podría utilizarse para crear una nueva base de datos global de Aurora. Para obtener más información, consulte Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada.

También puede querer quitar clústeres de base de datos de Aurora porque desea eliminar una base de datos global de Aurora que ya no necesite. No puede eliminar la base de datos global de Aurora hasta después de eliminar (desasociar) todos los clústeres de base de datos de Aurora asociados y deje el principal para lo último. Para obtener más información, consulte Eliminación de una base de datos global de Amazon Aurora.

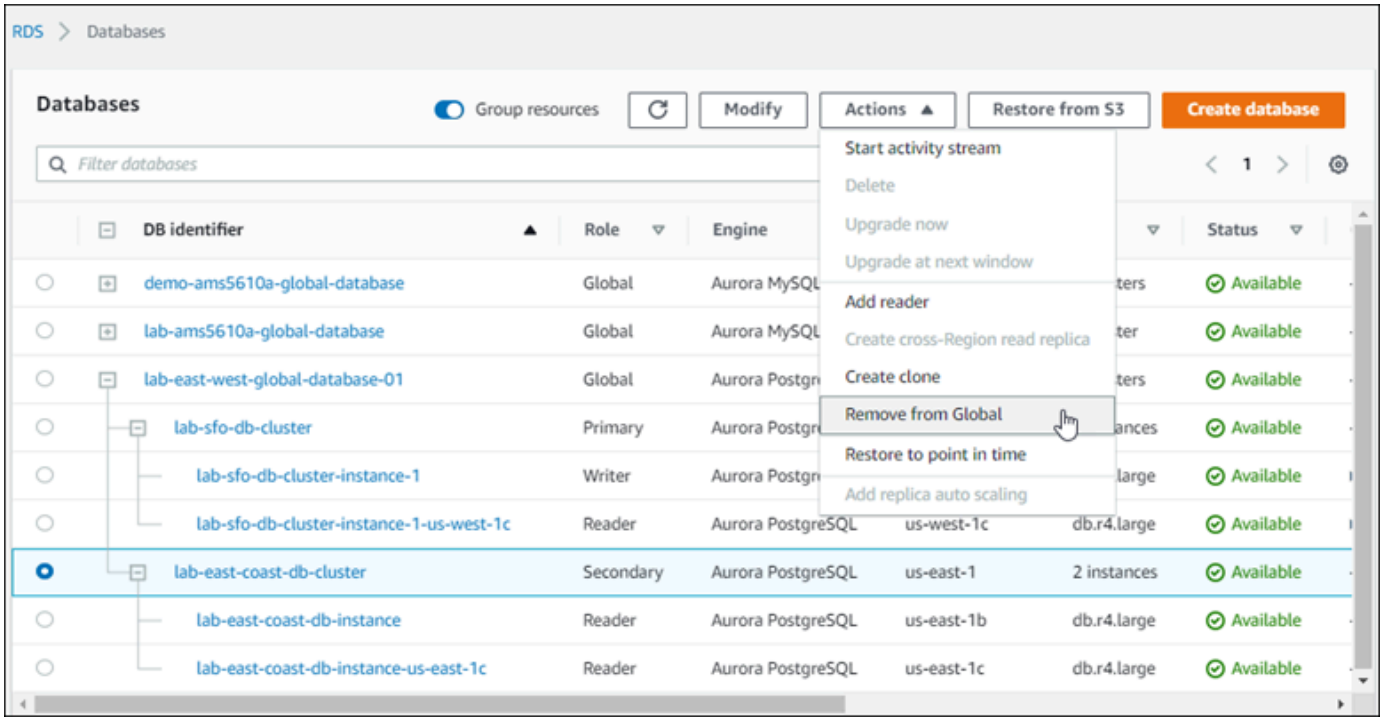
Cuando un clúster de base de datos de Aurora se desasocia de la base de datos global de Aurora, ya no se sincroniza con el principal. Se convierte en un clúster de base de datos de Aurora aprovisionado independiente con capacidades completas de lectura/escritura.

Puede quitar clústeres de base de datos Aurora de la base de datos de Aurora global mediante Consola de administración de AWS, la AWS CLI o la API de RDS.

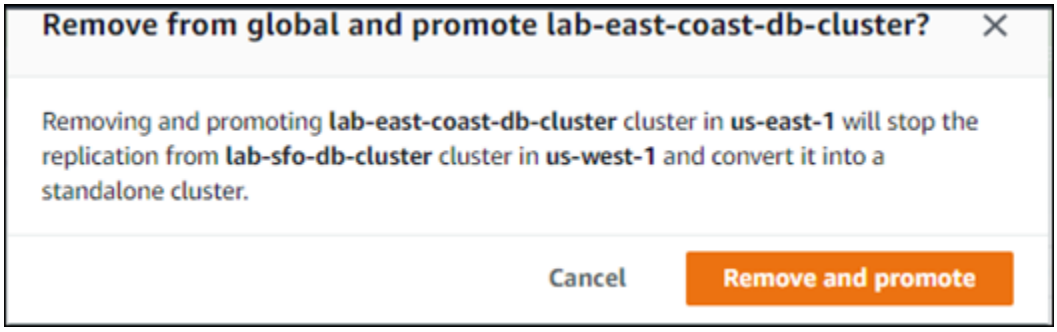
Consola

Para quitar un clúster de Aurora de una base de datos global de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione el clúster en la página Databases (Bases de datos).
3. En Actions (Acciones), elija Remove from Global (Eliminar desde global).

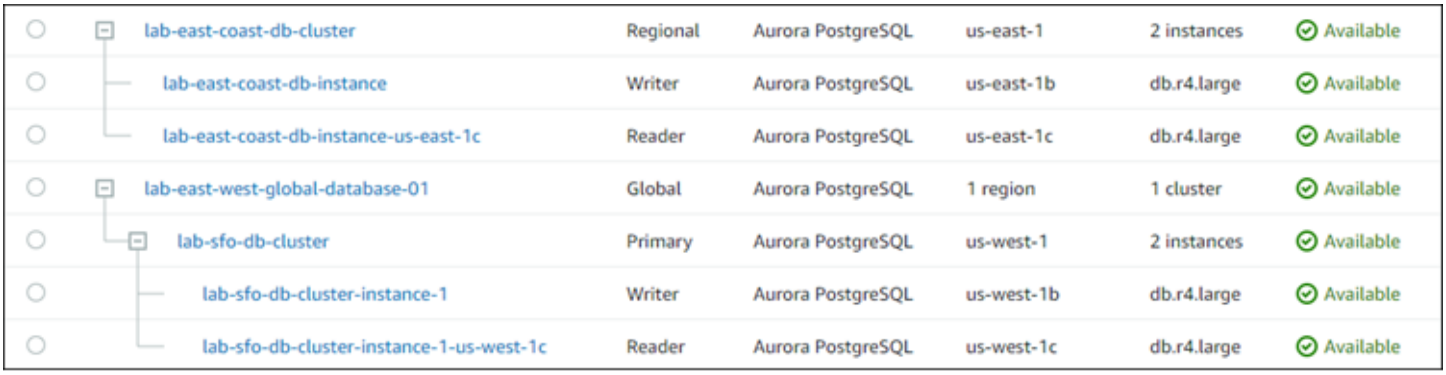


Abra un mensaje para confirmar que desea separar el secundario de la base de datos Aurora global.



4. Elija Remove and promote (Eliminar y promover) para quitar el clúster de la base de datos global.

El clúster de base de datos Aurora ya no sirve como secundario en la base de datos Aurora global y ya no está sincronizado con el clúster de base de datos principal. Es un clúster de base de datos Aurora independiente con capacidad completa de lectura/escritura.



Tras eliminar o borrar todos los clústeres secundarios, podrá eliminar el clúster principal del mismo modo. No puede separar (quitar) el clúster de base de datos Aurora principal de una base de datos Aurora global hasta después de quitar todos los clústeres secundarios.

La base de datos global de Aurora puede permanecer en la lista Databases (Bases de datos), con cero regiones y AZ. Puede eliminar si ya no desea utilizar esta base de datos Aurora global. Para obtener más información, consulte Eliminación de una base de datos global de Amazon Aurora.

AWS CLI

Para eliminar un clúster Aurora de una base de datos global de Aurora, ejecute el comando CLI remove-from-global-cluster con los siguientes parámetros:

- `--global-cluster-identifier` – El nombre (identificador) de la base de datos Aurora global.
- `--db-cluster-identifier` – Nombre de cada clúster de base de datos Aurora que se va a quitar de la base de datos Aurora global. Quite todos los clústeres de base de datos Aurora secundarios antes de quitar el primario.

Los siguientes comandos eliminan un clúster secundario y, después, el clúster primario de una base de datos global de Aurora.

Para Linux, macOS o:Unix

```
aws rds --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_database_id

aws rds --region primary_region \
  remove-from-global-cluster \
```

```
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_database_id
```

Repita el comando `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* para cada Región de AWS secundaria de la base de datos global de Aurora.

Para Windows:

```
aws rds --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_database_id

aws rds --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_database_id
```

Repita el comando `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* para cada Región de AWS secundaria de la base de datos global de Aurora.

API de RDS

Para eliminar un clúster de Aurora de una base de datos global de Aurora con la API de RDS, ejecute la acción RemoveFromGlobalCluster.

Eliminación de una base de datos global de Amazon Aurora

Dado que una base de datos Aurora global suele contener datos empresariales esenciales, no puede eliminar la base de datos global y los clústeres asociados en un único paso. Para completar la eliminación de una base de datos global de Aurora, haga lo siguiente:

- Elimine todos los clústeres de base de datos secundarios de la base de datos Aurora global. Cada clúster se convierte en un clúster de base de datos Aurora independiente. Para saber cómo hacerlo, consulte Eliminación de un clúster de una base de datos global de Amazon Aurora.
- En cada clúster de base de datos Aurora independiente, elimine todas las réplicas Aurora.
- Elimine el clúster secundario de la base de datos global de Aurora. Esto se convierte en un clúster de base de datos Aurora independiente.

- Desde el clúster principal de la base de datos de Aurora, primero elimine todas las réplicas Aurora y, a continuación, elimine la instancia de base de datos del escritor.

La eliminación de la instancia de escritor del clúster de base de datos Aurora recién independiente también normalmente elimina el clúster de base de datos Aurora y la base de datos Aurora global.

Para obtener más información general, consulte Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora.

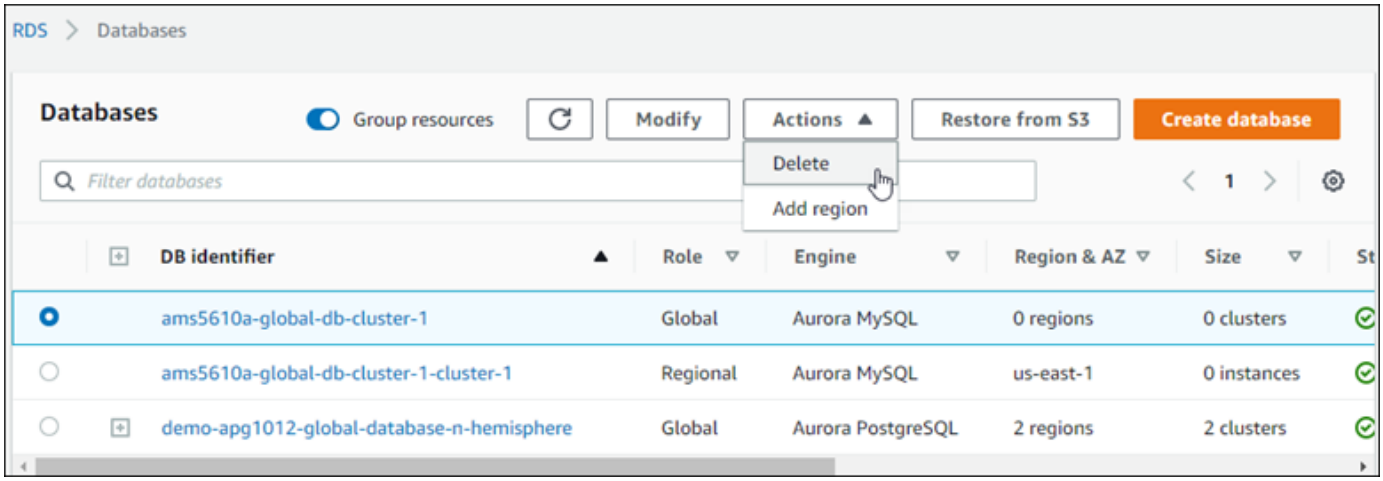
Para eliminar una base de datos de Aurora global, puede utilizar la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para eliminar una base de datos global de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos y busque la base de datos Aurora global que desea eliminar en el listado.
3. Confirme que todos los demás clústeres se han borrado de la base de datos global de Aurora. La base de datos global de Aurora debe mostrar 0 regiones y AZ, y tener un tamaño de 0 clústeres.

   Si la base de datos Aurora global contiene clústeres de base de datos Aurora, no puede eliminarla. Si es necesario, desconecte los clústeres de base de datos Aurora principal y secundaria de la base de datos Aurora global. Para obtener más información, consulte Eliminación de un clúster de una base de datos global de Amazon Aurora.
4. Elija la base de datos de Aurora global en la lista y, a continuación, elija Eliminar en el menú Acciones.



AWS CLI

Para eliminar una base de datos global de Aurora, ejecute el comando delete-global-cluster de la CLI con el nombre de la Región de AWS y el identificador de base de datos global de Aurora, como se muestra en el siguiente ejemplo.

Para Linux, macOS o:Unix

```
aws rds --region primary_region delete-global-cluster \
   --global-cluster-identifier global_database_id
```

En:Windows

```
aws rds --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_database_id
```

API de RDS

Para eliminar un clúster que forme parte de una base de datos global de Aurora con la API de RDS, ejecute la operación DeleteGlobalCluster.

Etiquetado de recursos de la base de datos global de Amazon Aurora

Con la característica de la base de datos global de Aurora, puede aplicar etiquetas de RDS a los recursos en distintos niveles de una base de datos global. Si no está familiarizado con el uso de las etiquetas de AWS o los recursos de Aurora, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS antes de aplicar etiquetas en su base de datos global.

> **Note**
>
> Como los datos de las etiquetas de los procesos de AWS forman parte de sus mecanismos de elaboración de informes de costes, no incluya ningún dato confidencial ni información de identificación personal (PII) en los nombres o valores de las etiquetas.

Puede aplicar etiquetas a los siguientes tipos de recursos de una base de datos global:

- El objeto contenedor de toda la base de datos global. Este recurso se conoce como clúster global.

  Después de crear el clúster global mediante la operación Agregar región de AWS en la consola, puede agregar etiquetas mediante la página de detalles del clúster global. En la pestaña Etiquetas de la página de detalles del clúster global, puede agregar, eliminar o modificar las etiquetas y sus valores asociados seleccionando Administrar etiquetas.

  Con la API de RDS y la AWS CLI, puede agregar etiquetas al clúster global al mismo tiempo que lo crea. También puede agregar, eliminar o modificar etiquetas para un clúster global existente.
- El clúster principal. Aquí se utilizan los mismos procedimientos para trabajar con etiquetas que para los clústeres de Aurora independientes. Puede configurar las etiquetas antes de convertir el clúster de Aurora original en una base de datos global. Puede agregar, eliminar o modificar las etiquetas y sus valores asociados seleccionando Administrar etiquetas en la pestaña Etiquetas de la página de detalles del clúster de base de datos global.
- Cualquier clúster secundario. Aquí se utilizan los mismos procedimientos para trabajar con etiquetas que para los clústeres de Aurora independientes. Puede configurar las etiquetas al mismo tiempo que crea un clúster de Aurora secundario mediante la acción Agregar región de AWS en la consola. Puede agregar, eliminar o modificar las etiquetas y sus valores asociados seleccionando Administrar etiquetas en la pestaña Etiquetas de la página de detalles del clúster de base de datos global.
- Instancias de bases de datos individuales dentro de los clústeres principal o secundario. Aquí se utilizan los mismos procedimientos para trabajar con etiquetas que para las instancias de base de datos de Aurora o RDS. Puede configurar las etiquetas al mismo tiempo que agrega una nueva instancia de base de datos al clúster de Aurora secundario mediante la acción Agregar lector en la consola. Puede agregar, eliminar o modificar las etiquetas y sus valores asociados seleccionando Administrar etiquetas en la pestaña Etiquetas de la página de detalles de la instancia de base de datos.

Estos son algunos ejemplos de los tipos de etiquetas que puede asignar en una base de datos global:

- Puede agregar etiquetas al clúster global para registrar información general sobre su aplicación, como identificadores anónimos que representen a los propietarios y contactos de su organización. Puede utilizar etiquetas para representar las propiedades de la aplicación que utiliza la base de datos global.
- Puede agregar etiquetas al clúster principal y a los clústeres secundarios para realizar un seguimiento de los costos de la aplicación a nivel de la región de AWS. Para obtener información acerca del procedimiento, consulte Funcionamiento de la facturación de AWS con etiquetas en Amazon RDS.
- Puede agregar etiquetas a instancias de base de datos específicas con los clústeres de Aurora para indicar su propósito especial. Por ejemplo, dentro del clúster principal, es posible que tenga una instancia del lector con una prioridad de conmutación por error baja que se utilice exclusivamente para la generación de informes. Una etiqueta puede distinguir esta instancia de base de datos de uso especial de otras instancias dedicadas a la alta disponibilidad dentro del clúster principal.
- Puede utilizar etiquetas en todos los niveles de los recursos de su base de datos global para controlar el acceso mediante las políticas de IAM. Para obtener más información, consulte Control del acceso a recursos de AWS en la Guía del usuario de AWS Identity and Access Management.

  > **Tip**
  >
  > En la Consola de administración de AWS, agregue etiquetas al contenedor del clúster global como un paso independiente después de crearlo. Si desea evitar cualquier intervalo de tiempo en el que el clúster global exista sin etiquetas de control de acceso, puede aplicar las etiquetas durante la operación `CreateGlobalCluster` creando ese recurso mediante la AWS CLI, la API de RDS o una plantilla de CloudFormation.

- Puede usar etiquetas a nivel de clúster, o para el clúster global, para registrar información sobre el control de calidad y las pruebas de su aplicación. Por ejemplo, puede especificar una etiqueta en un clúster de base de datos para registrar la última vez que realizó una transición a ese clúster. Puede especificar una etiqueta en el clúster global para registrar la hora del último simulacro de recuperación ante desastres de toda la aplicación.

Ejemplos de la AWS CLI de etiquetado para bases de datos globales

Los siguientes ejemplos de la AWS CLI muestran cómo puede especificar y examinar las etiquetas de todos los tipos de recursos de Aurora de su base de datos global.

Puede especificar etiquetas para el contenedor de clústeres global con el comando `create-global-cluster`. En el siguiente ejemplo se crea un clúster global y se le asignan dos etiquetas. Las etiquetas tienen claves `tag1` y `tag2`.

```
$  aws rds create-global-cluster --global-cluster-identifier my_global_cluster_id \
  --engine aurora-mysql --tags Key=tag1,Value=val1 Key=tag2,Value=val2
```

Puede enumerar las etiquetas en el contenedor de clústeres global con el comando `describe-global-clusters`. Cuando se trabaja con etiquetas, se suele ejecutar primero este comando para recuperar el nombre de recurso de Amazon (ARN) del clúster global. El ARN se utiliza como parámetro en los siguientes comandos para trabajar con etiquetas. El siguiente comando muestra la información de etiqueta del atributo `TagList`. También muestra el ARN, que se utiliza como parámetro en los ejemplos posteriores.

```
$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "GlobalClusterArn": "my_global_cluster_arn",
            ...
            "TagList": [
                {
                    "Value": "val1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                }
            ]
        }
    ]
}
```

Puede agregar etiquetas nuevas con el comando `add-tags-to-resource`. Con este comando, se especifica el nombre de recurso de Amazon (ARN) del clúster global en lugar de su identificador. Si agrega una etiqueta con el mismo nombre que una etiqueta existente, se sobrescribirá el valor de esa etiqueta. Si incluye espacios o caracteres especiales en los valores de las etiquetas, cite los valores según corresponda para su sistema operativo o intérprete de comandos. En el siguiente ejemplo se modifican las etiquetas del clúster global del ejemplo anterior. Originalmente, el clúster tenía etiquetas con claves `tag1` y `tag2`. Una vez finalizado el comando, el clúster global tiene una nueva etiqueta con clave `tag3` y la etiqueta con clave `tag1` tiene un valor diferente.

```
$  aws rds add-tags-to-resource --resource-name my_global_cluster_arn \
  --tags Key=tag1,Value="new value for tag1" Key=tag3,Value="entirely new tag"

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                },
                {
                    "Value": "entirely new tag",
                    "Key": "tag3"
                }
            ]
        }
    ]
}
```

Puede eliminar una etiqueta del clúster global con el comando `remove-tags-from-resource`. Con este comando, solo se especifica un conjunto de claves de etiqueta, sin ningún valor de etiqueta. En el siguiente ejemplo se modifican las etiquetas del clúster global del ejemplo anterior.

Originalmente, el clúster tenía etiquetas con claves `tag1`, `tag2` y `tag3`. Una vez finalizado el comando, solo queda la etiqueta con la clave `tag1`.

```
$  aws rds remove-tags-from-resource --resource-name my_global_cluster_arn --tag-keys
 tag2 tag3

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                }
            ]
        }
    ]
}
```

Conexión a la base de datos global de Amazon Aurora

Cada base de datos global de Aurora incluye un punto de conexión del escritor que Aurora actualiza automáticamente para enrutar las solicitudes a la instancia del escritor actual del clúster de base de datos principal. Con el punto de conexión del escritor, no tiene que modificar la cadena de conexión después de cambiar la ubicación de la región principal mediante las funciones administradas de transición y conmutación por error de la Base de datos global de Aurora. Para obtener más información sobre el uso del punto de conexión del escritor junto con la transición y la conmutación por error de la base de datos global de Aurora, consulte Uso de la transición o la conmutación por error en la base de datos global de Amazon Aurora. Para obtener información sobre cómo conectarse a una base de datos global de Aurora con un proxy de RDS, consulte Uso del proxy de RDS con bases de datos globales de Aurora.

Temas

- Elección del punto de conexión que se adapte a las necesidades de su aplicación
- Visualización de los puntos de conexión de una base de datos global de Amazon Aurora

- Consideraciones sobre el uso de los puntos de conexión del escritor global

Elección del punto de conexión que se adapte a las necesidades de su aplicación

La conexión a una base de datos global de Aurora depende de su necesidad de leer o escribir desde la base de datos y de la región de AWS a la que desee dirigir sus solicitudes. Estas son algunas situaciones de uso típicas:

- Enrutamiento de solicitudes a la instancia de escritor: conéctese al punto de conexión de escritor de Base de datos global de Aurora si necesita ejecutar instrucciones de lenguaje de manipulación de datos (DML) y de definición de datos (DDL), o si necesita una coherencia sólida entre las lecturas y las escrituras. Ese punto de conexión redirige las solicitudes a la instancia del escritor del clúster principal de la base de datos global. Este punto de conexión se actualiza automáticamente para dirigir las solicitudes a la instancia del escritor, lo que elimina la necesidad de actualizar la aplicación cada vez que se cambia la ubicación del escritor en el clúster global. También puede utilizar el punto de conexión global para enviar solicitudes de lectura/escritura entre regiones a su escritor.

  > **Note**
  >
  > Si configuró la base de datos global antes de que estuviera disponible el punto de conexión del escritor de la base de datos global de Aurora, la aplicación podría conectarse al punto de conexión del clúster principal. En este caso, le recomendamos que cambie la configuración de conexión para utilizar en su lugar el punto de conexión del escritor global. De este modo, se evita la necesidad de cambiar la configuración de la conexión después de cada transición o conmutación por error de la base de datos global de Aurora.
  > La primera parte del nombre del punto de conexión del escritor es el nombre de la base de datos global de Aurora. Por lo tanto, si cambia el nombre de la base de datos global de Aurora, el nombre del punto de conexión del escritor cambia y cualquier código que lo utilice debe actualizarse con el nuevo nombre.

- Escalado de lecturas más cercanas a la región de la aplicación: para escalar las solicitudes de solo lectura en la misma región o en una región de AWS cercana a la de la aplicación, conéctese al punto de conexión del lector de los clústeres de Aurora principal o secundario.
- Escalado de lecturas con escrituras ocasionales entre regiones: para instrucción de DML ocasionales, como las relacionadas con el mantenimiento y la limpieza de datos, conéctese al

punto de conexión del lector de un clúster secundario que tenga habilitada el reenvío de escritura. Con el reenvío de escrituras, Aurora reenvía automáticamente las instrucciones escritas al escritor de la región principal de su base de datos global de Aurora. El reenvío de escrituras proporciona los siguientes beneficios:

- No es necesario hacer el trabajo pesado para establecer la conectividad entre los clústeres secundario y principal para enviar escrituras entre regiones.
- No es necesario dividir las solicitudes de lectura y escritura en la aplicación.
- No es necesario desarrollar una lógica compleja para administrar la coherencia de las solicitudes de lectura después de escritura.

Sin embargo, con el reenvío de escritura, es necesario actualizar el código o la configuración de la aplicación para conectarse al punto de conexión de lectura de la región principal recién promocionada tras realizar una conmutación por error o transición entre regiones. Le recomendamos que supervise la latencia de las operaciones que se realizan mediante el reenvío de escritura para evitar la sobrecarga que supone el procesamiento de las solicitudes de escritura. Por último, el reenvío de escritura no admite determinadas operaciones de MySQL o PostgreSQL, como realizar instrucciones `SELECT FOR UPDATE` o cambios en el lenguaje de definición de datos (DDL).

Para obtener más información sobre el uso del reenvío de escrituras en todas las regiones de AWS, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

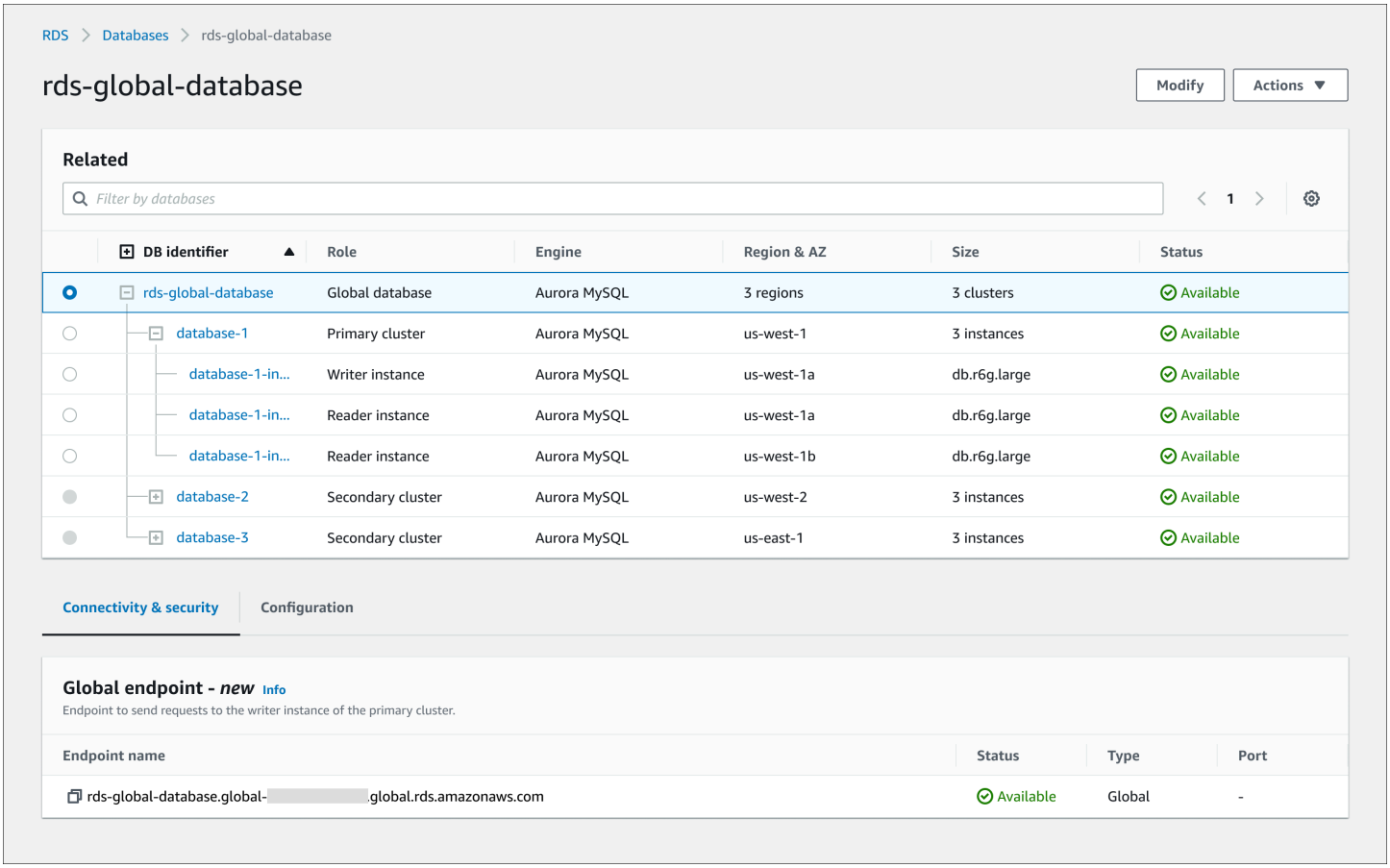
Para obtener más información sobre los distintos tipos de puntos de conexión de Aurora, consulte Conexión a un clúster de base de datos Amazon Aurora.

Visualización de los puntos de conexión de una base de datos global de Amazon Aurora

Al ver una base de datos global de Aurora en la consola, puede consultar todos los puntos de conexión asociados a todos los clústeres. En la siguiente figura se muestra un ejemplo de los tipos de puntos de conexión que ve al consultar los detalles de su clúster de base de datos principal:

- Escritor global: el único punto de conexión de lectura/escritura que siempre apunta a la instancia de base de datos de escritor actual del clúster de base de datos global.
- Escritor: el punto de conexión de la conexión para las solicitudes de lectura/escritura al clúster de base de datos principal del clúster de base de datos global.

- Lector: el punto de conexión de la conexión para las solicitudes de solo lectura a un clúster de base de datos principal o secundario del clúster de base de datos global. Para minimizar la latencia, elija el punto de conexión del lector de la Región de AWS o la Región de AWS que esté más cercana a usted.



Consola

Par ver los puntos de conexión de una base de datos global

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. En la lista, elija la base de datos global o el clúster de base de datos principal o secundario cuyos puntos de conexión desea ver.
4. Seleccione la pestaña Conectividad y seguridad para ver los detalles del punto de conexión. Los puntos de conexión mostrados dependen del tipo de clúster que ha seleccionado, de la siguiente manera:

- Base de datos global: el punto de conexión del escritor global.
- Clúster de base de datos principal: el punto de conexión del escritor global y el punto de conexión del clúster y el punto de conexión del lector del clúster principal.
- Clúster de base de datos secundario: el punto de conexión del clúster y el punto de conexión del lector del clúster secundario. En un clúster secundario, el punto de conexión del clúster muestra el estado inactivo porque no administra las solicitudes de escritura. Aún puede conectarse al punto de conexión del clúster, pero solo para consultas de lectura.

AWS CLI

Para ver el punto de conexión del escritor del clúster global, use el comando describe-global-clusters de la AWS CLI, como en el siguiente ejemplo.

```
aws rds describe-global-clusters --region aws_region
{
    "GlobalClusters": [
        {
            "GlobalClusterIdentifier": "global_cluster_id",
            "GlobalClusterResourceId": "cluster-unique_string",
            "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:global_cluster_id",
            "Status": "available",
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",
            "GlobalClusterMembers": [

              ...
            ],
            "Endpoint":
 "global_cluster_id.global-unique_string.global.rds.amazonaws.com"
        }
    ]
}
```

Para ver los puntos de conexión del clúster y del lector de los clústeres de base de datos miembros del clúster global, utilice el comando describe-db-clusters de la AWS CLI, como en el siguiente ejemplo. Los valores devueltos para `Endpoint` y `ReaderEndpoint` son los puntos de conexión del clúster y del lector, respectivamente.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier db_cluster_id
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "az_1",
                "az_2",
                "az_3"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "db_cluster_id",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-08-01T18:21:11.301Z",
            "Endpoint":
 "db_cluster_id.cluster-unique_string.primary_region.rds.amazonaws.com",
            "ReaderEndpoint": "db_cluster_id.cluster-
ro-unique_string.primary_region.rds.amazonaws.com",
            "MultiAZ": false,
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",

            "ReadReplicaIdentifiers": [
                "arn:aws:rds:secondary_region:123456789012:cluster:db_cluster_id"
            ],
            "DBClusterMembers": [
                {
                    "DBInstanceIdentifier": "db_instance_id",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                }
            ],

            ...
            "TagList": [],
            "GlobalWriteForwardingRequested": false
        }
    ]
}
```

API de RDS

Para ver el punto de conexión del escritor del clúster global, utilice la operación DescribeGlobalClusters de la API de RDS. Para ver los puntos de conexión del clúster y del lector de los clústeres de base de datos miembros del clúster global, utilice el comando DescribeDBClusters de la API de RDS.

Consideraciones sobre el uso de los puntos de conexión del escritor global

Puede hacer un uso eficaz de los puntos de conexión del escritor de la base de datos global de Aurora siguiendo estas pautas y prácticas recomendadas:

- Para minimizar las interrupciones tras una conmutación por error o transición entre regiones, puede configurar la conectividad de VPC entre el procesamiento de la aplicación y las regiones de AWS principal y secundaria. Por ejemplo, supongamos que tiene aplicaciones o sistemas cliente que se ejecutan en la misma VPC que el clúster principal. Si se promociona el clúster secundario, el punto de conexión del escritor global cambia automáticamente para apuntar a ese clúster. Si bien el punto de conexión del escritor global le permite evitar cambiar la configuración de conexión de su aplicación, sus aplicaciones no pueden acceder a las direcciones IP de la VPC de la región de AWS principal recién promocionada hasta que configure las redes entre las dos VPC. Consulte Amazon VPC-to-Amazon VPC connectivity options para evaluar las diferentes opciones para configurar esta conectividad.
- La actualización del punto de conexión del escritor global tras una transición o conmutación por error de base de datos global puede tardar mucho tiempo en función de la duración del almacenamiento en caché del sistema de nombres de dominio (DNS). Consulte el manual del administrador de base de datos Amazon Aurora MySQL para obtener más información. La base de datos global de Aurora emite un evento de RDS cuando ve el cambio de DNS en el punto de conexión del escritor global. Puede utilizar el evento para diseñar estrategias que garanticen que la memoria caché del DNS no se extienda más allá del tiempo transcurrido desde que se haya generado el evento. Para obtener más información, consulte Eventos de clúster de bases de datos.
- La base de datos global de Aurora replica los datos de forma asíncrona. Los métodos de conmutación por error entre regiones pueden provocar algunos datos de transacciones de escritura que no se replicaron en el secundario elegido antes de que se iniciara la conmutación por error. Si bien Aurora intenta bloquear las escrituras en la región de AWS principal original, la conmutación por error puede provocar problemas de cerebro dividido. Las consideraciones para minimizar la pérdida de datos y el riesgo de división cerebral también se aplican a los puntos de

conexión del escritor de base de datos global de Aurora. Para obtener más información, consulte Ejecución de la conmutación por error administrada para bases de datos globales de Aurora.

Uso del reenvío de escritura en una base de datos Amazon Aurora global

Puede reducir el número de puntos de enlace que necesita administrar para las aplicaciones que se ejecutan en la base de datos Aurora global mediante el reenvío de escritura. Si el reenvío de escritura está habilitado, los clústeres secundarios de una base de datos Aurora global podrán reenviar sentencias SQL que realizan operaciones de escritura al clúster principal. El clúster principal actualiza el origen y, a continuación, propaga los cambios resultantes a todas las regiones secundarias AWS.

La configuración de reenvío de la escritura le ayuda a evitar la implementación de su propio mecanismo para enviar operaciones de escritura desde una región secundaria a la región primaria de AWS. Aurora maneja la configuración de redes entre regiones. Aurora también transmite toda la sesión necesaria y el contexto transaccional para cada sentencia. Los datos siempre se cambian primero en el clúster principal y, a continuación, se replican de nuevo en los clústeres secundarios Aurora. De esta manera, el clúster principal siempre es la fuente fiable, ya que tiene la copia más actualizada de todos sus datos.

Temas

- Uso del reenvío de escritura en una base de datos global de Aurora MySQL
- Uso del reenvío de escritura en una base de datos Aurora PostgreSQL global

Uso del reenvío de escritura en una base de datos global de Aurora MySQL

Temas

- Disponibilidad regional y por versiones del reenvío de escritura en Aurora MySQL
- Habilitación del reenvío de escritura en Aurora MySQL
- Comprobación de si un clúster secundario tiene habilitado el reenvío de escritura
- Compatibilidad de las aplicaciones y SQL con el reenvío de escritura en Aurora MySQL
- Aislamiento y coherencia del reenvío de escritura en Aurora MySQL
- Ejecución de instrucciones multiparte con reenvío de escritura en Aurora MySQL

- Transacciones con reenvío de escritura en Aurora MySQL
- Parámetros de configuración para el reenvío de escritura en Aurora MySQL
- Métricas de Amazon CloudWatch para el reenvío de escritura en Aurora MySQL
- Variables de estado de Aurora MySQL para el reenvío de escritura

Disponibilidad regional y por versiones del reenvío de escritura en Aurora MySQL

El reenvío de escritura es compatible con Aurora MySQL 2.08.1 y versiones posteriores en todas las regiones donde estén disponibles las bases de datos globales basadas en Aurora MySQL.

Para obtener más información acerca la disponibilidad regional y por versiones de las bases de datos globales de Aurora MySQL, consulte Bases de datos globales de Aurora con Aurora MySQL.

Habilitación del reenvío de escritura en Aurora MySQL

De forma predeterminada, el reenvío de escritura no está habilitado cuando se agrega un clúster secundario a una base de datos global de Aurora.

Para habilitar el reenvío de escritura mediante la Consola de administración de AWS, elija la casilla de verificación Activar el reenvío de escritura global bajo Reenvío de escritura de réplicas de lectura cuando agregue una región en una base de datos global. Para un clúster secundario existente, modifique el clúster a Activar el reenvío de escritura global. Para desactivar el reenvío de escritura, desactive la casilla de verificación Activar el reenvío de escritura global al agregar la región o modificar el clúster secundario.

Para habilitar el reenvío de escritura mediante la AWS CLI, utilice la opción `--enable-global-write-forwarding`. Esta opción funciona cuando crea un nuevo clúster secundario mediante el comando `create-db-cluster`. También funciona cuando modifica un clúster secundario existente mediante el comando `modify-db-cluster`. Requiere que la base de datos global utilice una versión de Aurora que admita el reenvío de escritura. Puede desactivar el reenvío de escritura mediante la opción `--no-enable-global-write-forwarding` con estos mismos comandos de la CLI.

Para habilitar el reenvío de escritura mediante la API de Amazon RDS, establezca el parámetro `EnableGlobalWriteForwarding` en `true`. Este parámetro funciona cuando crea un nuevo clúster secundario mediante la operación `CreateDBCluster`. También funciona cuando modifica un clúster secundario existente mediante la operación `ModifyDBCluster`. Requiere que la base

de datos global utilice una versión de Aurora que admita el reenvío de escritura. Puede desactivar el reenvío de escritura estableciendo el parámetro `EnableGlobalWriteForwarding` en `false`.

> **Note**
>
> Para que una sesión de base de datos utilice el reenvío de escritura, especifique una configuración para el parámetro de configuración `aurora_replica_read_consistency`. Haga esto en cada sesión que utilice la característica de reenvío de escritura. Para obtener información acerca de este parámetro, consulte Aislamiento y coherencia del reenvío de escritura en Aurora MySQL.
>
> La característica RDS Proxy no admite el valor `SESSION` de la variable `aurora_replica_read_consistency`. Esto puede provocar un comportamiento inesperado.

Los siguientes ejemplos de CLI muestran cómo puede configurar una base de datos global de Aurora con reenvío de escritura habilitado o deshabilitado. Los elementos resaltados representan los comandos y las opciones que son importantes para especificar y mantener la coherencia al configurar la infraestructura de una base de datos global de Aurora.

En el siguiente ejemplo se crea una base de datos global de Aurora, un clúster principal y un clúster secundario con reenvío de escritura habilitado. Sustituya sus propias opciones por el nombre de usuario, la contraseña y las regiones principales y secundarias de AWS.

```
# Create overall global database.
aws rds create-global-cluster --global-cluster-identifier write-forwarding-test \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create primary cluster, in the same AWS Region as the global database.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-1 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --master-username user_name --master-user-password password \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
```

```
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create secondary cluster, in a different AWS Region than the global database,
# with write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2 \
  --enable-global-write-forwarding

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2
```

El siguiente ejemplo continúa desde el anterior. Crea un clúster secundario sin el reenvío de escritura habilitado y, a continuación, habilita el reenvío de escritura. Una vez finalizado este ejemplo, todos los clústeres secundarios de la base de datos global tendrán habilitado el reenvío de escritura.

```
# Create secondary cluster, in a different AWS Region than the global database,
# without write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
```

```
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds modify-db-cluster --db-cluster-identifier write-forwarding-test-cluster-2 \
  --region us-east-2 \
  --enable-global-write-forwarding
```

Comprobación de si un clúster secundario tiene habilitado el reenvío de escritura

Para determinar si puede utilizar el reenvío de escritura desde un clúster secundario, puede comprobar si el clúster tiene el atributo `"GlobalWriteForwardingStatus": "enabled"`.

En la Consola de administración de AWS, en la pestaña Configuración de la página de detalles del clúster, verá el estado Habilitado para Reenvío de escritura de réplica de lectura global.

Para ver el estado de la configuración de reenvío de escritura global de todos los clústeres, ejecute el siguiente comando de AWS CLI.

Un clúster secundario muestra el valor `"enabled"` o `"disabled"` para indicar si el reenvío de escritura está activado o desactivado. Un valor de `null` indica que el reenvío de escritura no está disponible para ese clúster. O el clúster no forma parte de una base de datos global o es el clúster principal en lugar de un clúster secundario. El valor también puede ser `"enabling"` o `"disabling"` si el reenvío de escritura está en proceso de ser activado o desactivado.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu

[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
```

```
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Para buscar todos los clústeres secundarios que tienen habilitado el reenvío de escritura global, ejecute el siguiente comando. Este comando también devuelve el punto de enlace del lector del clúster. Utilice el punto de conexión del lector del clúster secundario para cuando se utiliza el reenvío de escritura desde el secundario al primario en la base de datos Aurora global.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilidad de las aplicaciones y SQL con el reenvío de escritura en Aurora MySQL

Puede utilizar los siguientes tipos de instrucciones SQL con reenvío de escritura:

- Instrucciones de lenguaje de manipulación de datos (DML) como `INSERT`, `DELETE` y `UPDATE`. Existen algunas restricciones sobre las propiedades de estas instrucciones que puede utilizar con el reenvío de escritura, como se describe a continuación.
- Instrucciones `SELECT ... LOCK IN SHARE MODE` y `SELECT FOR UPDATE`.
- Instrucciones `PREPARE` y `EXECUTE`.

Algunas instrucciones no están permitidas o pueden producir resultados obsoletos cuando se utilizan en una base de datos global con reenvío de escritura. Por ello, la configuración

`EnableGlobalWriteForwarding` está desactivada de forma predeterminada para los clústeres secundarios. Antes de activarla, asegúrese de que el código de la aplicación no se vea afectado por ninguna de estas restricciones.

Las siguientes restricciones se aplican a las instrucciones SQL que utiliza con el reenvío de escritura. En algunos casos, puede utilizar las instrucciones en clústeres secundarios con reenvío de escritura habilitado en el nivel de clúster. Este enfoque funciona si el reenvío de escritura no está activado dentro de la sesión por el parámetro de configuración `aurora_replica_read_consistency`. Intentar usar una instrucción cuando no está permitida por el reenvío de escritura provoca un mensaje de error con el siguiente formato.

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

Lenguaje de definición de datos (DDL)

Conéctese al clúster principal para ejecutar las instrucciones de lenguaje de definición de datos. No puede ejecutarlas desde instancias de base de datos del lector.

Actualizar una tabla permanente con datos de una tabla temporal

Puede utilizar tablas temporales en clústeres secundarios con el reenvío de escritura habilitado. Sin embargo, no puede utilizar una instrucción DML para modificar una tabla permanente si la instrucción hace referencia a una tabla temporal. Por ejemplo, no puede utilizar una instrucción `INSERT ... SELECT` que saque los datos de una tabla temporal. La tabla temporal existe en el clúster secundario y no está disponible cuando la instrucción se ejecuta en el clúster principal.

Transacciones XA

No puede utilizar las siguientes instrucciones en un clúster secundario cuando el reenvío de escritura esté activado dentro de la sesión. Puede utilizar estas instrucciones en clústeres secundarios que no tengan habilitado el reenvío de escritura o en sesiones en las que la configuración `aurora_replica_read_consistency` esté vacía. Antes de activar el reenvío de escritura dentro de una sesión, compruebe si su código utiliza estas instrucciones.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Instrucciones LOAD para tablas permanentes

No puede utilizar las siguientes instrucciones en un clúster secundario con reenvío de escritura habilitado.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
        LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Puede cargar datos en una tabla temporal de un clúster secundario. Sin embargo, asegúrese de ejecutar cualquier instrucción de LOAD que haga referencia a tablas permanentes solo en el clúster principal.

Instrucciones de complemento

No puede utilizar las siguientes instrucciones en un clúster secundario con reenvío de escritura habilitado.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Declaraciones SAVEPOINT

No puede utilizar las siguientes instrucciones en un clúster secundario cuando el reenvío de escritura esté activado dentro de la sesión. Puede utilizar estas instrucciones en clústeres secundarios que no tengan habilitado el reenvío de escritura o en sesiones en las que la configuración `aurora_replica_read_consistency` esté en blanco. Compruebe si su código utiliza estas instrucciones antes de activar el reenvío de escritura dentro de una sesión.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Aislamiento y coherencia del reenvío de escritura en Aurora MySQL

En las sesiones que utilizan reenvío de escritura, solo puede utilizar el nivel de aislamiento `REPEATABLE READ`. Aunque también puede utilizar el nivel de aislamiento `READ COMMITTED` con clústeres de solo lectura en regiones secundarias de AWS, ese nivel de aislamiento no funciona con el reenvío de escritura. Para obtener información acerca de los niveles de aislamiento de `REPEATABLE READ` y `READ COMMITTED`, consulte Niveles de aislamiento de Aurora MySQL.

Puede controlar cuál es el grado de coherencia de lectura en un clúster secundario. El nivel de coherencia de lectura determina cuánto espera el clúster secundario antes de cada operación de lectura para garantizar que algunos de los cambios o todos los cambios se repliquen desde el clúster principal. Puede ajustar el nivel de coherencia de lectura para asegurarse de que todas las operaciones de escritura reenviadas desde la sesión estén visibles en el clúster secundario antes de cualquier consulta posterior. También puede utilizar esta configuración para asegurarse de que las consultas del clúster secundario siempre vean las actualizaciones más recientes del clúster principal. Esto es así incluso para los presentados por otros periodos de sesiones u otros grupos temáticos. Para especificar este tipo de comportamiento para la aplicación, elija un valor para el parámetro de nivel de sesión `aurora_replica_read_consistency`.

> **Important**
>
> Establezca siempre el parámetro `aurora_replica_read_consistency` para cualquier sesión para la que desee reenviar escrituras. De lo contrario, Aurora no habilitará el reenvío de escritura en esa sesión. Este parámetro tiene un valor vacío por defecto, por lo que debe elegir un valor específico cuando utilice este parámetro. El parámetro `aurora_replica_read_consistency` solo tiene efecto en clústeres secundarios donde está habilitado el reenvío de escritura.
> Para la versión 2 de Aurora MySQL y la versión 3 anterior a la 3.04, utilice `aurora_replica_read_consistency` como variable de sesión.
> Para la versión 3.04 y versiones posteriores de Aurora MySQL, puede usar `aurora_replica_read_consistency` como variable de sesión o como parámetro de clúster de base de datos.

Para el parámetro `aurora_replica_read_consistency`, puede especificar los valores `EVENTUAL`, `SESSION`, y `GLOBAL`.

A medida que aumenta el nivel de coherencia, la aplicación pasa más tiempo esperando que los cambios se propaguen entre las regiones de AWS. Puede buscar el equilibrio entre un tiempo de respuesta rápido y asegurarse de que los cambios realizados en otras ubicaciones estén completamente disponibles antes de que se ejecuten las consultas.

Con la coherencia de lectura establecida en `EVENTUAL`, las consultas en una región secundaria de AWS que utiliza el reenvío de escritura pueden ver datos ligeramente obsoletos debido al retardo de reproducción. Los resultados de las operaciones de escritura de la misma sesión no son visibles hasta que la operación de escritura se realiza en la región principal y se replica en la región actual.

La consulta no espera a que los resultados actualizados estén disponibles. Por lo tanto, podría recuperar los datos antiguos o los datos actualizados, en función del momento de las instrucciones y la cantidad de retardo de replicación.

Con la coherencia de lectura establecida en `SESSION`, todas las consultas de una región secundaria de AWS que utiliza el reenvío de escritura verán los resultados de todos los cambios realizados en esa sesión. Los cambios son visibles independientemente de si la transacción está confirmada. Si es necesario, la consulta espera a que los resultados de las operaciones de escritura reenviadas se repliquen en la región actual. No espera a que se actualicen los resultados de las operaciones de escritura realizadas en otras regiones o en otras sesiones dentro de la región actual.

Con la coherencia de lectura establecida en `GLOBAL`, una sesión de una región secundaria de AWS verá los cambios realizados por esa sesión. También verá todos los cambios confirmados tanto de la región principal de AWS como de otras regiones secundarias de AWS. Cada consulta puede esperar un tiempo, que variará en función de la cantidad de retardo de la sesión. La consulta continúa cuando el clúster secundario está actualizado con todos los datos confirmados del clúster principal, a partir del momento en que comenzó la consulta.

Para obtener más información sobre todos los parámetros relacionados con el reenvío de escritura, consulte Parámetros de configuración para el reenvío de escritura en Aurora MySQL.

Ejemplos de uso del reenvío de escritura

Estos ejemplos utilizan `aurora_replica_read_consistency` como variable de sesión. Para la versión 3.04 y versiones posteriores de Aurora MySQL, puede usar `aurora_replica_read_consistency` como variable de sesión o como parámetro de clúster de base de datos.

En el ejemplo siguiente, el clúster principal se encuentra en la región de US East (N. Virginia) El clúster secundario se encuentra en la región de EE.UU. Este (Ohio). El ejemplo muestra los efectos de ejecutar sentencias `INSERT` seguidas de instrucciones `SELECT`. Dependiendo de cuál sea el valor de la configuración `aurora_replica_read_consistency`, los resultados pueden diferir en función del momento en que se produzcan las instrucciones. Para lograr una mayor coherencia, puede esperar brevemente antes de emitir la instrucción `SELECT`. O Aurora puede esperar automáticamente hasta que los resultados terminen de replicarse antes de continuar con `SELECT`.

En este ejemplo, hay una configuración de coherencia de lectura de `eventual`. Ejecutar una instrucción `INSERT` inmediatamente seguida de una instrucción `SELECT` todavía devuelve el valor de

`COUNT(*)`. Este valor refleja el número de filas antes de insertar la nueva fila. Al ejecutar `SELECT` de nuevo poco tiempo después se devuelve el recuento de filas actualizado. Las instrucciones `SELECT` no esperan.

```
mysql> set aurora_replica_read_consistency = 'eventual';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Con una configuración de coherencia de lectura de `session`, una instrucción `SELECT` inmediatamente después de una instrucción `INSERT` espera hasta que los cambios de la instrucción `INSERT` sean visibles. Las instrucciones `SELECT` posteriores no esperan.

```
mysql> set aurora_replica_read_consistency = 'session';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)
mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
```

```
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Con la configuración de coherencia de lectura todavía establecida en `session`, al introducir una breve espera después de realizar una instrucción `INSERT`, el recuento de filas actualizado estará disponible para cuando se ejecute la siguiente instrucción `SELECT`.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Con una configuración de coherencia de lectura de `global`, cada instrucción `SELECT` espera para asegurarse de que todos los cambios de datos que se realicen a partir de la hora de inicio de la instrucción sean visibles antes de realizar la consulta. La cantidad de espera de cada instrucción `SELECT` varía en función de la cantidad de retardo de replicación entre los clústeres principal y secundario.

```
mysql> set aurora_replica_read_consistency = 'global';
mysql> select count(*) from t1;
+----------+
| count(*) |
```

```
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Ejecución de instrucciones multiparte con reenvío de escritura en Aurora MySQL

Una instrucción DML puede constar de varias partes, como una instrucción `INSERT ... SELECT` o una instrucción `DELETE ... WHERE`. En este caso, la instrucción completa se reenvía al clúster principal y se ejecuta allí.

Transacciones con reenvío de escritura en Aurora MySQL

Que la transacción se reenvíe al clúster principal o no depende del modo de acceso de la transacción. Puede especificar el modo de acceso de la transacción mediante la instrucción `SET TRANSACTION` o la instrucción `START TRANSACTION`. También puede cambiar el valor de la variable de sesión transaction_read_only para especificar el modo de acceso de la transacción. Solo puede cambiar este valor de sesión cuando esté conectado a un clúster de base de datos que tenga habilitado el reenvío de escritura.

Si una transacción de larga duración no emite ninguna instrucción durante un período de tiempo significativo, podría exceder el período de tiempo de espera de inactividad. Este período tiene un valor predeterminado de un minuto. Puede aumentarlo hasta un día. El clúster principal cancela las transacciones que superan el tiempo de espera de inactividad. La siguiente instrucción que envíe recibirá un error de tiempo de espera. A continuación, Aurora revertirá la transacción.

Este tipo de error puede producirse en otros casos cuando el reenvío de escritura deja de estar disponible. Por ejemplo, Aurora cancela cualquier transacción que utilice reenvío de escritura si reinicia el clúster principal o si desactiva la configuración de reenvío de escritura.

Parámetros de configuración para el reenvío de escritura en Aurora MySQL

Los grupos de parámetros del clúster de Aurora contienen nuevos ajustes para la característica de reenvío de escritura. Como se trata de parámetros de clúster, todas las instancias de base de datos de cada clúster tienen los mismos valores para estas variables. Los detalles sobre estos parámetros se resumen en la tabla siguiente, con notas de uso después de la tabla.

Nombre	Ámbito	Tipo	Valor predeterminado	Valores válidos
`aurora_fwd_master_idle_timeout` (Aurora MySQL, versión 2)	Global	entero sin signo	60	1-86 400
`aurora_fwd_master_max_connections_pct` (Aurora MySQL, versión 2)	Global	entero largo sin signo	10	0–90
`aurora_fwd_writer_idle_timeout` (Aurora MySQL, versión 3)	Global	entero sin signo	60	1-86 400
`aurora_fwd_writer_max_connections_pct` (Aurora MySQL, versión 3)	Global	entero largo sin signo	10	0–90
`aurora_replica_read_consistency`	Sesión para la versión 2 y la versión 3 anterior a la 3.04, global para la versión	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Nombre	Ámbito	Tipo	Valor predeterminado	Valores válidos
	3.04 y superior			

Para controlar las solicitudes de escritura entrantes de clústeres secundarios, utilice esta configuración en el clúster principal:

- `aurora_fwd_master_idle_timeout`,:`aurora_fwd_writer_idle_timeout` la cantidad de segundos que el clúster principal espera actividad en una conexión que se reenvía desde un clúster secundario antes de cerrarla. Si la sesión permanece inactiva al finalizar este período, Aurora la cancela.
- `aurora_fwd_master_max_connections_pct`,:`aurora_fwd_writer_max_connections_pct` el límite superior en conexiones de base de datos que se puede utilizar en una instancia de base de datos de escritor para gestionar las consultas reenviadas desde los lectores. Se expresa como un porcentaje de la configuración `max_connections` de la instancia de base de datos de escritor en el clúster principal. Por ejemplo, si `max_connections` es 800 y `aurora_fwd_master_max_connections_pct` o `aurora_fwd_writer_max_connections_pct` es 10, el escritor permite un máximo de 80 sesiones reenviadas simultáneas. Estas conexiones provienen del mismo grupo de conexiones administrado por la configuración `max_connections`.

  Esta configuración solo se aplica en el clúster principal, cuando uno o más clústeres secundarios tienen habilitado el reenvío de escritura. Si disminuye el valor, las conexiones existentes no se ven afectadas. Aurora tendrá en cuenta el nuevo valor de la configuración al intentar crear una nueva conexión desde un clúster secundario. El valor predeterminado es 10, que representa el 10% del valor `max_connections`. Si habilita el reenvío de consultas en cualquiera de los clústeres secundarios, esta configuración debe tener un valor distinto de cero para que las operaciones de escritura de clústeres secundarios se realicen correctamente. Si el valor es cero, las operaciones de escritura recibirán el código de error `ER_CON_COUNT_ERROR` con el mensaje `Not enough connections on writer to handle your request`.

El parámetro `aurora_replica_read_consistency` habilita el reenvío de escritura. Usted lo usa en cada sesión. Puede especificar `EVENTUAL`, `SESSION` o `GLOBAL` para el nivel de coherencia

de lectura. Para obtener más información sobre los niveles de consistencia, consulte Aislamiento y coherencia del reenvío de escritura en Aurora MySQL. Las siguientes reglas se aplican a este parámetro:

- El valor predeterminado es '' (vacío).
- El reenvío de escritura solo está disponible en una sesión si `aurora_replica_read_consistency` está establecido en `EVENTUAL`, `SESSION` o `GLOBAL`. Este parámetro solo es relevante en instancias de lector de clústeres secundarios que tienen habilitado el reenvío de escritura y que se encuentran en una base de datos global de Aurora.
- No puede establecer esta variable (cuando está vacía) o sin establecer (cuando ya está configurada) dentro de una transacción multideclaración. Sin embargo, puede cambiarlo de un valor válido (`EVENTUAL`, `SESSION` o `GLOBAL`) a otro valor válido (`EVENTUAL`, `SESSION` o `GLOBAL`) durante una transacción de este tipo.
- La variable no puede ser `SET` cuando el reenvío de escritura no está habilitado en el clúster secundario.

Métricas de Amazon CloudWatch para el reenvío de escritura en Aurora MySQL

Las siguientes métricas de Amazon CloudWatch se aplican al clúster principal cuando se utiliza el reenvío de escritura en uno o más clústeres secundarios. Todas estas métricas se miden en la instancia de base de datos de escritor del clúster principal.

Métrica de CloudWatch	Unidad	Descripción
`AuroraDMLRejectedMasterFull`	Recuento	El número de consultas reenviadas que se han rechazado porque la sesión en la instancia de base de datos de escritura está completa. Para Aurora MySQL, versión 2.
`AuroraDMLRejectedWriterFull`	Recuento	El número de consultas reenviadas que se han rechazado porque la sesión en

Métrica de CloudWatch	Unidad	Descripción
		la instancia de base de datos de escritura está completa. Para Aurora MySQL versión 3.
`ForwardingMasterDMLLatency`	Milisegundos	Tiempo medio para procesar cada instrucción DML reenviada en la instancia de base de datos de escritor. No incluye el tiempo que tarda el clúster secundario en reenviar la solicitud de escritura ni el tiempo necesario para replicar los cambios en el clúster secundario. Para Aurora MySQL, versión 2.
`ForwardingMasterDMLThroughput`	Recuento por segundo	Número de instrucciones DML reenviadas procesadas cada segundo por esta instancia de base de datos de escritor. Para Aurora MySQL, versión 2.
`ForwardingMasterOpenSessions`	Recuento	Número de sesiones reenviadas en la instancia de base de datos de escritor. Para Aurora MySQL, versión 2.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingWriterDMLLatency`	Milisegundos	Tiempo medio para procesar cada instrucción DML reenviada en la instancia de base de datos de escritor. No incluye el tiempo que tarda el clúster secundario en reenviar la solicitud de escritura ni el tiempo necesario para replicar los cambios en el clúster secundario. Para Aurora MySQL versión 3.
`ForwardingWriterDMLThroughput`	Recuento por segundo	Número de instrucciones DML reenviadas procesadas cada segundo por esta instancia de base de datos de escritor. Para Aurora MySQL versión 3.
`ForwardingWriterOpenSessions`	Recuento	Número de sesiones reenviadas en la instancia de base de datos de escritor. Para Aurora MySQL versión 3.

Las siguientes métricas de CloudWatch se aplican a cada clúster secundario. Estas métricas se miden en cada instancia de base de datos de lector de un clúster secundario con reenvío de escritura habilitado.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingReplicaDMLLatency`	Milisegundos	Tiempo medio de respuesta de los DML reenviados durante la réplica.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingReplicaDMLThroughput`	Recuento por segundo	Número de sentencias DML reenviadas procesadas por segundo.
`ForwardingReplicaOpenSessions`	Recuento	Número de sesiones que utilizan el reenvío de escritura en una instancia de base de datos del lector.
`ForwardingReplicaReadWaitLatency`	Milisegundos	Tiempo de espera medio que una instrucción `SELECT` de una instancia de base de datos del lector espera para ponerse al día con el clúster principal. El grado en que la instancia de base de datos de lector espera antes de procesar una consulta depende de la configuración `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Recuento por segundo	Número total de `SELECT` instrucciones procesadas cada segundo en todas las sesiones que reenvían escrituras.
`ForwardingReplicaSelectLatency`	Milisegundos	Latencia `SELECT` reenviada, media sobre todas las declaraciones `SELECT` reenviadas dentro del periodo de monitoreo.

Métrica de CloudWatch	Unidad	Descripción
`ForwardingReplicaSelectThroughput`	Recuento por segundo	Rendimiento `SELECT` reenviado, media por segundo dentro del período de supervisión.

Variables de estado de Aurora MySQL para el reenvío de escritura

Las siguientes variables de estado de Aurora MySQL se aplican al clúster principal cuando se utiliza el reenvío de escritura en uno o más clústeres secundarios. Todas estas métricas se miden en la instancia de base de datos de escritor del clúster principal.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_master_dml_stmt_count`	Recuento	Número total de instrucciones DML reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL, versión 2.
`Aurora_fwd_master_dml_stmt_duration`	Microsegundos	Duración total de las instrucciones DML reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL, versión 2.
`Aurora_fwd_master_open_sessions`	Recuento	Número de sesiones reenviadas en la instancia de base de datos de escritor. Para Aurora MySQL, versión 2.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_master_select_stmt_count`	Recuento	Número total de instancias `SELECT` reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL, versión 2.
`Aurora_fwd_master_select_stmt_duration`	Microsegundos	Duración total de las instrucciones `SELECT` reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL, versión 2.
`Aurora_fwd_writer_dml_stmt_count`	Recuento	Número total de instrucciones DML reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL versión 3.
`Aurora_fwd_writer_dml_stmt_duration`	Microsegundos	Duración total de las instrucciones DML reenviadas a esta instancia de base de datos de escritor.
`Aurora_fwd_writer_open_sessions`	Recuento	Número de sesiones reenviadas en la instancia de base de datos de escritor. Para Aurora MySQL versión 3.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_writer_select_stmt_count`	Recuento	Número total de instancias `SELECT` reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL versión 3.
`Aurora_fwd_writer_select_stmt_duration`	Microsegundos	Duración total de las instrucciones `SELECT` reenviadas a esta instancia de base de datos de escritor. Para Aurora MySQL versión 3.

Las siguientes variables de estado de Aurora MySQL se aplican a cada clúster secundario. Estas métricas se miden en cada instancia de base de datos de lector de un clúster secundario con reenvío de escritura habilitado.

Variable de estado de Aurora MySQL	Unidad	Descripción
`Aurora_fwd_replica_dml_stmt_count`	Recuento	Número total de instrucciones DML reenviadas desde esta instancia de base de datos de lector.
`Aurora_fwd_replica_dml_stmt_duration`	Microsegundos	Duración total de las instrucciones DML reenviadas desde esta instancia de base de datos de lector.
`Aurora_fwd_replica_errors_session_limit`	Recuento	Número de sesiones rechazadas por el clúster principal debido a una de las

Variable de estado de Aurora MySQL	Unidad	Descripción
		siguientes condiciones de error: • writer full • Too many forwarded statements in progress.
`Aurora_fwd_replica_open_sessions`	Recuento	Número de sesiones que utilizan el reenvío de escritura en una instancia de base de datos del lector.
`Aurora_fwd_replica_read_wait_count`	Recuento	Número total de esperas de lectura tras escritura en esta instancia de base de datos del lector.
`Aurora_fwd_replica_read_wait_duration`	Microsegundos	Duración total de las esperas debido a la configuración de coherencia de lectura en esta instancia de base de datos de lector.
`Aurora_fwd_replica_select_stmt_count`	Recuento	Número total de instrucciones `SELECT` reenviadas desde esta instancia de base de datos de lector.
`Aurora_fwd_replica_select_stmt_duration`	Microsegundos	Duración total de las instrucciones `SELECT` reenviadas desde esta instancia de base de datos de lector.

Uso del reenvío de escritura en una base de datos Aurora PostgreSQL global

Temas

- Disponibilidad regional y de versiones del reenvío de escritura en Aurora PostgreSQL
- Habilitación del reenvío de escritura en Aurora PostgreSQL
- Comprobación de si un clúster secundario tiene habilitado el reenvío de escritura en Aurora PostgreSQL
- Compatibilidad de las aplicaciones con el reenvío de escritura en Aurora PostgreSQL
- Aislamiento y coherencia del reenvío de escritura en Aurora PostgreSQL
- Modos de acceso a transacciones con reenvío de escritura
- Ejecutar instrucciones multiparte con reenvío de escritura en Aurora PostgreSQL
- Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL
- Métricas de Amazon CloudWatch para el reenvío de escritura en Aurora PostgreSQL
- Eventos de espera para el reenvío de escritura en Aurora PostgreSQL

Disponibilidad regional y de versiones del reenvío de escritura en Aurora PostgreSQL

En la versión 16 de Aurora PostgreSQL y en las versiones principales posteriores, todas las versiones secundarias admiten el reenvío de escritura global. En las versiones anteriores de Aurora PostgreSQL, el reenvío de escritura global es compatible con la versión 15.4 y versiones secundarias posteriores, y con la versión 14.9 y versiones secundarias posteriores. El reenvío de escritura está disponible en todas las regiones de AWS en las que hay bases de datos globales de Aurora basadas en PostgreSQL.

Para obtener más información acerca de la disponibilidad regional y de versiones de las bases de datos Aurora PostgreSQL globales, consulte Bases de datos globales de Aurora con Aurora PostgreSQL.

Habilitación del reenvío de escritura en Aurora PostgreSQL

De forma predeterminada, el reenvío de escritura no está habilitado cuando se agrega un clúster secundario a una base de datos global de Aurora. Puede habilitar el reenvío de escritura para un clúster de base de datos secundario al crearlo o en cualquier momento posterior. Si lo necesita,

podrá desactivarlo más adelante. La activación o desactivación del reenvío de escritura no provoca tiempos de inactividad ni un reinicio.

> **Note**
>
> Puede utilizar el reenvío de escritura local para las aplicaciones que tengan escrituras ocasionales y requieran coherencia de lectura después de escritura, que es la capacidad de leer la última escritura de una transacción.

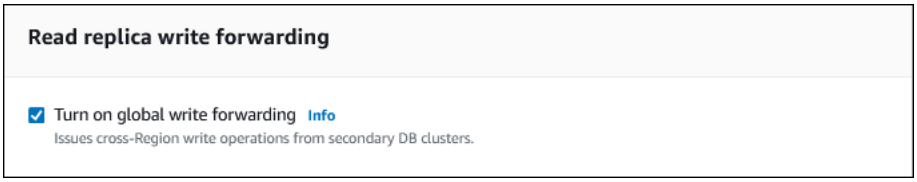
Consola

En la consola, puede activar o desactivar el reenvío de escritura al crear o modificar un clúster secundario de base de datos.

Habilitar o deshabilitar el reenvío de escritura al crear un clúster de base de datos secundario

Al crear un nuevo clúster de base de datos secundario, el reenvío de escritura se activa seleccionando la casilla Turn on global write forwarding (Activar el reenvío de escritura global) en Read replica write forwarding (Reenvío de escritura de réplica de lectura. También puede quitar la marca de la casilla para desactivarla. Para crear un clúster de base de datos secundario, siga las instrucciones indicadas en Creación de un clúster de base de datos de Amazon Aurora para su motor de base de datos.

La siguiente captura de pantalla muestra la sección Read replica write forwarding (Reenvío de escritura de réplica de lectura) con la casilla Turn on global write forwarding (Activar el reenvío de escritura global) seleccionada.



Habilitar o desactivar el reenvío de escritura al modificar un clúster de base de datos secundario

En la consola, puede modificar un clúster de base de datos secundario para habilitar o desactivar el reenvío de escritura.

Habilitar o desactivar el reenvío de escritura para un clúster de base de datos secundario con la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Seleccione Bases de datos.
3. Elija el clúster de base de datos secundario y elija Modify (Modificar).
4. En la sección Reenvío de escritura de réplica de lectura en la que aparece la casilla Turn on global write forwarding (Activar el reenvío de escritura global) seleccionada.
5. Elija Continuar.
6. En Schedule modifications (Programación de modificaciones), elija Apply immediately (Aplicar inmediatamente). Si elige Apply during the next scheduled maintenance window (Aplicar durante la próxima ventana de mantenimiento programada), Aurora ignora esta configuración y activa de inmediato el reenvío de escritura.
7. Elija Modify clúster (Modificar clúster).

AWS CLI

Para habilitar el reenvío de escritura mediante AWS CLI, utilice la opción `--enable-global-write-forwarding`. Esta opción funciona cuando crea un nuevo clúster secundario mediante el comando create-db-clúster. También funciona cuando modifica un clúster secundario existente mediante el comando modify-db-clúster. Requiere que la base de datos global utilice una versión de Aurora que admita el reenvío de escritura. Puede deshabilitar el reenvío de escritura mediante la opción `--no-enable-global-write-forwarding` con estos mismos comandos de la CLI.

Los siguientes procedimientos describen cómo habilitar o deshabilitar el reenvío de escritura para un clúster de base de datos secundario de su clúster global mediante AWS CLI.

Habilitar o desactivar el reenvío de escritura para un clúster de base de datos secundario existente

- Llame al comando AWS CLI de modify-db-clúster y suministre los siguientes valores:
  - `--db-cluster-identifier`: el nombre del clúster de bases de datos.
  - `--enable-global-write-forwarding` para activar o `--no-enable-global-write-forwarding` para desactivar.

El siguiente ejemplo habilita el reenvío de escritura para un clúster de base de datos `sample-secondary-db-cluster`.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-secondary-db-cluster \
    --enable-global-write-forwarding
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-secondary-db-cluster ^
    --enable-global-write-forwarding
```

API de RDS

Para habilitar el reenvío de escritura mediante la API de Amazon RDS, establezca el parámetro `EnableGlobalWriteForwarding` en `true`. Este parámetro funciona cuando crea un nuevo clúster secundario mediante la operación CreateDBclúster. También funciona cuando modifica un clúster secundario existente mediante la operación ModifyDBclúster. Requiere que la base de datos global utilice una versión de Aurora que admita el reenvío de escritura. Puede deshabilitar el reenvío de escritura estableciendo el parámetro `EnableGlobalWriteForwarding` en `false`.

Comprobación de si un clúster secundario tiene habilitado el reenvío de escritura en Aurora PostgreSQL

Para determinar si puede utilizar el reenvío de escritura desde un clúster secundario, puede comprobar si el clúster tiene el atributo `"GlobalWriteForwardingStatus": "enabled"`.

En la Consola de administración de AWS, verá `Read replica write forwarding` en la pestaña Configuration (Configuración) de la página de detalles del clúster. Para ver el estado de la configuración de reenvío de escritura global de todos los clústeres, ejecute el siguiente comando de AWS CLI.

Un clúster secundario muestra el valor `"enabled"` o `"disabled"` para indicar si el reenvío de escritura está activado o desactivado. Un valor de `null` indica que el reenvío de escritura no

está disponible para ese clúster. O el clúster no forma parte de una base de datos global o es el clúster principal en lugar de un clúster secundario. El valor también puede ser `"enabling"` o `"disabling"` si el reenvío de escritura está en proceso de ser activado o desactivado.

Example

```
aws rds describe-db-clusters --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Para buscar todos los clústeres secundarios que tienen habilitado el reenvío de escritura global, ejecute el siguiente comando. Este comando también devuelve el punto de enlace del lector del clúster. Utilice el punto de conexión del lector del clúster secundario para cuando se utiliza el reenvío de escritura desde el secundario al primario en la base de datos Aurora global.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilidad de las aplicaciones con el reenvío de escritura en Aurora PostgreSQL

Algunas instrucciones no están permitidas o pueden producir resultados obsoletos cuando se utilizan en una base de datos global con reenvío de escritura. Además, no se admiten funciones ni procedimientos definidos por el usuario. Por ello, la configuración `EnableGlobalWriteForwarding` está desactivada de forma predeterminada para los clústeres secundarios. Antes de activarla, asegúrese de que el código de la aplicación no se vea afectado por ninguna de estas restricciones.

Puede utilizar los siguientes tipos de instrucciones SQL con reenvío de escritura:

- Instrucciones de lenguaje de manipulación de datos (DML) como `INSERT`, `DELETE` y `UPDATE`
- Instrucciones `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`
- Instrucciones `PREPARE` y `EXECUTE`
- Instrucciones `EXPLAIN` con las instrucciones de esta lista

Los siguientes tipos de instrucciones SQL no son compatibles con el reenvío de escritura:

- Instrucciones en lenguaje de definición de datos (DDL)
- `ANALYZE`
- `CLUSTER`
- `COPY`
- Cursores: los cursores no son compatibles, así que debe asegurarse de cerrarlos antes de utilizar el reenvío de escritura.
- `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
- `LOCK`
- Instrucciones `SAVEPOINT`
- `SELECT INTO`
- `SET CONSTRAINTS`
- `TRUNCATE`
- `VACUUM`

Aislamiento y coherencia del reenvío de escritura en Aurora PostgreSQL

En las sesiones que utilizan reenvío de escritura, solo puede utilizar los niveles de aislamiento `REPEATABLE READ` y `READ COMMITTED`. Sin embargo, no se admite el nivel de `SERIALIZABLE` aislamiento.

Puede controlar cuál es el grado de coherencia de lectura en un clúster secundario. El nivel de coherencia de lectura determina cuánto espera el clúster secundario antes de cada operación de lectura para garantizar que algunos de los cambios o todos los cambios se repliquen desde el clúster principal. Puede ajustar el nivel de coherencia de lectura para asegurarse de que todas las operaciones de escritura reenviadas desde la sesión estén visibles en el clúster secundario antes de cualquier consulta posterior. También puede utilizar esta configuración para asegurarse de que las consultas del clúster secundario siempre vean las actualizaciones más recientes del clúster principal. Esto es así incluso para los presentados por otros periodos de sesiones u otros grupos temáticos. Para especificar este tipo de comportamiento para la aplicación, elija el valor adecuado para el parámetro `apg_write_forward.consistency_mode`. El parámetro `apg_write_forward.consistency_mode` solo tiene efecto en clústeres secundarios donde está habilitado el reenvío de escritura.

> **Note**
>
> Para el parámetro `apg_write_forward.consistency_mode`, puede especificar los valores `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. De forma predeterminada, el valor se establece en `SESSION`. Si se establece este valor en `OFF`, se desactiva el reenvío de escritura en la sesión.

A medida que aumenta el nivel de coherencia, la aplicación pasa más tiempo esperando que los cambios se propaguen entre las regiones de AWS. Puede buscar el equilibrio entre un tiempo de respuesta rápido y asegurarse de que los cambios realizados en otras ubicaciones estén completamente disponibles antes de que se ejecuten las consultas.

Con cada configuración del modo de coherencia disponible, el efecto es el siguiente:

- `SESSION`: todas las consultas de una región secundaria de AWS que utiliza el reenvío de escritura verán los resultados de todos los cambios realizados en esa sesión. Los cambios son visibles independientemente de si la transacción está confirmada. Si es necesario, la consulta espera a que los resultados de las operaciones de escritura reenviadas se repliquen en la región actual.

No espera a que se actualicen los resultados de las operaciones de escritura realizadas en otras regiones o en otras sesiones dentro de la región actual.

- `EVENTUAL`: las consultas en una región secundaria de AWS que utiliza el reenvío de escritura pueden ver datos ligeramente obsoletos debido al retardo de reproducción. Los resultados de las operaciones de escritura de la misma sesión no son visibles hasta que la operación de escritura se realiza en la región principal y se replica en la región actual. La consulta no espera a que los resultados actualizados estén disponibles. Por lo tanto, podría recuperar los datos antiguos o los datos actualizados, en función del momento de las instrucciones y la cantidad de retardo de replicación.
- `GLOBAL`: una sesión de una región secundaria de AWS puede ver los cambios realizados por esa sesión. También puede ver todos los cambios confirmados tanto de la región principal de AWS como de otras regiones secundarias de AWS. Cada consulta puede esperar un tiempo, que variará en función de la cantidad de retardo de la sesión. La consulta continúa cuando el clúster secundario está actualizado con todos los datos confirmados del clúster principal, a partir del momento en que comenzó la consulta.
- `OFF`: el reenvío de escritura está desactivado en la sesión.

Para obtener más información sobre todos los parámetros relacionados con el reenvío de escritura, consulte Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL.

Modos de acceso a transacciones con reenvío de escritura

Si el modo de acceso de la transacción está configurado en solo lectura, no se utiliza el reenvío de escritura. Solo puede establecer el modo de acceso en lectura/escritura cuando esté conectado a un clúster de base de datos y a una sesión que tenga habilitado el reenvío de escritura.

Para obtener más información sobre los modos de acceso a las transacciones, consulte SET TRANSACTION.

Ejecutar instrucciones multiparte con reenvío de escritura en Aurora PostgreSQL

Una instrucción DML puede constar de varias partes, como una instrucción `INSERT ... SELECT` o una instrucción `DELETE ... WHERE`. En este caso, la instrucción completa se reenvía al clúster principal y se ejecuta allí.

Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL

Los grupos de parámetros del clúster de Aurora contienen nuevos ajustes para la característica de reenvío de escritura. Como se trata de parámetros de clúster, todas las instancias de base de datos de cada clúster tienen los mismos valores para estas variables. Los detalles sobre estos parámetros se resumen en la tabla siguiente, con notas de uso después de la tabla.

Nombre	Ámbito	Tipo	Valor predeterminado	Valores válidos
`apg_write_forward.connect_timeout`	Sesión	segundos	30	0–2147483647
`apg_write_forward.consistency_mode`	Session	enum	Sesión	`SESSION`, `EVENTUAL`, `GLOBAL`, `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Sesión	milisegundos	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Sesión	milisegundos	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Global	int	25	1–100

El parámetro `apg_write_forward.max_forwarding_connections_percent` es el límite superior en slots de conexiones de base de datos que se puede utilizar para gestionar las consultas reenviadas desde los lectores. Se expresa como un porcentaje de la configuración `max_connections` de la instancia de base de datos de escritor en el clúster principal. Por ejemplo, si `max_connections` es `800` y `apg_write_forward.max_forwarding_connections_percent` es `10`, el escritor permite un máximo de 80 sesiones reenviadas simultáneas. Estas conexiones provienen del mismo grupo de conexiones administrado por la configuración `max_connections`. Esta configuración solo se aplica

en el clúster principal, cuando al menos uno de los clústeres secundarios tiene habilitado el reenvío de escritura.

Utilice la siguiente configuración en el clúster secundario:

- `apg_write_forward.consistency_mode`: un parámetro de nivel de sesión que controla el grado de coherencia de lectura en el clúster secundario. Los valores válidos son `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. De forma predeterminada, el valor se establece en `SESSION`. Si se establece este valor en `OFF`, se desactiva el reenvío de escritura en la sesión. Para obtener más información sobre los niveles de consistencia, consulte Aislamiento y coherencia del reenvío de escritura en Aurora PostgreSQL. Este parámetro solo es relevante en instancias de lector de clústeres secundarios que tienen habilitado el reenvío de escritura y que se encuentran en una base de datos global de Aurora.
- `apg_write_forward.connect_timeout`: el número máximo de segundos que espera el clúster secundario al establecer una conexión con el clúster principal antes de desistirse. Un valor de `0` significa esperar indefinidamente.
- `apg_write_forward.idle_in_transaction_session_timeout`: la cantidad de milisegundos que el clúster principal espera actividad en una conexión que se reenvía desde un clúster secundario que tiene una transacción abierta antes de cerrarla. Si la sesión permanece inactiva en la transacción al finalizar este período, Aurora la termina. Un valor de `0` deshabilita el tiempo de espera.
- `apg_write_forward.idle_session_timeout`: la cantidad de milisegundos que el clúster principal espera actividad en una conexión que se reenvía desde un clúster secundario antes de cerrarla. Si la sesión permanece inactiva al finalizar este período, Aurora la termina. Un valor de `0` desactiva el tiempo de espera.

Métricas de Amazon CloudWatch para el reenvío de escritura en Aurora PostgreSQL

Las siguientes métricas de Amazon CloudWatch se aplican al clúster principal cuando se utiliza el reenvío de escritura en uno o más clústeres secundarios. Todas estas métricas se miden en la instancia de base de datos de escritor del clúster principal.

Métrica de CloudWatch	Unidades y descripción
`AuroraForwardingWriterDMLThroughput`	Recuento por segundo Número de instrucciones DML reenviadas procesadas cada

Métrica de CloudWatch	Unidades y descripción
	segundo por esta instancia de base de datos de escritor.
`AuroraForwardingWriterOpenSessions`	Recuento Número de sesiones abiertas en esta instancia de base de datos de escritor que procesa las consultas reenviadas.
`AuroraForwardingWriterTotalSessions`	Recuento Número total de sesiones reenviadas en esta instancia de base de datos de escritor.

Las siguientes métricas de CloudWatch se aplican a cada clúster secundario. Estas métricas se miden en cada instancia de base de datos de lector de un clúster secundario con reenvío de escritura habilitado.

Métrica de CloudWatch	Unidad y descripción
`AuroraForwardingReplicaCommitThroughput`	Recuento por segundo Número de confirmaciones en las sesiones reenviadas por esta réplica cada segundo.
`AuroraForwardingReplicaDMLLatency`	Milisegundos. Tiempo medio de respuesta en milisegundos de los DML reenviados durante la réplica.
`AuroraForwardingReplicaDMLThroughput`	Recuento por segundo Número de instrucciones DML reenviadas procesadas en esta réplica por segundo.
`AuroraForwardingReplicaErrorSessionsLimit`	Recuento Número de sesiones rechazadas por el clúster primario porque se ha alcanzado el límite máximo de conexiones o el límite máximo de conexiones de reenvío de escritura.
`AuroraForwardingReplicaOpenSessions`	Recuento Número de sesiones que utilizan el reenvío de escritura en una instancia de réplica.

Métrica de CloudWatch	Unidad y descripción
`AuroraForwardingReplicaReadWaitLatency`	Milisegundos. Tiempo de espera medio en milisegundos que la réplica espera para ser coherente con el LSN del clúster principal. El grado en que la instancia de base de datos de lector espera depende de la configuración `apg_write_forward.consistency_mode` . Para obtener información sobre esta configuración, consulte the section called "Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL".

Eventos de espera para el reenvío de escritura en Aurora PostgreSQL

Amazon Aurora genera los siguientes eventos de espera cuando utiliza el reenvío de escritura con Aurora PostgreSQL.

Temas

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

El evento `IPC:AuroraWriteForwardConnect` se produce cuando un proceso de backend del clúster de base de datos secundario espera a que se abra una conexión con el nodo escritor del clúster de base de datos primario.

Causas probables del aumento del tiempo de espera

Este evento aumenta a medida que aumenta el número de intentos de conexión desde el nodo lector de una región secundaria al nodo escritor del clúster de base de datos principal.

Acciones

Reduzca el número de conexiones simultáneas desde un nodo secundario al nodo escritor de la región principal.

IPC:AuroraWriteForwardConsistencyPoint

El evento `IPC:AuroraWriteForwardConsistencyPoint` describe cuánto tiempo esperará una consulta de un nodo del clúster de base de datos secundario para que los resultados de las operaciones de escritura reenviadas se repliquen en la región actual. Este evento solo se genera si el parámetro de nivel de sesión `apg_write_forward.consistency_mode` se establece en uno de los siguientes valores:

- `SESSION`: las consultas de un nodo secundario esperan los resultados de todos los cambios realizados en esa sesión.
- `GLOBAL`: las consultas de un nodo secundario esperan los resultados de los cambios realizados en esa sesión, además de todos los cambios confirmados tanto de la región principal como de otras regiones secundarias del clúster global.

Para obtener más información sobre la configuración del parámetro `apg_write_forward.consistency_mode`, consulte the section called “Parámetros de configuración para el reenvío de escritura en Aurora PostgreSQL”.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- Aumento del retraso de réplica, medido por la métrica `ReplicaLag` de Amazon CloudWatch. Para obtener más información sobre esta métrica, consulte Monitoreo de replicación de Aurora PostgreSQL.
- Aumento de la carga en el nodo de escritor de la región principal o en el nodo secundario.

Acciones

Cambie el modo de coherencia según los requisitos de su aplicación.

IPC:AuroraWriteForwardExecute

El evento `IPC:AuroraWriteForwardExecute` se produce cuando un proceso de backend del clúster de base de datos secundario está esperando a que una consulta reenviada se complete y se obtengan los resultados del nodo escritor del clúster de base de datos primario.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- Obtención de una gran cantidad de filas del nodo escritor de la región principal.
- El aumento de la latencia de la red entre el nodo secundario y el nodo escritor de la región principal incrementa el tiempo que tarda el nodo secundario en recibir datos del nodo escritor.
- El aumento de la carga en el nodo secundario puede retrasar la transmisión de la solicitud de consulta desde el nodo secundario al nodo escritor de la región principal.
- El aumento de la carga en el nodo escritor de la región principal puede retrasar la transmisión de la solicitud de consulta desde el nodo escritor al nodo secundario.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

- Optimice las consultas para recuperar solo los datos necesarios.
- Optimice las operaciones de lenguaje de manipulación de datos (DML) para modificar únicamente los datos necesarios.
- Si el nodo escritor de la región principal o el nodo secundario está limitado por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

El `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` evento se produce cuando un proceso de backend del clúster de base de datos secundario que utiliza el modo de coherencia GLOBAL está esperando para obtener el punto de coherencia global del nodo escritor antes de ejecutar una consulta.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre el nodo secundario y el nodo escritor de la región principal incrementa el tiempo que tarda el nodo secundario en recibir datos del nodo escritor.
- El aumento de la carga en el nodo secundario puede retrasar la transmisión de la solicitud de consulta desde el nodo secundario al nodo escritor de la región principal.
- El aumento de la carga en el nodo escritor de la región principal puede retrasar la transmisión de la solicitud de consulta desde el nodo escritor al nodo secundario.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

- Cambie el modo de coherencia según los requisitos de su aplicación.
- Si el nodo escritor de la región principal o el nodo secundario está limitado por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardXactAbort

El evento `IPC:AuroraWriteForwardXactAbort` se produce cuando un proceso de backend del clúster de base de datos secundario está esperando el resultado de una consulta de limpieza remota. Las consultas de limpieza se emiten para devolver el proceso al estado correspondiente después de cancelar una transacción de reenvío de escritura. Amazon Aurora las ejecuta porque ha detectado un error o porque un usuario ha emitido un comando `ABORT` explícito o ha cancelado una consulta en ejecución.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre el nodo secundario y el nodo escritor de la región principal incrementa el tiempo que tarda el nodo secundario en recibir datos del nodo escritor.
- El aumento de la carga en el nodo secundario puede retrasar la transmisión de la solicitud de consulta de limpieza desde el nodo secundario al nodo escritor de la región principal.

- El aumento de la carga en el nodo escritor de la región principal puede retrasar la transmisión de la solicitud de consulta desde el nodo escritor al nodo secundario.

Acciones

Recomendamos diferentes acciones en función de las causas del evento de espera.

- Investigue por qué se ha cancelado la transacción.
- Si el nodo escritor de la región principal o el nodo secundario está limitado por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardXactCommit

El evento `IPC:AuroraWriteForwardXactCommit` se produce cuando un proceso de backend del clúster de base de datos secundario está esperando el resultado de un comando de transacción de confirmación reenviado.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre el nodo secundario y el nodo escritor de la región principal incrementa el tiempo que tarda el nodo secundario en recibir datos del nodo escritor.
- El aumento de la carga en el nodo secundario puede retrasar la transmisión de la solicitud de consulta desde el nodo secundario al nodo escritor de la región principal.
- El aumento de la carga en el nodo escritor de la región principal puede retrasar la transmisión de la solicitud de consulta desde el nodo escritor al nodo secundario.

Acciones

Si el nodo escritor de la región principal o el nodo secundario está limitado por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

IPC:AuroraWriteForwardXactStart

El evento `IPC:AuroraWriteForwardXactStart` se produce cuando un proceso de backend del clúster de base de datos secundario está esperando el resultado de un comando de transacción de inicio reenviado.

Causas probables del aumento del tiempo de espera

Algunas de las causas más comunes que provocan tiempos de espera más largos son las siguientes:

- El aumento de la latencia de la red entre el nodo secundario y el nodo escritor de la región principal incrementa el tiempo que tarda el nodo secundario en recibir datos del nodo escritor.
- El aumento de la carga en el nodo secundario puede retrasar la transmisión de la solicitud de consulta desde el nodo secundario al nodo escritor de la región principal.
- El aumento de la carga en el nodo escritor de la región principal puede retrasar la transmisión de la solicitud de consulta desde el nodo escritor al nodo secundario.

Acciones

Si el nodo escritor de la región principal o el nodo secundario está limitado por la CPU o por el ancho de banda de la red, puede cambiarlo por un tipo de instancia con más capacidad de CPU o más ancho de banda.

Uso de la transición o la conmutación por error en la base de datos global de Amazon Aurora

La característica de base de datos global de Aurora proporciona más protección de continuidad empresarial y recuperación ante desastres (BCDR) que la alta disponibilidad estándar proporcionada por un clúster de bases de datos de Aurora en una sola Región de AWS. Al utilizar una base de datos global de Aurora, puede planificar y recuperarse más rápido de desastres regionales o interrupciones totales del nivel de servicio inusuales y no previstos.

Puede consultar las siguientes directrices y procedimientos para planificar, probar e implementar su estrategia de BCDR mediante la característica de base de datos global de Aurora.

Temas

- Planificación de la continuidad empresarial y la recuperación ante desastres

- Ejecución de transiciones para bases de datos globales de Amazon Aurora
- Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada
- Administración de RPO para bases de datos globales basadas en Aurora PostgreSQL–
- Resiliencia entre regiones para clústeres secundarios de una base de datos global

Planificación de la continuidad empresarial y la recuperación ante desastres

Para planificar su estrategia de continuidad empresarial y recuperación ante desastres, resulta útil comprender la siguiente terminología del sector y su relación con las características de la base de datos global de Aurora.

La recuperación de desastres suele obedecer a los dos objetivos empresariales siguientes:

- Objetivo de tiempo de recuperación (RTO): el tiempo que tarda un sistema en volver a un estado operativo después de un desastre o interrupción del servicio. En otras palabras, el RTO mide el tiempo de inactividad. Para la base de datos global de Aurora, el RTO puede ser del orden de minutos.
- Objetivo de punto de recuperación (RPO): la cantidad de datos que se pueden perder (medidos en el tiempo) después de un desastre o interrupción del servicio. Esta pérdida de datos suele deberse a un retraso en la replicación asíncrona. Para una base de datos global de Aurora, el RPO se mide normalmente en segundos. Con una base de datos global basada en Aurora PostgreSQL–, puede utilizar el parámetro `rds.global_db_rpo` para establecer y realizar un seguimiento del límite superior de RPO, pero hacerlo podría afectar al procesamiento de transacciones en el nodo escritor del clúster principal. Para obtener más información, consulte Administración de RPO para bases de datos globales basadas en Aurora PostgreSQL–.

Realizar una transición o una conmutación por error con la base de datos global de Aurora implica promocionar un clúster de base de datos secundario al clúster de base de datos principal. El término “interrupción regional” se utiliza a menudo para describir una variedad de situaciones de error. El peor de los casos podría ser una interrupción generalizada provocada por un evento catastrófico que afecte a cientos de kilómetros cuadrados. Sin embargo, la mayoría de las interrupciones están mucho más localizadas y afectan solo a un pequeño subconjunto de los servicios en la nube o los sistemas de los clientes. Tenga en cuenta el alcance total de la interrupción para asegurarse de que la conmutación por error entre regiones sea la solución apropiada y elegir el método de conmutación por error adecuado para la situación. La decisión de utilizar el enfoque de transición o conmutación por error depende de la situación de interrupción concreta:

- Conmutación por error: utilice este enfoque para recuperarse de una interrupción imprevista. Con este enfoque, realiza una conmutación por error entre regiones a uno de los clústeres de bases de datos secundario de su base de datos global de Aurora. El RPO de este enfoque suele ser un valor distinto de cero medido en segundos. La cantidad de pérdida de datos depende del retraso en la replicación de la base de datos global Aurora en las Regiones de AWS en el momento del error. Para obtener más información, consulte Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada.
- Transición: esta operación se denominaba anteriormente “conmutación por error planificada administrada”. Utilice este enfoque para situaciones controladas, como el mantenimiento operativo y otros procedimientos operativos planificados en los que todos los clústeres de Aurora y otros servicios con los que interactúan se encuentren en buen estado. Dado que esta característica sincroniza los clústeres secundarios de base de datos con el principal antes de realizar cualquier otro cambio, el RPO es 0 (sin pérdida de datos). Para obtener más información, consulte Ejecución de transiciones para bases de datos globales de Amazon Aurora.

> **Note**
>
> Antes de realizar una transición o conmutación por error en un clúster de base de datos de Aurora secundario sin periféricos, debe agregarle una instancia de base de datos. Para obtener más información acerca de los clústeres de base de datos sin pantalla, consulte Creación de un clúster de base de datos de Aurora sin pantalla en una región secundaria.

Ejecución de transiciones para bases de datos globales de Amazon Aurora

> **Note**
>
> Antes, las transiciones se denominaban conmutaciones por error planificadas administradas.

Al utilizar las transiciones, puede cambiar la región del clúster principal de forma rutinaria. Este enfoque está destinado a situaciones controladas, como el mantenimiento operativo y otros procedimientos operativos planificados.

Existen tres casos de uso frecuentes en los que se utilizan las transiciones.

- Para los requisitos de “rotación regional” impuestos a sectores específicos. Por ejemplo, es posible que los reglamentos de los servicios financieros exijan que los sistemas de nivel 0 se cambien a una región diferente durante varios meses para garantizar que los procedimientos de recuperación de desastres se ensayen con cierta asiduidad.
- Para aplicaciones multirregionales del tipo “seguir el sol”. Por ejemplo, es posible que una empresa desee ofrecer escrituras con menor latencia en diferentes regiones en función del horario laboral en distintas zonas horarias.
- Como método sin pérdida de datos para conmutar por recuperación a la región principal original tras una conmutación por error.

> **Note**
>
> Las transiciones están diseñadas para usarse en una base de datos global de Aurora en la que todos los clústeres de Aurora y otros servicios con los que interactúan se encuentran en buen estado. Para la recuperación de una interrupción no programada, siga el procedimiento correspondiente en Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada.
>
> Solo puede realizar transiciones entre regiones en la base de datos global de Aurora si los clústeres de base de datos principal y secundario tienen las mismas versiones principal, secundaria y de nivel de parche del motor. Según el motor y las versiones del motor, es posible que los niveles de parche deban ser idénticos o diferentes. Para obtener una lista de los motores y las versiones de motores que permiten estas operaciones entre clústeres principales y secundarios con diferentes niveles de parches, consulte Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones.
>
> Antes de iniciar la transición, compruebe las versiones del motor de su clúster global para asegurarse de que admiten la transición entre regiones administrada y actualícelas, si es necesario.

Durante una transición, Aurora convierte el clúster de la región secundaria elegida en el clúster principal. El mecanismo de transición mantiene la topología de replicación existente de la base de datos global: sigue teniendo el mismo número de clústeres de Aurora en las mismas regiones. Antes de que Aurora inicie el proceso de transición, espera a que los clústeres de la región secundaria de destino estén completamente sincronizados con el clúster de la región principal. El clúster de base de datos en la región principal se convierte en solo de lectura. El clúster secundario elegido promociona uno de sus nodos de solo de lectura al estado de escritor completo, lo que permite que el clúster

secundario asuma el rol de clúster principal. Dado que el clúster secundario de destino se sincronizó con el principal al principio del proceso, el nuevo principal continúa las operaciones para la base de datos global de Aurora sin perder ningún dato. La base de datos no estará disponible durante un breve periodo, mientras los clústeres principales y secundarios seleccionados asumen nuevas funciones.

> **Note**
>
> Para administrar las ranuras de replicación para Aurora PostgreSQL después de realizar una transición, consulte Administración de ranuras lógicas para Aurora PostgreSQL .

Para optimizar la disponibilidad de las aplicaciones, se recomienda hacer lo siguiente antes de utilizar esta característica:

- Realice esta operación durante los horarios menos concurridos o en otro momento cuando las escrituras en el clúster de base de datos principal sean mínimas.
- Compruebe los tiempos de retraso para todos los clústeres secundarios de base de datos Aurora de la base de datos global de Aurora. Para todas las bases de datos globales basadas en Aurora PostgreSQL y para las bases de datos globales basadas en Aurora MySQL a partir de las versiones de motor 3.04.0 y posteriores, o 2.12.0 y posteriores, utilice Amazon CloudWatch para ver la métrica `AuroraGlobalDBRPOLag` de todos los clústeres de bases de datos secundarios. Para ver las versiones secundarias anteriores de las bases de datos globales basadas en Aurora MySQL, consulte, en cambio, la métrica `AuroraGlobalDBReplicationLag`. Estas métricas muestran el retraso (en milisegundos) de la replicación a un clúster secundario con respecto al clúster principal de base de datos. Este valor es directamente proporcional al tiempo que tarda Aurora en completar la transición. Por lo tanto, cuanto mayor sea el valor de retraso, más tiempo llevará la transición. Cuando examine estas métricas, hágalo desde el clúster principal actual.

  Para obtener más información acerca de las métricas de CloudWatch para Aurora, consulte Métricas de nivel de clúster para Amazon Aurora.
- El clúster de base de datos secundario que se promociona durante una transición puede tener ajustes de configuración diferentes a los del clúster de base de datos principal anterior. Se recomienda mantener la coherencia entre los siguientes tipos de ajustes de configuración en todos los clústeres de los clústeres de la base de datos global de Aurora. Esto ayuda a minimizar los problemas de rendimiento, incompatibilidades de carga de trabajo y otros comportamientos anómalos después de una transición.

- Configure un grupo de parámetros de clúster de base de datos de Aurora para el nuevo clúster principal, si es necesario: cuando se promociona un clúster secundario de base de datos para que asuma el rol principal, el grupo de parámetros del secundario se puede configurar de manera diferente que el del principal. Si es así, modifique el grupo de parámetros del clúster secundario de base de datos promocionado para que se ajuste a la configuración del clúster principal. Para saber cómo hacerlo, consulte Modificación de parámetros para una base de datos Aurora global.
- Configurar herramientas y opciones de monitoreo, como Amazon CloudWatch Events y alarmas – Configurar el clúster de base de datos promocionado con la misma capacidad de registro, alarmas, etc. según sea necesario para la base de datos global. Al igual que con los grupos de parámetros, la configuración de estas características no se hereda del clúster principal durante el proceso de transición. Algunas métricas de CloudWatch, como el retraso de la replicación, solo están disponibles para las regiones secundarias. Por lo tanto, una transición cambia la forma de ver esas métricas y configurar las alarmas en ellas, y podría requerir cambios en los paneles predefinidos. Para obtener más información acerca de los clústeres de base de datos de Aurora y la supervisión, consulte Supervisión de métricas de Amazon Aurora con Amazon CloudWatch.
- Configurar integraciones con otros servicios de AWS: si la base de datos global de Aurora se integra con servicios de AWS, como AWS Secrets Manager, AWS Identity and Access Management, Amazon S3 y AWS Lambda, asegúrese de configurar sus integraciones con estos servicios según sea necesario. Para obtener más información sobre la integración de bases de datos globales de Aurora con IAM, Amazon S3 y Lambda, consulte Uso de las bases de datos globales de Amazon Aurora con otros servicios de AWS. Para obtener más información sobre Secrets Manager, consulte Cómo automatizar la replicación de secretos en AWS Secrets Manager entre Regiones de AWS.

Si utiliza el punto de conexión del escritor de la base de datos global de Aurora, no es necesario cambiar la configuración de conexión en su aplicación. Compruebe que los cambios de DNS se hayan propagado y que pueda conectarse y realizar operaciones de escritura en el nuevo clúster principal. A continuación, podrá reanudar el funcionamiento completo de la aplicación.

Suponga que las conexiones de su aplicación utilizan el punto de conexión del clúster principal anterior, en lugar del punto de conexión del escritor global. En ese caso, asegúrese de cambiar la configuración de conexión de la aplicación para utilizar el punto de conexión del clúster del nuevo clúster principal. Si aceptó los nombres proporcionados al crear la base de datos global de Aurora, puede cambiar el punto de enlace quitando `-ro` de la cadena del punto de enlace del

clúster promocionado en la aplicación. Por ejemplo, el punto de enlace del clúster secundario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` se convierte en `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` cuando ese clúster se promueve a principal.

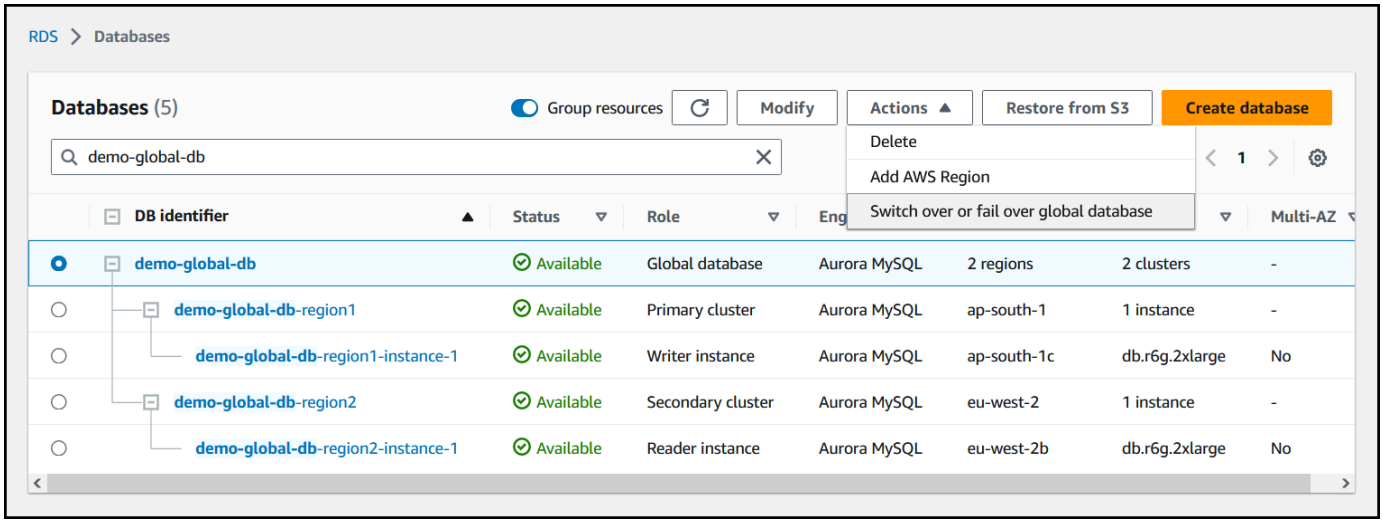
Si utiliza RDS Poxy, asegúrese de redirigir las operaciones de escritura de la aplicación al punto de conexión de lectura o escritura correspondiente del proxy asociado al nuevo clúster principal. Este punto de conexión proxy puede ser el punto de conexión predeterminado o un punto de conexión de lectura o escritura personalizado. Para obtener más información, consulte Cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales.

Puede realizar una transición de la base de datos global de Aurora mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

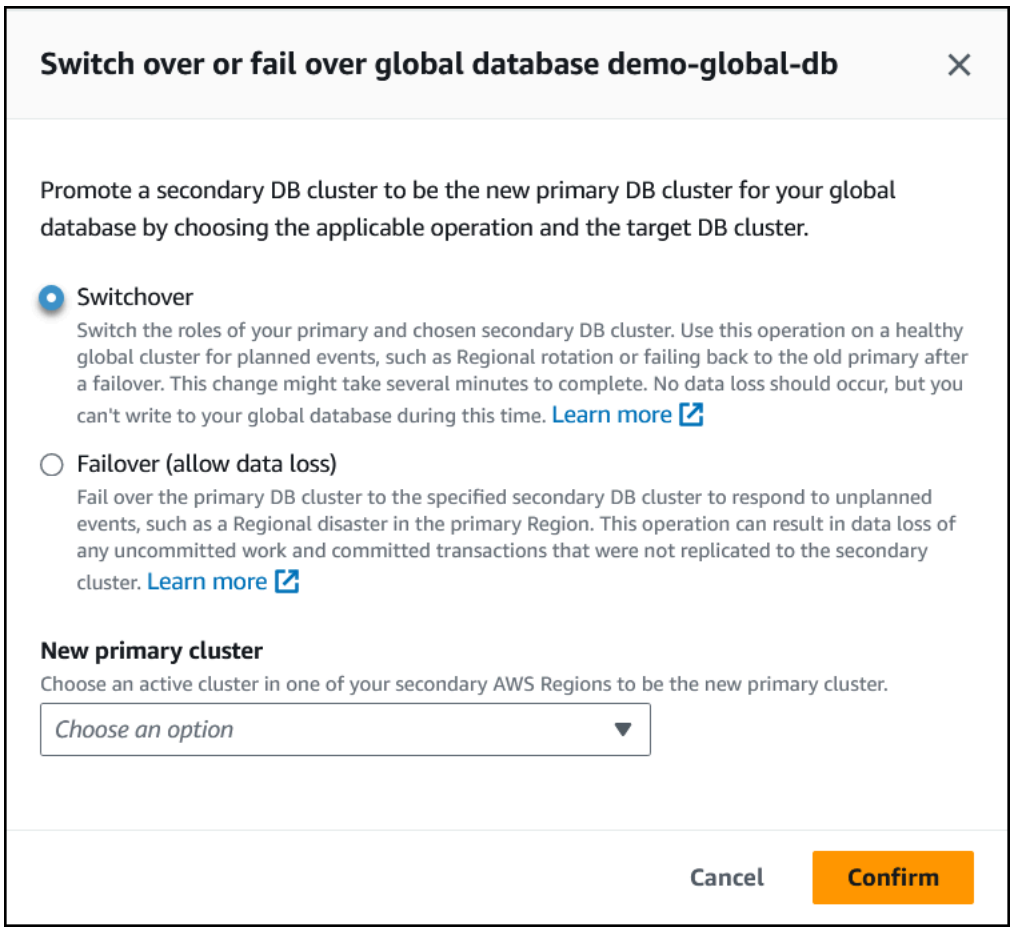
Consola

Para realizar la transición en la base de datos global de Aurora

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Bases de datos y busque la base de datos global de Aurora que tiene intención de someter al proceso de transición.
3. Elija Transición o conmutación por error de base de datos global en el menú Acciones.

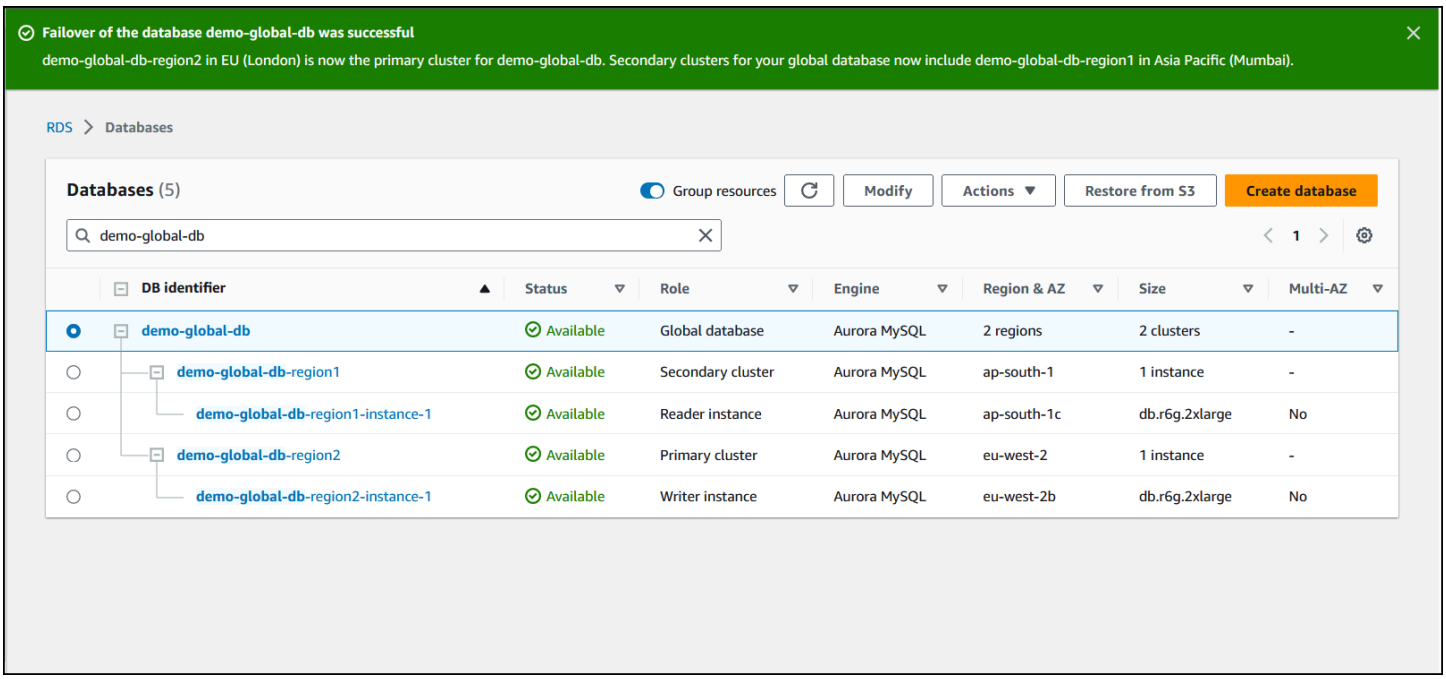


4. Elija Transición.



5. Para Nuevo clúster principal, elija un clúster activo en una de sus Regiones de AWS secundarias para que sea el nuevo clúster principal.
6. Elija Confirmar.

Cuando finalice la transición, podrá ver los clústeres de base de datos de Aurora y su estado actual en la lista Bases de datos, como se muestra en la siguiente imagen.



AWS CLI

Para realizar la transición en una base de datos global de Aurora

Utilice el comando de la CLI `switchover-global-cluster` para realizar una transición en la base de datos global de Aurora. Con el comando, pase valores para los siguientes parámetros.

- `--region`: especifique la Región de AWS donde se ejecuta el clúster de base de datos principal de la base de datos global de Aurora.
- `--global-cluster-identifier` – Especifique el nombre de la base de datos global de Aurora.
- `--target-db-cluster-identifier` – Especifique el nombre de recurso de Amazon (ARN) del clúster de base de datos de Auroraque desea promover para que sea el principal de la base de datos global de Aurora.

Para Linux, macOS o Unix:

```
aws rds --region region_of_primary \
   switchover-global-cluster --global-cluster-identifier global_database_id \
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

Para Windows:

```
aws rds --region region_of_primary ^
   switchover-global-cluster --global-cluster-identifier global_database_id ^
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

API de RDS

Para realizar una transición de la base de datos global de Aurora, ejecute la operación de la API SwitchoverGlobalCluster.

Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada

En raras ocasiones, su base de datos global de Aurora puede experimentar una interrupción inesperada en su Región de AWS principal. Si esto sucede, el clúster principal de base de datos de Aurora y su nodo de escritor no estarán disponibles, y cesará la replicación entre el clúster principal y el secundario. Para minimizar el tiempo de inactividad (RTO) y la pérdida de datos (RPO), puede trabajar rápidamente para realizar una conmutación por error entre regiones.

La base de datos global de Aurora tiene dos métodos de conmutación por error que puede usar en una situación de recuperación ante desastres:

- Conmutación por error administrada: este método se recomienda para la recuperación de desastres. Si utiliza este método, Aurora vuelve a añadir automáticamente la antigua región principal a la base de datos global como región secundaria cuando vuelve a estar disponible. De este modo, se mantiene la topología original del clúster global. Para obtener información sobre cómo utilizar este método, consulte Ejecución de la conmutación por error administrada para bases de datos globales de Aurora.
- Conmutación por error manual: este método alternativo se puede utilizar cuando la conmutación por error administrada no es una opción, por ejemplo, cuando las regiones principal y secundaria utilizan versiones de motor incompatibles. Para obtener información sobre cómo utilizar este método, consulte Ejecución de la conmutación por error manual para bases de datos globales de Aurora.

> **Important**
>
> Ambos métodos de conmutación por error pueden provocar la pérdida de datos de transacciones de escritura que no se replicaron en el secundario elegido antes de que se produjera el evento de conmutación por error. Sin embargo, el proceso de recuperación que

promueve una instancia de base de datos del clúster de base de datos secundario elegido a instancia de base de datos principal de escritor garantiza que los datos estén en un estado coherente desde el punto de vista de las transacciones. Las conmutaciones por error también son susceptibles de provocar problemas de cerebro dividido.

Ejecución de la conmutación por error administrada para bases de datos globales de Aurora

Este enfoque tiene por objeto garantizar la continuidad empresarial en caso de que se produzca un verdadero desastre regional o una interrupción total del nivel de servicio.

Durante una conmutación por error administrada, el clúster secundario de la región secundaria elegida se convierte en el nuevo clúster principal. El clúster secundario elegido promueve uno de sus nodos de solo de lectura al estado de escritor completo. Este paso permite que el clúster asuma el rol de clúster principal. La base de datos no estará disponible durante un breve periodo, mientras el clúster asume su nuevo rol. Tan pronto como la región principal antigua esté en buen estado y vuelva a estar disponible, Aurora la volverá a agregar automáticamente al clúster global como región secundaria. De este modo, se mantiene la topología de replicación existente de la base de datos global de Aurora.

> **Note**
>
> Para administrar las ranuras de replicación para Aurora PostgreSQL después de realizar una conmutación por error, consulte Administración de ranuras lógicas para Aurora PostgreSQL .

> **Note**
>
> Solo puede realizar conmutaciones por error administradas entre regiones en la base de datos global de Aurora si los clústeres de base de datos principal y secundario tienen las mismas versiones principal, secundaria y de nivel de parche del motor. Según el motor y las versiones del motor, es posible que los niveles de parche deban ser idénticos o diferentes. Para obtener una lista de los motores y las versiones de motores que permiten estas operaciones entre clústeres principales y secundarios con diferentes niveles de parches, consulte Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones. Antes de iniciar la conmutación por error, compruebe las versiones del motor de su clúster global para asegurarse de que admiten la transición entre

regiones administrada y actualícelas, si es necesario. Si las versiones del motor requieren niveles de parches idénticos, pero ejecutan niveles de parche distintos, puede realizar la conmutación por error manualmente por medio de los pasos que se indican en Ejecución de la conmutación por error manual para bases de datos globales de Aurora.

La conmutación por error administrada no espera a que los datos se sincronicen entre la región secundaria elegida y la región principal actual. Como la base de datos global de Aurora replica los datos de forma asíncrona, es posible que no todas las transacciones se repliquen en la región AWS secundaria elegida antes de que se promocione para aceptar todas las capacidades de lectura/escritura.

Para garantizar que los datos estén en un estado uniforme, Aurora crea un nuevo volumen de almacenamiento para la antigua región principal una vez que se recupera. Antes de crear el nuevo volumen de almacenamiento en la región AWS, Aurora intenta tomar una instantánea del volumen de almacenamiento anterior en el punto en que se produjo el error. De esta forma, puede restaurar la instantánea y recuperar cualquiera de los datos que se hayan perdido de esta. Si esta operación se realiza correctamente, Aurora coloca la instantánea con el nombre `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`* en la sección de instantáneas de la Consola de administración de AWS. También puede usar el comando `describe-db-cluster-snapshots` de la AWS CLI o la operación de la API `DescribeDBClusterSnapshots` para ver los detalles de la instantánea.

Al iniciar una conmutación por error administrada, Aurora también intenta detener el tráfico de escritura a través de la capa de almacenamiento de Aurora, de alta disponibilidad. A este mecanismo lo denominamos “delimitación de escritura”. Si el proceso se realiza correctamente, Aurora emite un evento de RDS que le notifica que las escrituras se han detenido. En el improbable caso de que se produzcan varios fallos en las zonas de disponibilidad en una región, es posible que el proceso de escritura no se lleve a cabo a tiempo. En ese caso, Aurora emite un evento de RDS que le informa de que se ha agotado el tiempo de espera del proceso para detener las escrituras. Si se puede acceder al clúster principal anterior en la red, Aurora registra estos eventos allí. De lo contrario, Aurora registra los eventos en el nuevo clúster principal. Para obtener más información acerca de estos eventos, consulte Eventos de clúster de bases de datos. Dado que limitar la escritura es el mejor intento, es posible que las escrituras se acepten momentáneamente en la antigua región principal, lo que provocaría problemas de cerebro dividido.

Le recomendamos que realice las siguientes tareas antes de realizar una conmutación por error con la base de datos global de Aurora. Si lo hace, se minimiza la posibilidad de que se produzcan

problemas de cerebro dividido o de que se recuperen datos no replicados de la instantánea del clúster principal anterior.

- Para evitar que se envíen escrituras al clúster principal de la base de datos global de Aurora, desconecte las aplicaciones.
- Asegúrese de que todas las aplicaciones que se conecten al clúster de base de datos principal utilicen el punto de conexión del escritor global. Este punto de conexión tiene un valor que permanece igual incluso cuando una nueva región se convierte en el clúster principal debido a una transición o una conmutación por error. Aurora implementa medidas de seguridad adicionales para minimizar la posibilidad de pérdida de datos en las operaciones de escritura enviadas a través del punto de conexión global. Para obtener más información acerca de los puntos de conexión del escritor global, consulte Conexión a la base de datos global de Amazon Aurora.
- Si utiliza el punto de conexión del escritor global y sus capas de aplicaciones o redes almacenan en caché los valores de DNS, reduzca el tiempo de vida (TTL) de la memoria caché de DNS a un valor bajo, por ejemplo, 5 segundos. De esta forma, la aplicación registra rápidamente los cambios de DNS en el punto de conexión del escritor global. Aunque Aurora intenta bloquear las escrituras en la antigua región principal, no se garantiza que la acción tenga éxito. Al reducir la duración de la memoria caché del DNS, se reduce aún más la probabilidad de que se produzcan problemas de cerebro dividido. Como alternativa, puede comprobar el evento de RDS que le informa cuando Aurora observe los cambios de DNS en el punto de conexión del escritor global. De esta forma, puede validar que su aplicación también registró el cambio de DNS antes de reiniciar el tráfico de escritura de la aplicación.
- Compruebe los tiempos de retraso para todos los clústeres secundarios de base de datos de Aurora de la base de datos global de Aurora. La elección de la región secundaria con el menor retraso de replicación puede minimizar la pérdida de datos con respecto a la región principal que actualmente presenta errores.

  Para todas las versiones de bases de datos globales basadas en Aurora PostgreSQL y para las bases de datos globales basadas en Aurora MySQL a partir de las versiones de motor 3.04.0 y posteriores, o 2.12.0 y posteriores, utilice Amazon CloudWatch para ver la métrica `AuroraGlobalDBRPOLag` de todos los clústeres de base de datos secundarios. Para ver las versiones secundarias anteriores de las bases de datos globales basadas en Aurora MySQL, consulte, en cambio, la métrica `AuroraGlobalDBReplicationLag`. Estas métricas muestran el retraso (en milisegundos) de la replicación a un clúster secundario con respecto al clúster principal de base de datos.

Para obtener más información acerca de las métricas de CloudWatch para Aurora, consulte Métricas de nivel de clúster para Amazon Aurora.

Durante una conmutación por error administrada, el clúster secundario de base de datos elegido se promueve a su nuevo rol de clúster principal. Sin embargo, no hereda las diversas opciones de configuración del clúster principal de base de datos. Una falta de coincidencia en la configuración puede provocar problemas de rendimiento, incompatibilidades de carga de trabajo y otros comportamientos anómalos. Para evitar estos problemas, recomendamos que se resuelvan las diferencias entre los clústeres de bases de datos globales de Aurora para lo siguiente:

- Configure grupo de parámetros de clúster de base de datos de Aurora para el nuevo clúster principal, si es necesario: puede configurar los grupos de parámetros de clúster de base de datos de Aurora de forma independiente para cada clúster de Aurora de la base de datos global de Aurora. Por lo tanto, cuando se promueve un clúster secundario de base de datos para que asuma el rol principal, su grupo de parámetros puede configurarse de manera diferente que para el principal. Si es así, modifique el grupo de parámetros del clúster secundario de base de datos promocionado para que se ajuste a la configuración del clúster principal. Para saber cómo hacerlo, consulte Modificación de parámetros para una base de datos Aurora global.
- Configurar herramientas y opciones de monitoreo, como Amazon CloudWatch Events y alarmas – Configurar el clúster de base de datos promocionado con la misma capacidad de registro, alarmas, etc. según sea necesario para la base de datos global. Al igual que con los grupos de parámetros, la configuración de estas características no se hereda del clúster principal durante el proceso de conmutación por error. Algunas métricas de CloudWatch, como el retraso de la replicación, solo están disponibles para las regiones secundarias. Por lo tanto, una conmutación por error cambia la forma de ver esas métricas y configurar las alarmas en ellas, y podría requerir cambios en los paneles predefinidos. Para obtener más información acerca de la supervisión de clústeres de base de datos de Aurora, consulte Supervisión de métricas de Amazon Aurora con Amazon CloudWatch.
- Configure integraciones con otros servicios de AWS: si la base de datos global de Aurora se integra con otros servicios de AWS, como AWS Secrets Manager, AWS Identity and Access Management, Amazon S3 y AWS Lambda, debe asegurarse de que están configurados según sea necesario para que se pueda acceder desde cualquier región secundaria. Para obtener más información sobre la integración de bases de datos globales de Aurora con IAM, Amazon S3 y Lambda, consulte Uso de las bases de datos globales de Amazon Aurora con otros servicios

de AWS. Para obtener más información sobre Secrets Manager, consulte Cómo automatizar la replicación de secretos en AWS Secrets Manager entre Regiones de AWS.

Por lo general, el clúster secundario elegido asume el rol principal en cuestión de minutos. En cuanto la instancia de base de datos del escritor de la nueva región principal esté disponible, podrá conectar sus aplicaciones a ella y reanudar sus cargas de trabajo. Una vez que Aurora promueve el nuevo clúster principal, reconstruye automáticamente todos los clústeres regionales secundarios adicionales.

Como las bases de datos globales de Aurora utilizan la replicación asíncrona, el retraso de la replicación en cada región secundaria puede variar. Aurora reconstruye estas regiones secundarias para que tengan exactamente los mismos datos de un momento dado que el nuevo clúster de región principal. La duración de la tarea de reconstrucción completa puede tardar entre unos minutos y varias horas, según el tamaño del volumen de almacenamiento y la distancia entre las regiones. Cuando los clústeres de la región secundaria terminen de reconstruirse a partir de la nueva región principal, estarán disponibles para el acceso de lectura.

Tan pronto como se promocione y esté disponible el nuevo escritor principal, el clúster de la nueva región principal podrá gestionar las operaciones de lectura y escritura de la base de datos global de Aurora.

Si utiliza el punto de conexión global, no es necesario cambiar la configuración de conexión en su aplicación. Compruebe que los cambios de DNS se hayan propagado y que pueda conectarse y realizar operaciones de escritura en el nuevo clúster principal. A continuación, podrá reanudar el funcionamiento completo de la aplicación.

Si no utiliza el punto de conexión global, asegúrese de cambiar el punto de conexión de la aplicación para que utilice el punto de conexión del clúster para el clúster de base de datos principal recién promocionado. Si aceptó los nombres proporcionados al crear la base de datos global de Aurora, puede cambiar el punto de enlace quitando `-ro` de la cadena del punto de enlace del clúster promocionado en la aplicación.

Por ejemplo, el punto de enlace del clúster secundario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` se convierte en `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` cuando ese clúster se promueve a principal.

Si utiliza RDS Poxy, asegúrese de redirigir las operaciones de escritura de la aplicación al punto de conexión de lectura o escritura correspondiente del proxy asociado al nuevo clúster principal. Este punto de conexión proxy puede ser el punto de conexión predeterminado o un punto de conexión

de lectura o escritura personalizado. Para obtener más información, consulte Cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales.

Para restaurar la topología original del clúster de base de datos global, Aurora monitoriza la disponibilidad de la antigua región principal. Tan pronto como la región esté en buen estado y vuelva a estar disponible, Aurora la volverá a agregarla automáticamente al clúster global como región secundaria. Antes de crear el nuevo volumen de almacenamiento en la antigua región principal, Aurora intenta tomar una instantánea del volumen de almacenamiento anterior en el punto en que se produjo el error. Lo hace para que pueda usarla para recuperar cualquiera de los datos perdidos. Si esta operación se realiza correctamente, Aurora crea una instantánea con el nombre `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`*. Puede encontrar esta instantánea en la sección Instantáneas de la Consola de administración de AWS. También puede ver esta instantánea en la información devuelta por la operación de la API DescribeDBClusterSnapshots.

> **Note**
>
> La instantánea del volumen de almacenamiento anterior es una instantánea del sistema que está sujeta al período de retención de la copia de seguridad configurado en el clúster principal anterior. Para conservar esta instantánea más allá del período de retención, puede copiarla para guardarla como una instantánea manual. Para obtener más información sobre la copia de instantáneas, incluido el precio, consulte Copia de una instantánea de clúster de base de datos.

Una vez restaurada la topología original, puede conmutar por recuperación la base de datos global a la región principal original realizando una operación de transición cuando sea más conveniente para su empresa y su carga de trabajo. Para ello, siga los pasos que se indican en Ejecución de transiciones para bases de datos globales de Amazon Aurora.

Puede realizar una conmutación por error con la base de datos global de Aurora mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

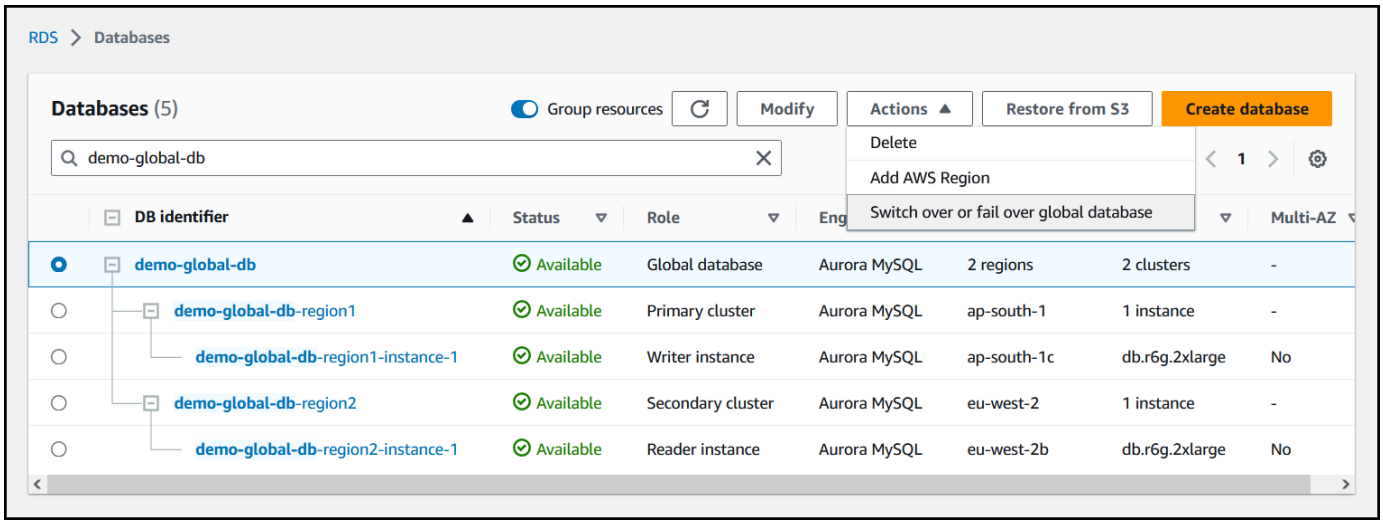
Consola

Para realizar la conmutación por error administrada en la base de datos global de Aurora

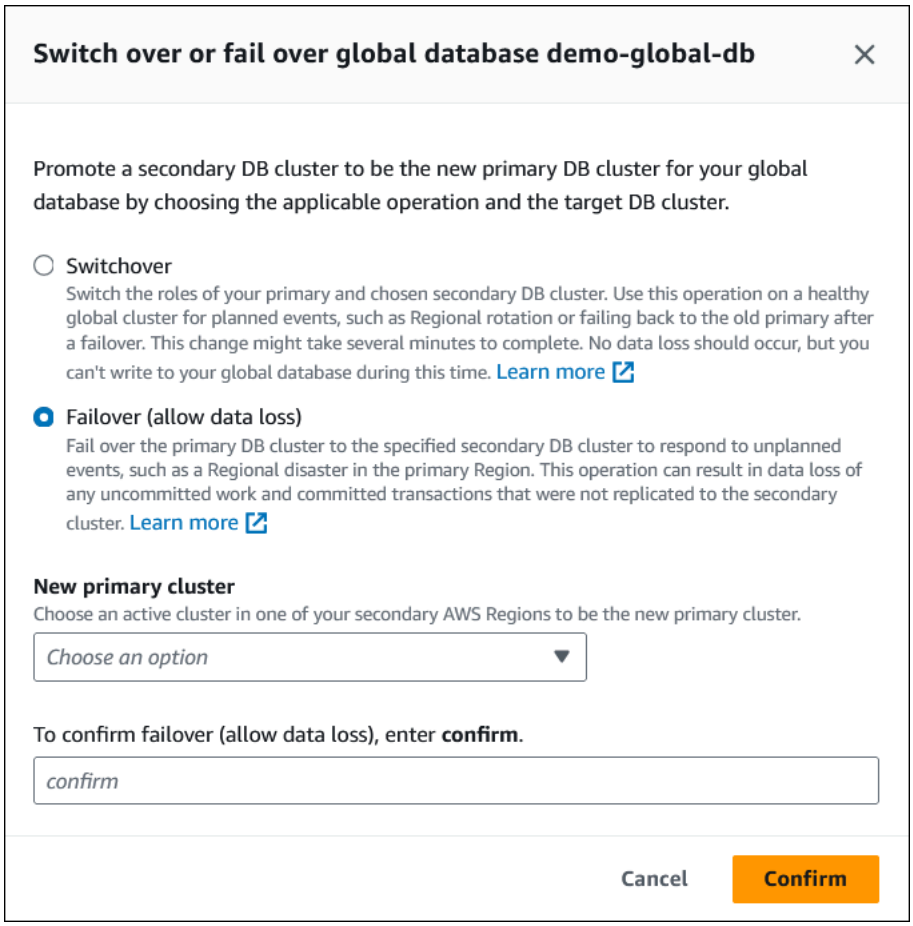
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. Seleccione Bases de datos y busque la base de datos global de Aurora en la que desea llevar a cabo la conmutación por error.

3. Elija Transición o conmutación por error de base de datos global en el menú Acciones.

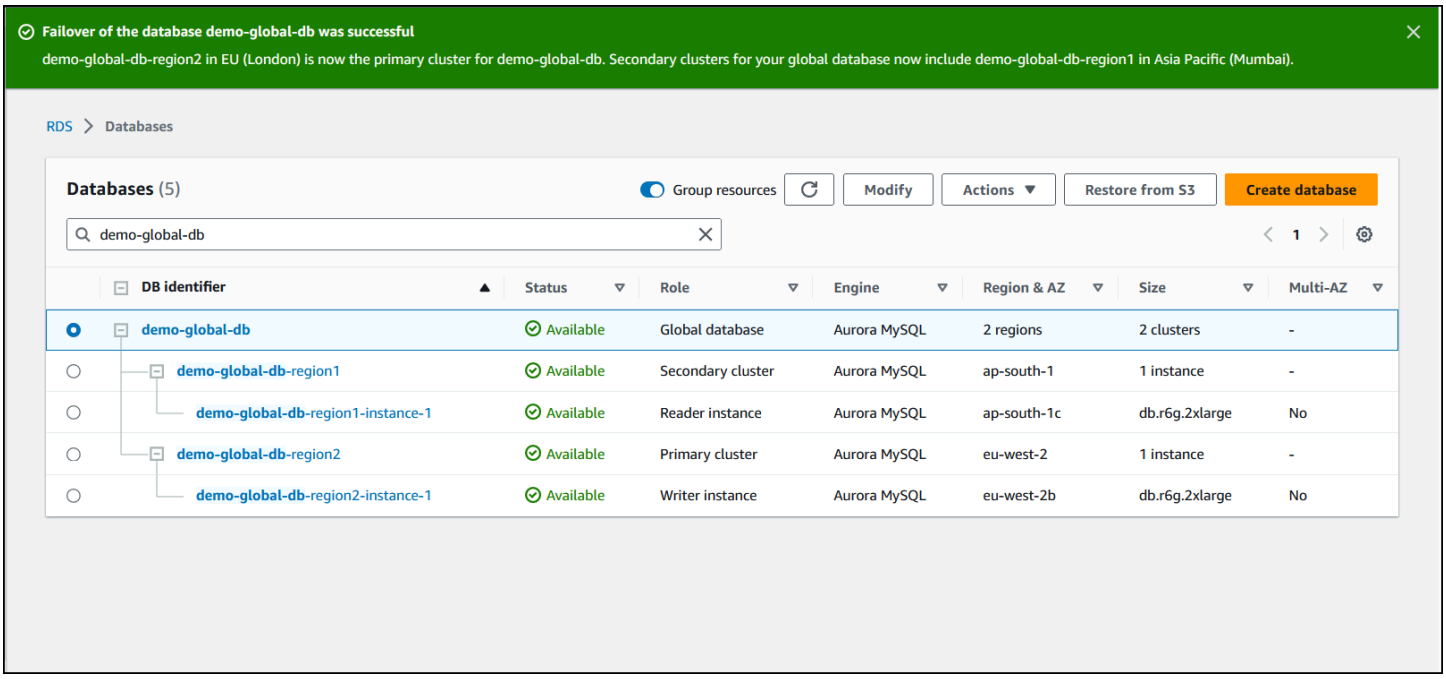


4. Elija Conmutación por error (permitir la pérdida de datos).



5. Para Nuevo clúster principal, elija un clúster activo en una de sus Regiones de AWS secundarias para que sea el nuevo clúster principal.
6. Introduzca **`confirm`** y, a continuación, elija Confirmar.

Cuando finalice la conmutación por error, podrá ver los clústeres de base de datos de Aurora y su estado actual en la lista Bases de datos, como se muestra en la siguiente imagen.



AWS CLI

Para realizar la conmutación por error administrada en una base de datos global de Aurora

Utilice el comando de la CLI `failover-global-cluster` para realizar una conmutación por error con la base de datos global de Aurora. Con el comando, pase valores para los siguientes parámetros.

- `--region`: especifique la Región de AWS donde se ejecuta el clúster secundario de la base de datos que desea que sea el nuevo clúster principal de la base de datos global de Aurora.
- `--global-cluster-identifier` – Especifique el nombre de la base de datos global de Aurora.
- `--target-db-cluster-identifier`: especifique el nombre de recurso de Amazon (ARN) del clúster de la base de datos de Aurora que desea promover para que sea el clúster principal de la base de datos global de Aurora.

- `--allow-data-loss`: indique explícitamente que es una operación de conmutación por error en lugar de una operación de transición. Una operación de conmutación por error puede tener como resultado la pérdida de algunos datos si los componentes de replicación asincrónica no han completado el envío de todos los datos replicados a la región secundaria.

Para Linux, macOS o Unix:

```
aws rds --region region_of_selected_secondary \
   failover-global-cluster --global-cluster-identifier global_database_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Para Windows:

```
aws rds --region region_of_selected_secondary ^
   failover-global-cluster --global-cluster-identifier global_database_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

API de RDS

Para realizar una conmutación por error con la base de datos global de Aurora, ejecute la operación de API FailoverGlobalCluster.

Ejecución de la conmutación por error manual para bases de datos globales de Aurora

En algunos casos, es posible que no pueda utilizar el proceso de conmutación por error administrada. Un ejemplo es si los clústeres de bases de datos principal y secundario no ejecutan versiones de motor compatibles. En este caso, puede seguir este proceso manual para realizar una conmutación por error en la región secundaria de destino.

> **Tip**
>
> Recomendamos que comprenda este proceso antes de usarlo. Tenga un plan listo para continuar rápidamente a la primera señal de un problema en toda la región. Puede utilizar Amazon CloudWatch periódicamente para realizar el seguimiento de los tiempos de retraso de los clústeres secundarios y estar preparado para identificar la región secundaria con el menor retraso de replicación. Asegúrese de probar su plan para verificar que sus

procedimientos sean completos y precisos, y que el personal esté formado para realizar una conmutación por error de recuperación de desastres antes de que realmente suceda.

Para realizar una conmutación por error manual en un clúster secundario después de una interrupción no planificada en la región principal

1. Deje de emitir instrucciones DML y otras operaciones de escritura en el clúster de base de datos de Aurora principal en la Región de AWS con la interrupción.
2. Identifique un clúster de base de datos de Aurora de una Región de AWS secundaria para usarlo como un nuevo clúster de base de datos principal. Si tiene dos (o más) Regiones de AWS secundarias en la base de datos global de Aurora, elija el clúster secundario que tenga el menor retraso de replicación.
3. Desconecte el clúster secundario de la base de datos global de Aurora elegido.

   La eliminación de un clúster de base de datos secundario de una base de datos global de Aurora detiene inmediatamente la replicación de la principal a esta secundaria y la promueve a un clúster de base de datos de Aurora aprovisionado e independiente con capacidades completas de lectura y escritura. Todavía está disponible cualquier otro clúster secundario de base de datos de Aurora asociado con el clúster principal de la región con la interrupción y puede aceptar llamadas desde la aplicación. También consumen recursos. Debido a que está recreando la base de datos global de Aurora, elimine los otros clústeres de base de datos secundarios antes de crear la nueva base de datos global de Aurora en los siguientes pasos. De este modo, se evitan incoherencias con respecto a los datos entre los clústeres de la base de datos global de Aurora (problemas del tipo cerebro dividido).

   Para obtener más información sobre los pasos para desasociar clústeres, consulte Eliminación de un clúster de una base de datos global de Amazon Aurora.
4. Vuelva a configurar la aplicación para enviar todas las operaciones de escritura a este clúster de base de datos de Aurora ahora independiente con su nuevo punto de enlace. Si aceptó los nombres proporcionados al crear la base de datos global de Aurora, puede cambiar el punto de enlace quitando `-ro` de la cadena del punto de enlace del clúster en la aplicación.

   Por ejemplo, el punto de enlace del clúster secundario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` se convierte en `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` cuando ese clúster se desasocia de las bases de datos globales de Aurora.

Este clúster de base de datos de Aurora se convierte en el clúster principal de una nueva base de datos global de Aurora cuando comienza a agregarle regiones en el siguiente paso.

Si utiliza RDS Poxy, asegúrese de redirigir las operaciones de escritura de la aplicación al punto de conexión de lectura o escritura correspondiente del proxy asociado al nuevo clúster principal. Este punto de conexión proxy puede ser el punto de conexión predeterminado o un punto de conexión de lectura o escritura personalizado. Para obtener más información, consulte Cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales.

5. Agregue una Región de AWS al clúster de base de datos. Al hacerlo, comienza el proceso de reproducción de clúster principal a secundario. Para obtener más información sobre los pasos para agregar una región, consulte Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.
6. Agregue más Regiones de AWS según sea necesario para recrear la topología necesaria para admitir su aplicación.

Asegúrese de que las escrituras de la aplicación se envíen al clúster de base de datos de Aurora correcto antes, durante y después de realizar estos cambios. De este modo, se evitan incoherencias con respecto a los datos entre los clústeres de la base de datos global de Aurora (problemas del tipo cerebro dividido).

Si realizó una reconfiguración en respuesta a una interrupción en una Región de AWS, puede volver a hacer que la Región de AWS sea de nuevo la región principal después de que se resuelva la interrupción. Para ello, añada la antigua Región de AWS a la nueva base de datos global y, a continuación, utilice el proceso de transición para cambiar su rol. La base de datos global de Aurora debe utilizar una versión de Aurora PostgreSQL o Aurora MySQL que admita la transiciones. Para obtener más información, consulte Ejecución de transiciones para bases de datos globales de Amazon Aurora.

Administración de RPO para bases de datos globales basadas en Aurora PostgreSQL–

Con una base de datos global basada en Aurora PostgreSQL, puede administrar el objetivo de punto de recuperación (RPO) de la base de datos global de Aurora mediante el parámetro `rds.global_db_rpo`. El RPO representa la cantidad máxima de datos que se pueden perder en caso de interrupción.

Cuando establece un RPO para la base de datos global basada en Aurora PostgreSQL–, Aurora monitorea el tiempo de retraso de RPO de todos los clústeres secundarios para asegurarse de que al menos un clúster secundario permanezca dentro de la ventana de RPO de destino. El tiempo de retraso de RPO es otra métrica basada en tiempo.

El objetivo de punto de recuperación (RPO) se utiliza cuando la base de datos reanuda las operaciones en una nueva Región de AWS después de una conmutación por error. Aurora evalúa los tiempos de retraso de RPO y RPO para confirmar (o bloquear) transacciones en el clúster principal de la siguiente manera:

- Confirma la transacción si al menos un clúster secundario de base de datos tiene un tiempo de retraso de RPO menor que el RPO.
- Bloquea la transacción si todos los clústeres secundarios de base de datos tienen tiempos de retraso de RPO mayores que el RPO. También registra el evento en el archivo de registro de PostgreSQL y emite eventos de “espera” que muestran las sesiones bloqueadas.

En otras palabras, si todos los clústeres secundarios están detrás del RPO fijado, Aurora detiene las transacciones en el clúster principal hasta que al menos uno de los clústeres secundarios se recupere. Las transacciones en pausa se reanudan y confirman tan pronto como el tiempo de retraso de al menos un clúster secundario de base de datos sea menor que el RPO. El resultado es que ninguna transacción se puede confirmar hasta que se cumpla el RPO.

El parámetro `rds.global_db_rpo` es dinámico. Si decide que no quiere que todas las transacciones de escritura se detengan hasta que el retraso disminuya lo suficiente, puede restablecerlo rápidamente. En este caso, Aurora reconoce e implementa el cambio tras un breve retraso.

> **Important**
>
> En una base de datos global con solo dos regiones de AWS, recomendamos mantener el valor predeterminado del parámetro `rds.global_db_rpo` en el grupo de parámetros de la región secundaria. De lo contrario, si se realiza una conmutación por error en debido a una pérdida de la región de AWS principal, Aurora podría pausar las transacciones. En su lugar, espere a que Aurora vuelva a crear el clúster en la región de AWS antigua que ha fallado antes de cambiar este parámetro para aplicar un RPO máximo.

Si establece este parámetro como se describe a continuación, también puede supervisar las métricas que genera. Puede hacerlo mediante `psql` u otra herramienta para consultar el clúster principal de la base de datos global de Aurora y obtener información detallada sobre las operaciones de la base de datos global basada en Aurora PostgreSQL–. Para saber cómo hacerlo, consulte Supervisión de bases de datos globales basadas en Aurora PostgreSQL.

Temas

- Establecimiento del objetivo de punto de recuperación
- Visualización del objetivo de punto de recuperación
- Desactivación del objetivo de punto de recuperación

Establecimiento del objetivo de punto de recuperación

El parámetro de `rds.global_db_rpo` controla la configuración de RPO para una base de datos PostgreSQL. Este parámetro es compatible con Aurora PostgreSQL. Valores válidos para el rango de `rds.global_db_rpo` de 20 segundos a 2 147 483 647 segundos (68 años). Elija un valor realista para satisfacer las necesidades de su negocio y su caso de uso. Por ejemplo, es posible que desee permitir hasta 10 minutos para su RPO, en cuyo caso establece el valor en 600.

Puede establecer este valor para la base de datos global basada en Aurora PostgreSQL–mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para establecer el RPO

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija el clúster principal de la base de datos global de Aurora y abra la pestaña Configuration (Configuración) para buscar su grupo de parámetros de clúster de base de datos. Por ejemplo, el grupo de parámetros predeterminado para un clúster principal de base de datos que ejecuta Aurora PostgreSQL 11.7 es `default.aurora-postgresql11`.

   Los grupos de parámetros no se pueden editar directamente. En su lugar, puede hacer lo siguiente:

   - Cree un grupo de parámetros de clúster de base de datos personalizado con el grupo de parámetros predeterminado apropiado como punto de partida. Por ejemplo, cree un grupo

de parámetros de clúster de base de datos personalizado basado en el `default.aurora-postgresql11`.

- En su grupo de parámetros de base de datos personalizado, establezca el valor del parámetro rds.global_db_rpo para satisfacer su caso de uso. Los valores válidos van desde 20 segundos hasta el valor entero máximo de 2 147 483 647 (68 años).
- Aplique el grupo de parámetros de clúster de base de datos modificado a su clúster de base de datos de Aurora

Para obtener más información, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

AWS CLI

Para establecer el parámetro `rds.global_db_rpo`, utilice el comando de CLI modify-db-cluster-parameter-group. En el comando, especifique el nombre del grupo de parámetros del clúster principal y los valores para el parámetro RPO.

En el ejemplo siguiente se establece el RPO en 600 segundos (10 minutos) para el grupo de parámetros de clúster principal de base de datos denominado `my_custom_global_parameter_group`.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Para Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

API de RDS

Para modificar el parámetro `rds.global_db_rpo`, utilice la operación ModifyDBClusterParameterGroup de la API de Amazon RDS.

Visualización del objetivo de punto de recuperación

El objetivo de punto de recuperación (RPO) de una base de datos global se almacena en el parámetro `rds.global_db_rpo` para cada clúster de base de datos. Puede conectarse al punto de enlace del clúster secundario que desea ver y utilizar `psql` para consultar la instancia de este valor.

```
db-name=>show rds.global_db_rpo;
```

Si no se establece este parámetro, la consulta devuelve lo siguiente:

```
rds.global_db_rpo
-------------------
 -1
(1 row)
```

La siguiente respuesta es de un clúster secundario de base de datos que tiene una configuración de RPO de 1 minuto.

```
rds.global_db_rpo
-------------------
 60
(1 row)
```

También puede utilizar la CLI para obtener valores para averiguar si `rds.global_db_rpo` está activo en cualquiera de los clústeres de base de datos de Aurora mediante el uso de la CLI para obtener los valores de todos los parámetros `user` del clúster.

Para Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameters \
 --db-cluster-parameter-group-name lab-test-apg-global \
 --source user
```

Para Windows:

```
aws rds describe-db-cluster-parameters ^
 --db-cluster-parameter-group-name lab-test-apg-global *
 --source user
```

El comando devuelve un resultado similar al siguiente para todos los parámetros `user` que no son `default-engine` o parámetros de clúster de base de datos de `system`.

```
{
    "Parameters": [
        {
            "ParameterName": "rds.global_db_rpo",
            "ParameterValue": "60",
            "Description": "(s) Recovery point objective threshold, in seconds, that
 blocks user commits when it is violated.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "20-2147483647",
            "IsModifiable": true,
            "ApplyMethod": "immediate",
            "SupportedEngineModes": [
                "provisioned"
            ]
        }
    ]
}
```

Para obtener más información sobre la visualización de parámetros del grupo de parámetros de clúster, consulte Visualización de los valores de parámetros de un grupo de parámetros de clúster de base de datos en Amazon Aurora.

Desactivación del objetivo de punto de recuperación

Para desactivar el RPO, restablezca el parámetro `rds.global_db_rpo`. Puede restablecer los parámetros mediante la Consola de administración de AWS, la AWS CLI o la API RDS.

Consola

Para deshabilitar el RPO

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Parameter groups (Grupos de parámetros).
3. En la lista, elija el grupo de parámetros de clúster de base de datos principal.
4. Elija Edit parameters (Editar parámetros).

5. Elija la casilla situada junto al parámetro rds.global_db_rpo.
6. Elija Restablecer.
7. Cuando la pantalla muestre Restablecer parámetros en el grupo de parámetros de base de datos, elija Restablecer parámetros.

Para obtener más información sobre cómo restablecer un parámetro con la consola, consulte Modificación de los parámetros en un grupo de parámetros de clúster de base de datos en Amazon Aurora.

AWS CLI

Para restablecer el parámetro `rds.global_db_rpo`, utilice el comando reset-db-cluster-parameter-group.

Para Linux, macOS o Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name global_db_cluster_parameter_group \
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

Para Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name global_db_cluster_parameter_group ^
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

API de RDS

Para restablecer el parámetro `rds.global_db_rpo`, utilice la operación ResetDBClusterParameterGroup de la API de Amazon RDS.

Resiliencia entre regiones para clústeres secundarios de una base de datos global

Las versiones 16.6, 15.10, 14.15, 13.18, 12.22 o posteriores y las versiones de Aurora MySQL 3.09 o posteriores contienen mejoras de disponibilidad que permiten que las réplicas de lectura de regiones secundarias mantengan la continuidad del servicio durante eventos no planificados, como errores de hardware, interrupciones de la red entre regiones de AWS, grandes volúmenes de transferencias de datos entre los clústeres, etc.

Aunque las réplicas de lectura siguen disponibles para las solicitudes de las aplicaciones, es posible que el retraso en la replicación siga aumentando hasta que se resuelva el evento imprevisto. Puede supervisar el desfase entre los clústeres principal y secundario mediante la métrica de `AuroraGlobalDBProgressLag` CloudWatch. Para medir el retraso de extremo a extremo, incluido cualquier retraso entre el volumen del clúster y las instancias de base de datos del clúster secundario, agregue los valores de las métricas de CloudWatch `AuroraGlobalDBProgressLag` y `AuroraReplicaLag`. Para obtener más información sobre las métricas, consulte Referencia de métricas para Amazon Aurora.

La disponibilidad de lectura de la base de datos global para Aurora MySQL y versiones anteriores de Aurora PostgreSQL puede verse afectada durante estos eventos no planificados.

Para obtener más información acerca de las nuevas características de Aurora PostgreSQL 16.6, 15.10, 14.15, 13.18 y 12.22, consulte PostgreSQL 16.6 en las notas de la versión de Aurora PostgreSQL.

Para obtener más información acerca de nuevas características en las versiones 3.09 y posteriores de Aurora MySQL, consulte Actualizaciones de motor de base de datos para la versión 3 de Amazon Aurora MySQL en las notas de la versión de Aurora MySQL.

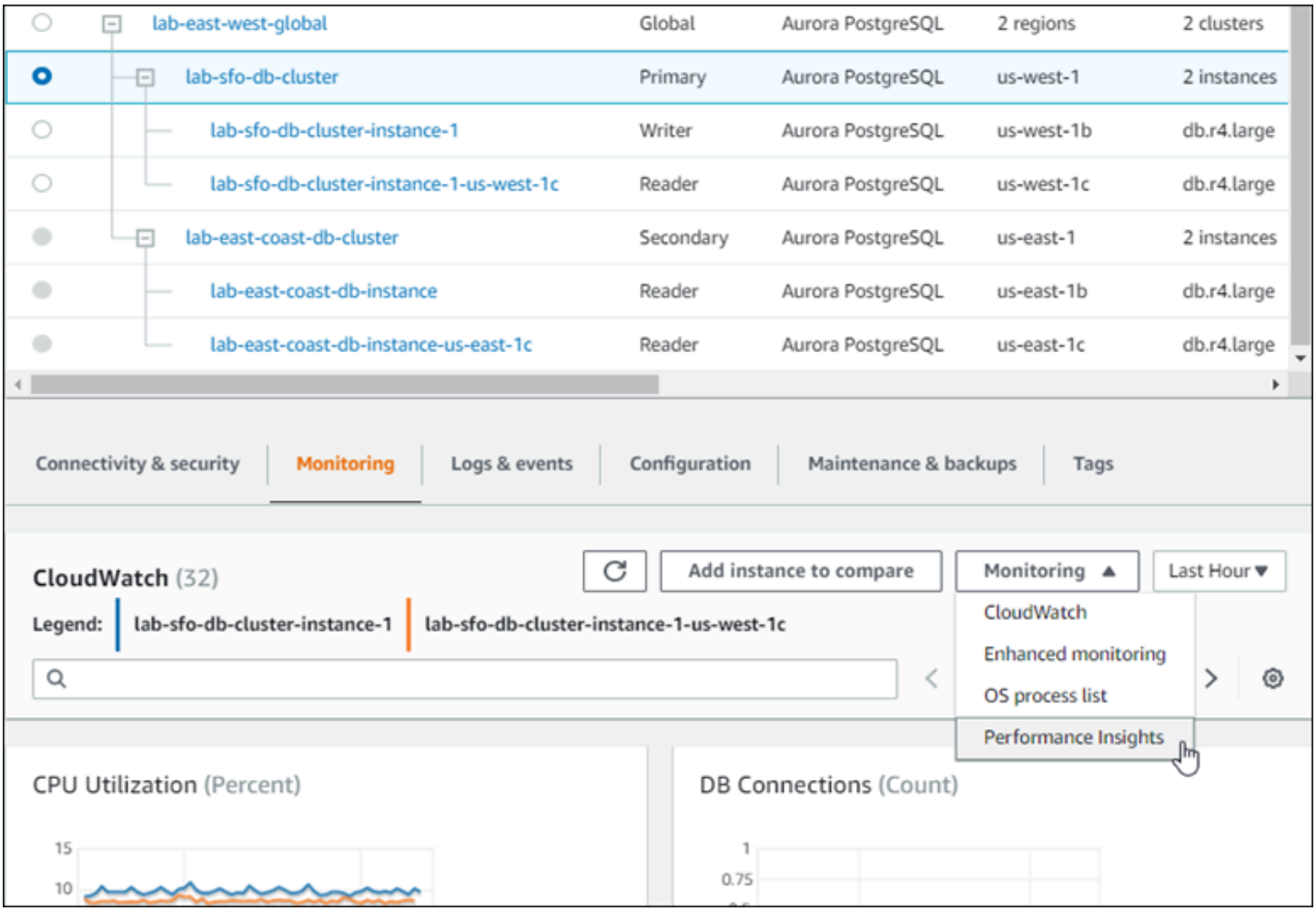
Monitoreo de una base de datos global de Amazon Aurora

Al crear los clústeres de base de datos Aurora que componen la base de datos Aurora global, puede elegir muchas opciones que le permitan supervisar el rendimiento del clúster de base de datos. Las opciones incluyen:

- Amazon RDS Información sobre rendimiento – Permite el esquema de rendimiento en el motor de base de datos Aurora subyacente. Para obtener más información sobre la información sobre rendimiento y bases de datos Aurora globales, consulte Supervisión de una base de datos Amazon Aurora global con Amazon RDS la información sobre rendimiento.
- Supervisión mejorada – Genera métricas para la utilización de procesos o subprocesos en la CPU. Para obtener más información sobre la supervisión mejorada, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.
- Amazon CloudWatch Logs – Publica los tipos de registro especificados en CloudWatch Logs. Los registros de errores se publican de forma predeterminada, pero puede elegir otros registros específicos del motor de base de datos Aurora.
  - Para los clústeres de base de datos Aurora basados en Aurora MySQL–, puede exportar el registro de auditoría, el registro general y el registro de consulta lenta.

  - En el caso de clústeres de base de datos de Aurora basados en Aurora PostgreSQL, puede exportar el registro de PostgreSQL.
- Para bases de datos globales basadas en Aurora MySQL, puede consultar determinadas tablas `information_schema` para comprobar el estado de la base de datos global de Aurora y sus instancias. Para aprender a hacerlo, consulte Supervisión de las bases de datos globales basadas en Aurora MySQL.
- Para bases de datos globales basadas en Aurora PostgreSQL, puede utilizar funciones específicas para comprobar el estado de la base de datos global de Aurora y sus instancias. Para saber cómo hacerlo, consulte Supervisión de bases de datos globales basadas en Aurora PostgreSQL.

La siguiente captura de pantalla muestra algunas de las opciones disponibles en la pestaña Supervisión de un clúster de base de datos Aurora principal en una base de datos Aurora global.



Para obtener más información, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Supervisión de una base de datos Amazon Aurora global con Amazon RDS la información sobre rendimiento

Puede utilizar Amazon RDS Información sobre seguimiento para sus bases de datos globales Aurora. Esta función se habilita individualmente, para cada clúster de base de datos Aurora de la base de datos Aurora global. Para ello, elija Enable Performance Insights (Habilitar información sobre rendimiento) en la sección Additional configuration (Configuración adicional) de la página Crear base de datos. O bien, puede modificar los clústeres de base de datos de Aurora para utilizar esta característica después de que estén en funcionamiento. Puede habilitar o desactivar la información sobre rendimiento para cada clúster que forma parte de la base de datos Aurora global.

Los informes creados por la información sobre rendimiento se aplican a cada clúster de la base de datos global. Cuando agregue una nueva Región de AWS secundaria a una base de datos global de Aurora que ya está utilizando Información sobre rendimiento, habilítela en el clúster que se agregó recientemente. No hereda la configuración de Performance Insights de la base de datos global existente.

Puede cambiar Regiones de AWS mientras ve la página de Información sobre rendimiento para una instancia de base de datos que está conectada a una base de datos global. Sin embargo, es posible que no vea la información sobre rendimiento de inmediato después del cambio de Regiones de AWS. Aunque las instancias de base de datos pueden tener nombres idénticos en cada Región de AWS, la URL de Información sobre rendimiento asociada es distinta para cada instancia de base de datos. Después de cambiar las Regiones de AWS, elija el nombre de la instancia de base de datos de nuevo en el panel de navegación de Información sobre rendimiento.

Para las instancias de base de datos asociadas a una base de datos global, los factores que afectan al rendimiento pueden ser distintos en cada Región de AWS. Por ejemplo, las instancias de base de datos en cada Región de AWS pueden tener una capacidad diferente.

Para obtener más información sobre el uso de la información sobre rendimiento, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

Supervisión de bases de datos globales Aurora mediante las secuencias de actividad de base de datos

Utilizando la característica de secuencias de actividad de base de datos, puede supervisar y establecer alarmas para auditar la actividad en los clústeres de base de datos de su base de datos global. Se inicia una secuencia de actividad de base de datos en cada clúster de base de datos por

separado. Cada clúster suministra datos de auditoría a su propio flujo de Kinesis dentro de su propia Región de AWS. Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.

Supervisión de las bases de datos globales basadas en Aurora MySQL

Para ver el estado de una base de datos global basada en Aurora MySQL, consulte las tablas information_schema.aurora_global_db_status y information_schema.aurora_global_db_instance_status

> **Note**
>
> Las tablas `information_schema.aurora_global_db_status` y `information_schema.aurora_global_db_instance_status` solo están disponibles para las bases de datos globales de la versión 3.04.0 y posteriores de Aurora MySQL.

Para supervisar una base de datos global basada en Aurora MySQL

1. Conéctese al punto de conexión del clúster principal de la base de datos global mediante un cliente MySQL. Para obtener más información acerca de cómo conectarse, consulte Conexión a la base de datos global de Amazon Aurora.
2. Utilice la tabla `information_schema.aurora_global_db_status` en un comando psql para enumerar los volúmenes primario y secundario. Esta consulta devuelve los tiempos de retraso de los clústeres de bases de datos secundarios de la base de datos global, como en el siguiente ejemplo.

```
mysql> select * from information_schema.aurora_global_db_status;
```

```
AWS_REGION | HIGHEST_LSN_WRITTEN | DURABILITY_LAG_IN_MILLISECONDS |
 RPO_LAG_IN_MILLISECONDS | LAST_LAG_CALCULATION_TIMESTAMP | OLDEST_READ_VIEW_TRX_ID
-----------+---------------------+--------------------------------
+-------------------------+---------------------------------
+------------------------
us-east-1  |           183537946 |                              0   |
      0 |  1970-01-01 00:00:00.000000     |               0
us-west-2  |           183537944 |                            428 |
      0 |  2023-02-18 01:26:41.925000     |              20806982
(2 rows)
```

El resultado incluye una fila para cada clúster de base de datos de la base de datos global que contiene las siguientes columnas:

- AWS_REGION: la Región de AWS donde está este clúster de base de datos. Para ver tablas que muestres las Regiones de AWS por motor, consulte Disponibilidad por región.
- HIGHEST_LSN_WRITTEN: el número de secuencia de registro (LSN) más alto escrito actualmente en este clúster de base de datos.

  Un número de secuencia de registro (LSN) es un número secuencial único que identifica un registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción posterior.
- DURABILITY_LAG_IN_MILLISECONDS: la diferencia en los valores de marca temporal entre el `HIGHEST_LSN_WRITTEN` de un clúster de base de datos secundario y el `HIGHEST_LSN_WRITTEN` del clúster de base de datos principal. Este valor es siempre 0 en el clúster de base de datos principal de la base de datos global de Aurora.
- RPO_LAG_IN_MILLISECONDS: el retraso del objetivo de punto de recuperación (RPO). El retardo de RPO es el tiempo que tarda la transacción de usuario más reciente en almacenarse en un clúster de base de datos secundario después de almacenarse en el clúster de base de datos principal de una base de datos global de Aurora. Este valor es siempre 0 en el clúster de base de datos principal de la base de datos global de Aurora.

  En términos sencillos, esta métrica calcula el objetivo de punto de recuperación de cada clúster de base de datos de Aurora MySQL de una base de datos global de Aurora, es decir, cuántos datos podrían perderse si se produce una interrupción. Al igual que con el retraso, el RPO se mide en tiempo.
- LAST_LAG_CALCULATION_TIMESTAMP: la marca temporal que especifica cuándo se calcularon por última vez los valores para `DURABILITY_LAG_IN_MILLISECONDS` y `RPO_LAG_IN_MILLISECONDS`. Un valor temporal como `1970-01-01 00:00:00+00` significa que este es el clúster de base de datos principal.
- OLDEST_READ_VIEW_TRX_ID: el ID de la transacción más antigua a la que se puede purgar la instancia de base de datos del escritor.

3. Utilice la tabla `information_schema.aurora_global_db_instance_status` para enumerar todas las instancias de base de datos secundarias tanto para el clúster de base de datos principal como para los clústeres de base de datos secundarios.

```
mysql> select * from information_schema.aurora_global_db_instance_status;
```

```
SERVER_ID            |              SESSION_ID              | AWS_REGION
 | DURABLE_LSN | HIGHEST_LSN_RECEIVED | OLDEST_READ_VIEW_TRX_ID |
 OLDEST_READ_VIEW_LSN | VISIBILITY_LAG_IN_MSEC
---------------------+--------------------------------------+------------
+-------------+----------------------+------------------------
+----------------------+------------------------
ams-gdb-primary-i2   | MASTER_SESSION_ID                    | us-east-1  |
 183537698   |                    0 |                      0 |
 0 |                      0
ams-gdb-secondary-i1 | cc43165b-bdc6-4651-abbf-4f74f08bf931 | us-west-2  |
 183537689   |            183537692 |                 20806928 |
 183537682 |                       0
ams-gdb-secondary-i2 | 53303ff0-70b5-411f-bc86-28d7a53f8c19 | us-west-2  |
 183537689   |            183537692 |                 20806928 |
 183537682 |                     677
ams-gdb-primary-i1   | 5af1e20f-43db-421f-9f0d-2b92774c7d02 | us-east-1  |
 183537697   |            183537698 |                 20806930 |
 183537691 |                      21
(4 rows)
```

El resultado incluye una fila para cada instancia de base de datos de la base de datos global que contiene las siguientes columnas:

- SERVER_ID: el identificador del servidor de la instancia de base de datos.
- SESSION_ID: un identificador único para la sesión actual. Un valor `MASTER_SESSION_ID` identifica la instancia de base de datos de Writer (principal).
- AWS_REGION: la Región de AWS donde está esta instancia de base de datos. Para ver tablas que muestres las Regiones de AWS por motor, consulte Disponibilidad por región.
- DURABLE_LSN: el LSN hecho permanente en el almacenamiento.
- HIGHEST_LSN_RECEIVED: el LSN más alto recibido por la instancia de base de datos de la instancia de base de datos del escritor.
- OLDEST_READ_VIEW_TRX_ID: el ID de la transacción más antigua a la que puede purgar la instancia de base de datos del escritor.
- OLDEST_READ_VIEW_LSN: el LSN más antiguo utilizado por la instancia de base de datos para leer desde el almacenamiento.

- VISIBILITY_LAG_IN_MSEC: para los lectores del clúster de base de datos principal, cuánto se está retrasando esta instancia de base de datos con respecto a la instancia de base de datos del escritor en milisegundos. En el caso de los lectores de un clúster de base de datos secundario, cuánto se está retrasando esta instancia de base de datos respecto al volumen secundario en milisegundos.

Para ver cómo cambian estos valores con el tiempo, tenga en cuenta el siguiente bloque de transacciones en el que una inserción de tabla tarda una hora.

```
mysql> BEGIN;
mysql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
mysql> COMMIT;
```

En algunos casos, puede haber una desconexión de red entre el clúster de base de datos principal y el clúster de base de datos secundario después de la instrucción `BEGIN`. Si es así, el valor de DURABILITY_LAG_IN_MILLISECONDS del clúster de base de datos secundario comienza a aumentar. Al final de la instrucción `INSERT`, el valor de DURABILITY_LAG_IN_MILLISECONDS es 1 hora. Sin embargo, el valor de RPO_LAG_IN_MILLISECONDS es 0 porque todos los datos de usuario confirmados entre el clúster de base de datos principal y el clúster de base de datos secundario siguen siendo los mismos. Tan pronto como la instrucción `COMMIT` se completa, el valor de RPO_LAG_IN_MILLISECONDS aumenta.

Supervisión de bases de datos globales basadas en Aurora PostgreSQL

Utilice las funciones `aurora_global_db_status` y `aurora_global_db_instance_status` para ver el estado de una base de datos global basada en Aurora PostgreSQL.

> **Note**
>
> Solo Aurora PostgreSQL es compatible con las funciones `aurora_global_db_status` y `aurora_global_db_instance_status`.

Para supervisar una base de datos global basada en Aurora PostgreSQL

1. Conéctese al punto de enlace del clúster principal de la base de datos global mediante una utilidad PostgreSQL como psql. Para obtener más información acerca de cómo conectarse, consulte Conexión a la base de datos global de Amazon Aurora.

2. Utilice la función `aurora_global_db_status` de un comando psql para enumerar los volúmenes primario y secundario. Esto muestra los tiempos de retraso de los clústeres de bases de datos secundarios de la base de datos global.

```
postgres=> select * from aurora_global_db_status();
```

```
aws_region | highest_lsn_written | durability_lag_in_msec | rpo_lag_in_msec |
 last_lag_calculation_time  | feedback_epoch | feedback_xmin
------------+---------------------+------------------------+-----------------
+----------------------------+----------------+--------------
us-east-1  |         93763984222 |                     -1 |              -1 |
 1970-01-01 00:00:00+00     |              0 |              0
us-west-2  |         93763984222 |                    900 |            1090 |
 2020-05-12 22:49:14.328+00 |              2 |    3315479243
(2 rows)
```

El resultado incluye una fila para cada clúster de base de datos de la base de datos global que contiene las siguientes columnas:

- aws_region: la Región de AWS donde está este clúster de base de datos. Para ver tablas que muestres las Regiones de AWS por motor, consulte Disponibilidad por región.
- highest_lsn_written: el número de secuencia de registro (LSN) más alto escrito actualmente en este clúster de base de datos.

  Un número de secuencia de registro (LSN) es un número secuencial único que identifica un registro en el registro de transacciones de la base de datos. Los LSN se ordenan de tal manera que un LSN más grande representa una transacción posterior.
- durability_lag_in_msec: la diferencia de marca temporal entre el número de secuencia de registro más alto escrito en un clúster de base de datos secundario (`highest_lsn_written`) y el `highest_lsn_written` del clúster de base de datos principal.
- rpo_lag_in_msec: el retraso del objetivo del punto de recuperación (RPO). Este retraso es la diferencia de tiempo entre la confirmación de transacción de usuario más reciente almacenada en un clúster de base de datos secundario y la confirmación de transacción de usuario más reciente almacenada en el clúster de base de datos principal.
- last_lag_calculation_time: la marca temporal en la que se calcularon por última vez los valores para `durability_lag_in_msec` y `rpo_lag_in_msec`.

- feedback_epoch: el tiempo que utiliza el clúster de base de datos secundario cuando genera información de la espera activa.

  En espera en caliente es cuando un clúster de base de datos puede conectarse y realizar consultas mientras el servidor está en modo de recuperación o espera. La retroalimentación en espera en caliente es información sobre el clúster de base de datos cuando está en espera en caliente. Para obtener más información, consulte la documentación de PostgreSQL sobre Hot Standby.
- feedback_xmin: el ID de transacción activa mínima (más antigua) que utiliza el clúster de base de datos secundario.

3. Utilice la función `aurora_global_db_instance_status` para enumerar todas las instancias de base de datos secundarias tanto para el clúster de base de datos principal como para los clústeres de base de datos secundarios.

```
postgres=> select * from aurora_global_db_instance_status();
```

```
server_id                                    |              session_id
  | aws_region | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------------+--------------------------------------
+------------+-------------+------------------+----------------+---------------
+----------------------+-----------------------
apg-global-db-rpo-mammothrw-elephantro-1-n1 | MASTER_SESSION_ID
 | us-east-1  | 93763985102 |                  |                |               |
                     |
apg-global-db-rpo-mammothrw-elephantro-1-n2 | f38430cf-6576-479a-b296-dc06b1b1964a
 | us-east-1  | 93763985099 |      93763985102 |              2 |    3315479243 |
        93763985095 |                      10
apg-global-db-rpo-elephantro-mammothrw-n1   | 0d9f1d98-04ad-4aa4-8fdd-e08674cbbbfe
 | us-west-2  | 93763985095 |      93763985099 |              2 |    3315479243 |
        93763985089 |                    1017
(3 rows)
```

El resultado incluye una fila para cada instancia de base de datos de la base de datos global que contiene las siguientes columnas:

- server_id: el identificador del servidor de la instancia de base de datos.
- session_id: un identificador único para la sesión actual.

- aws_region: la Región de AWS donde está esta instancia de base de datos. Para ver tablas que muestres las Regiones de AWS por motor, consulte Disponibilidad por región.
- durable_lsn: el LSN hecho permanente en el almacenamiento.
- highest_lsn_rcvd: el LSN más alto recibido por la instancia de base de datos de la instancia de base de datos de escritor.
- feedback_epoch: la fecha de inicio que utiliza la instancia de base de datos cuando genera información en espera en caliente.

  La espera activa es cuando una instancia de base de datos puede conectarse y realizar consultas mientras el servidor está en modo de recuperación o espera. La retroalimentación en espera en caliente es información sobre la instancia de base de datos cuando está en espera en caliente. Para obtener más información, consulte la documentación de PostgreSQL sobre Hot Standby.
- feedback_xmin: el ID de transacción activo mínimo (más antiguo) utilizado por la instancia de base de datos.
- oldest_read_view_lsn: el LSN más antiguo utilizado por la instancia de base de datos para leer desde el almacenamiento.
- visibility_lag_in_msec: hasta qué punto esta instancia de base de datos se está quedando por detrás de la instancia de base de datos de escritor.

Para ver cómo cambian estos valores con el tiempo, tenga en cuenta el siguiente bloque de transacciones en el que una inserción de tabla tarda una hora.

```
psql> BEGIN;
psql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
psql> COMMIT;
```

En algunos casos, puede haber una desconexión de red entre el clúster de base de datos principal y el clúster de base de datos secundario después de la instrucción `BEGIN`. En tal caso, el valor del clúster de base de datos secundario `durability_lag_in_msec` comienza a aumentar. Al final de la instrucción `INSERT`, el valor `durability_lag_in_msec` es 1 hora. Sin embargo, el valor `rpo_lag_in_msec` es 0 porque todos los datos de usuario confirmados entre el clúster de base de datos principal y el clúster de base de datos secundario siguen siendo los mismos. En cuanto se complete la instrucción `COMMIT`, el valor `rpo_lag_in_msec` aumenta.

Uso de las implementaciones azul/verde para base de datos global de Amazon Aurora

Las implementaciones azul/verde de Amazon RDS ofrecen la capacidad de probar los cambios en las bases de datos de forma segura. Para la base de datos global, las implementaciones azul/verde le permiten realizar operaciones de actualización y mantenimiento con un tiempo de inactividad mínimo. Puede crear un entorno provisional totalmente administrado (verde) que refleje todo el entorno de producción (azul), incluido el clúster principal y todas las regiones secundarias asociadas en varias regiones de AWS. El entorno provisional refleja su configuración de producción, lo que le permite probar los cambios de forma fiable antes de cambiarse al nuevo entorno. Durante todo el proceso, las implementaciones azul/verde mantienen sincronizados los entornos provisionales y de producción. Cuando esté listo para convertir su entorno provisional en el nuevo entorno de producción, realice una transición azul/verde. Esta operación hace que las regiones principales y todas las secundarias pasen a un entorno verde, con un tiempo de inactividad normalmente inferior a un minuto. El servicio cambia automáticamente el nombre de los clústeres, las instancias y los puntos de conexión para que coincidan con el entorno de producción original, lo que permite a las aplicaciones acceder al nuevo entorno de producción sin necesidad de cambios de configuración y minimiza la sobrecarga operativa.

Temas

- Beneficios de usar implementaciones azul/verde con base de datos global de Aurora
- Como funcionan las implementaciones azul/verde con la base de datos global de Aurora

Beneficios de usar implementaciones azul/verde con base de datos global de Aurora

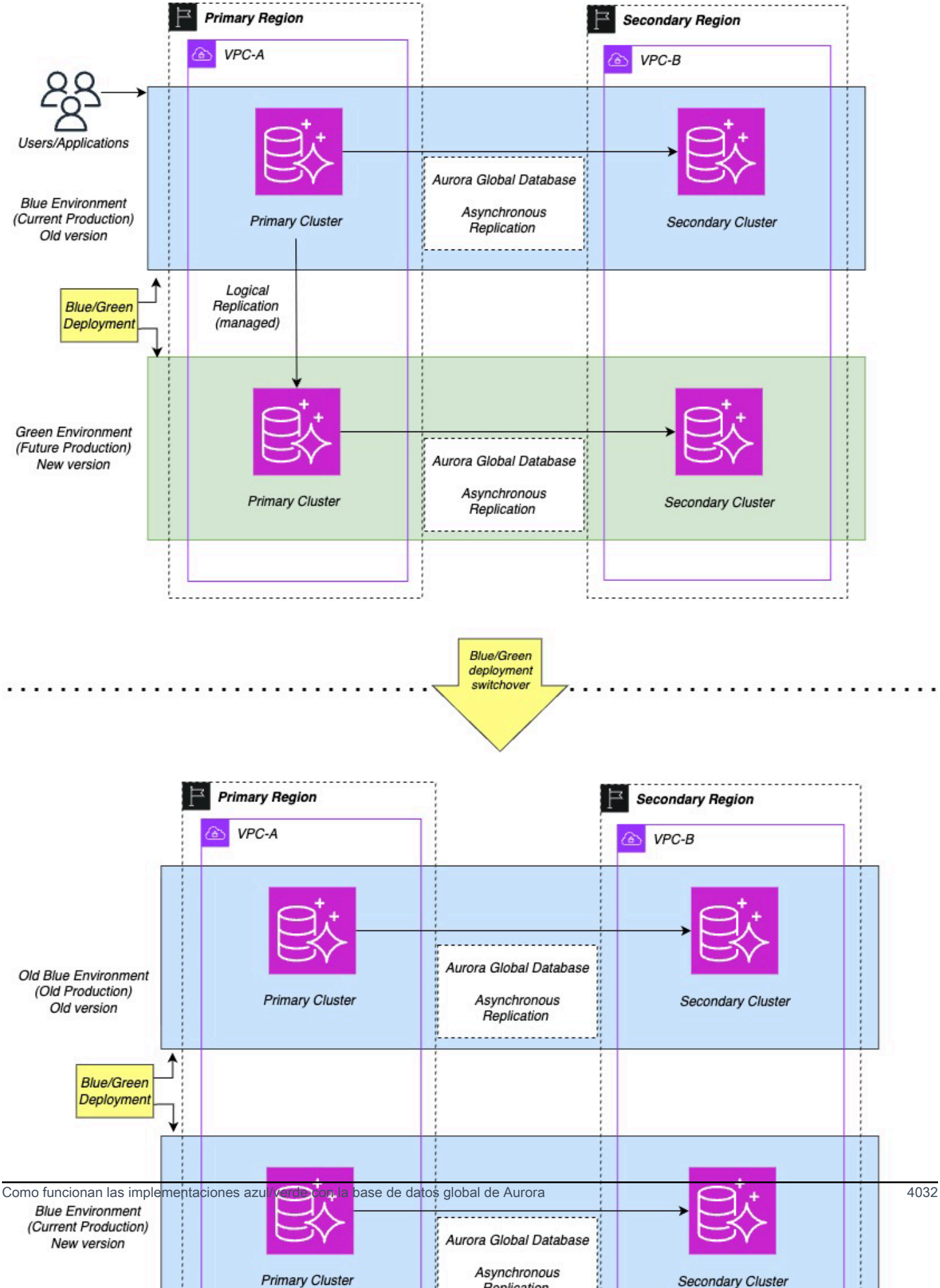
- Realice actualizaciones de la versión principal, la versión secundaria y el sistema, incluidos parches de bases de datos y actualizaciones del sistema operativo en las bases de datos globales de Aurora, a la vez que mantiene un tiempo de inactividad mínimo.
- Cree un entorno provisional (verde) listo para la producción en las regiones principal y secundaria de la base de datos global para probar e implementar las nuevas características de la base de datos.
- Realice la transición de forma segura mediante las barreras de protección de transición integradas, con un tiempo de inactividad normalmente inferior a un minuto, en función de la carga de trabajo.

- Mantiene las capacidades de recuperación ante desastres durante el proceso de transición azul/verde, lo que permite la conmutación por error de base de datos global durante la transición azul/verde.
- El tráfico se dirigirá al nuevo entorno de producción sin necesidad de realizar cambios en las aplicaciones.

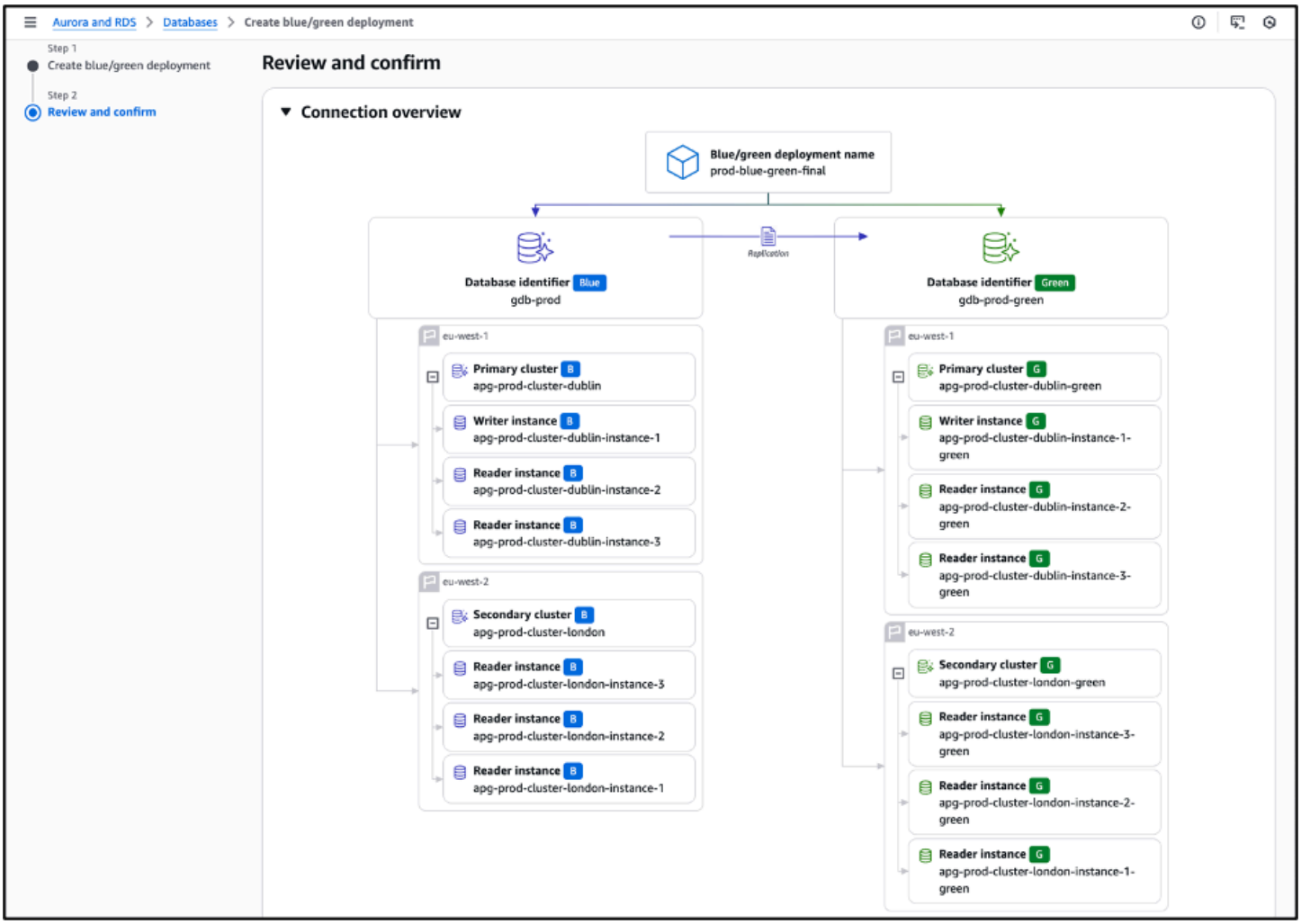
Como funcionan las implementaciones azul/verde con la base de datos global de Aurora

Para obtener más información sobre cómo crear, ver, cambiar y eliminar una implementación azul/verde, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos. Puede utilizarla para realizar actualizaciones de versiones principales o secundarias, mejorar el rendimiento de la base de datos mediante actualizaciones de parámetros y adoptar nuevas características de base de datos, con un tiempo de inactividad mínimo.

A continuación, se muestra una representación de cómo se ve una implementación azul/verde de la base de datos global de Aurora con una región secundaria antes y después de una transición azul/verde.


Primary Region
VPC-A
Secondary Region
VPC-B
Users/Applications
Blue Environment
(Current Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Logical
Replication
(managed)
Green Environment
(Future Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
deployment
switchover
Primary Region
VPC-A
Secondary Region
VPC-B
Old Blue Environment
(Old Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Blue Environment
(Current Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster

Puede crear una implementación azul/verde a partir de la región principal de la base de datos global. Seleccione las configuraciones del motor, como la versión principal o secundaria del motor, el grupo de parámetros de base de datos y el grupo de parámetros del clúster de base de datos para el entorno verde. Amazon RDS copia la topología del entorno azul para el entorno verde. A continuación, se muestra una representación visual en la Consola de administración de AWS.



> **Note**
>
> La conmutación por error global se admite durante una transición azul/verde, pero la transición global no se admite durante una transición azul/verde.

Cuando se inicia una conmutación por error global durante una transición azul/verde de RDS, la región de destino vuelve automáticamente al entorno azul o avanza al entorno verde antes de que se produzca la conmutación por error global.

Para obtener información sobre cómo crear, ver, cambiar y eliminar implementaciones azul/verde, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos. Siga el mismo flujo de trabajo para las bases de datos globales, con instrucciones específicas en los pasos correspondientes.

Uso de las bases de datos globales de Amazon Aurora con otros servicios de AWS

Puede utilizar sus bases de datos de Aurora globales con otros servicios de AWS, como Amazon S3 y AWS Lambda. Este proceso requiere que todos los clústeres de base de datos de Aurora en su base de datos global tengan los mismos privilegios, funciones externas, etc. en las respectivas Regiones de AWS. Dado que un clúster de base de datos Aurora secundario de solo lectura en una base de datos Aurora global se puede promover a la función de primario, se recomienda configurar privilegios de escritura con anticipación, en todos los clústeres de base de datos Aurora para cualquier servicio que planee utilizar con la base de datos Aurora global.

Los siguientes procedimientos resumen las acciones que se deben realizar para cada Servicio de AWS.

Para invocar características de AWS Lambda desde una base de datos de Aurora global

1. Para los clústeres de Aurora que componen la base de datos global de Aurora, realice los procedimientos de Invocación de una función de Lambda desde un clúster de base de datos de Amazon Aurora MySQL.
2. Para cada clúster de la base de datos Aurora global, establezca el (ARN) de la nueva función IAM (IAM).
3. Para permitir que los usuarios de una base de datos global de Aurora invoquen funciones Lambda, asocie el rol que ha creado en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con cada clúster en la base de datos global de Aurora.
4. Configure cada clúster de la base de datos global de Aurora para permitir conexiones salientes hacia Lambda. Para obtener instrucciones, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

Cargar datos desde Amazon S3.

1. Para los clústeres de Aurora que componen la base de datos global de Aurora, realice los procedimientos de Carga de datos en un clúster de base de datos Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3.
2. Para cada clúster de la base de datos global de Aurora, configure o bien el parámetro del clúster de la base de datos `aurora_load_from_s3_role` o el `aws_default_s3_role` con el Nombre de recurso de Amazon (ARN) del nuevo rol de IAM. Si no se ha especificado un rol de IAM para `aurora_load_from_s3_role`, Aurora utilizará el rol de IAM especificado en el parámetro `aws_default_s3_role`.
3. Para permitir que los usuarios de una base de datos global de Aurora accedan a S3, asocie el rol que ha creado en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con cada clúster de Aurora en la base de datos global.
4. Configure cada clúster de la base de datos global de Aurora para permitir conexiones salientes hacia S3. Para obtener instrucciones, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

Para guardar los datos consultados en Amazon S3

1. Para los clústeres de Aurora que componen la base de datos global de Aurora, lleve a cabo los procedimientos en Grabación de datos desde un clúster de base de datos Amazon Aurora MySQL en archivos de texto de un bucket de Amazon S3 o Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3.
2. Para cada clúster de la base de datos global de Aurora, configure o bien el parámetro del clúster de la base de datos `aurora_select_into_s3_role` o el `aws_default_s3_role` con el Nombre de recurso de Amazon (ARN) del nuevo rol de IAM. Si no se ha especificado un rol de IAM para `aurora_select_into_s3_role`, Aurora utilizará el rol de IAM especificado en el parámetro `aws_default_s3_role`.
3. Para permitir que los usuarios de una base de datos global de Aurora accedan a S3, asocie el rol que ha creado en Creación de un rol de IAM que permita a Amazon Aurora acceder a los servicios de AWS con cada clúster de Aurora en la base de datos global.
4. Configure cada clúster de la base de datos global de Aurora para permitir conexiones salientes hacia S3. Para obtener instrucciones, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.

Actualización de una base de datos global de Amazon Aurora

La actualización de una base de datos global de Aurora sigue los mismos procedimientos que la actualización de clústeres de base de datos de Aurora. Sin embargo, estas son algunas diferencias importantes a tener en cuenta antes de comenzar el proceso.

Se recomienda actualizar los clústeres de base de datos principal y secundaria a la misma versión. Solo puede realizar una conmutación por error administrada de la base de datos entre regiones en una base de datos global de Aurora si los clústeres de base de datos principal y secundario tienen las mismas versiones principal, secundaria y de nivel de revisión del motor. Sin embargo, los niveles de revisión pueden ser diferentes en función de la versión secundaria del motor. Para obtener más información, consulte Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones.

Actualizaciones de la versión principal

Cuando realice una actualización de versión principal de una base de datos global de Amazon Aurora, actualiza el clúster de la base de datos global en lugar de los clústeres individuales que contiene.

Para obtener información sobre cómo actualizar una base de datos global de Aurora PostgreSQL a una versión principal superior, consulte Actualizaciones importantes para bases de datos globales.

> **Note**
>
> Con una base de datos global de Aurora basada en Aurora PostgreSQL, no se puede realizar una actualización de versión importante del motor de base de datos de Aurora si la característica Objetivo de punto de recuperación (RPO) está habilitada. Para obtener información sobre la característica RPO, consulte Administración de RPO para bases de datos globales basadas en Aurora PostgreSQL–.

Para obtener información sobre cómo actualizar una base de datos global de Aurora PostgreSQL a una versión principal superior, consulte Actualizaciones mayores en el lugar para bases de datos globales.

> **Note**
>
> Con una base de datos global de Aurora basada en Aurora MySQL, puede realizar una actualización local desde la versión 2 a la versión 3 de Aurora MySQL si establece el parámetro `lower_case_table_names` en predeterminado y reinicia la base de datos global.
> Para realizar una actualización de la versión principal a Aurora MySQL versión 3 con `lower_case_table_names`, utilice el siguiente proceso:
>
> 1. Elimine todas las regiones secundarias del clúster global. Siga los pasos de Eliminación de un clúster de una base de datos global de Amazon Aurora.
> 2. Actualice la versión del motor de la región principal a Aurora MySQL versión 3. Siga los pasos de Pasos para realizar una actualización local.
> 3. Añada regiones secundarias al clúster global. Siga los pasos de Incorporación de una Región de AWS a una base de datos global de Amazon Aurora.
>
> También puede utilizar el método de restauración de instantáneas en su lugar. Para obtener más información, consulte Restauración de una instantánea de clúster de base de datos.

Actualizaciones de la versión secundaria

Para una actualización menor en una base de datos global de Aurora, actualice todos los clústeres secundarios antes de actualizar el clúster principal.

Para obtener información sobre cómo actualizar una base de datos global de Aurora PostgreSQL a una versión secundaria superior, consulte Cómo realizar actualizaciones de versión secundarias y aplicar revisiones. Para obtener información sobre cómo actualizar una base de datos global de Aurora MySQL a una versión secundaria superior, consulte Actualización de Aurora MySQL mediante la modificación de la versión del motor.

Antes de realizar la actualización, tenga en cuenta lo siguiente:

- La actualización de la versión secundaria de un clúster secundario no afecta en modo alguno a la disponibilidad ni al uso del clúster principal.
- Un clúster secundario debe tener al menos una instancia de base de datos para realizar una actualización menor.

- Si actualiza una base de datos global de Aurora MySQL a la versión 2.11.*, debe actualizar los clústeres de base de datos principales y secundarios a exactamente la misma versión, incluido el nivel de parche.
- Si actualiza una base de datos global de Aurora PostgreSQL, debe actualizar los clústeres de base de datos principales y secundarios a esa misma versión exactamente y el nivel de parche. Para actualizar el nivel de parche, aplique todas las acciones de mantenimiento pendientes en el clúster secundario.
- Para poder realizar la transición o la conmutación por error administrada entre regiones, puede que tenga que actualizar sus clústeres de bases de datos principal y secundario exactamente a la misma versión, incluido el nivel de parche. Este requisito se aplica a Aurora MySQL y a algunas versiones de Aurora PostgreSQL. Para obtener una lista de las versiones que permiten transiciones y conmutaciones por error entre clústeres que ejecutan diferentes niveles de parche, consulte Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones.

Compatibilidad de los niveles de parche para la transición o conmutación por error administrada entre regiones

Si su base de datos global de Aurora se ejecuta en una de las siguientes versiones secundarias del motor, podrá realizar una transición o conmutación por error administrada entre regiones aunque los niveles de parche de los clústeres de base de datos principal y secundario no coincidan. Para las versiones secundarias del motor anteriores a las de esta lista, los clústeres de base de datos principal y secundario se deben ejecutar en los mismos niveles de versión principal, secundaria y de parche para realizar una transición o conmutación por error administrada entre regiones. Asegúrese de revisar la información de la versión y las notas de la siguiente tabla cuando planifique las actualizaciones de su clúster principal, de los clústeres secundarios o de ambos.

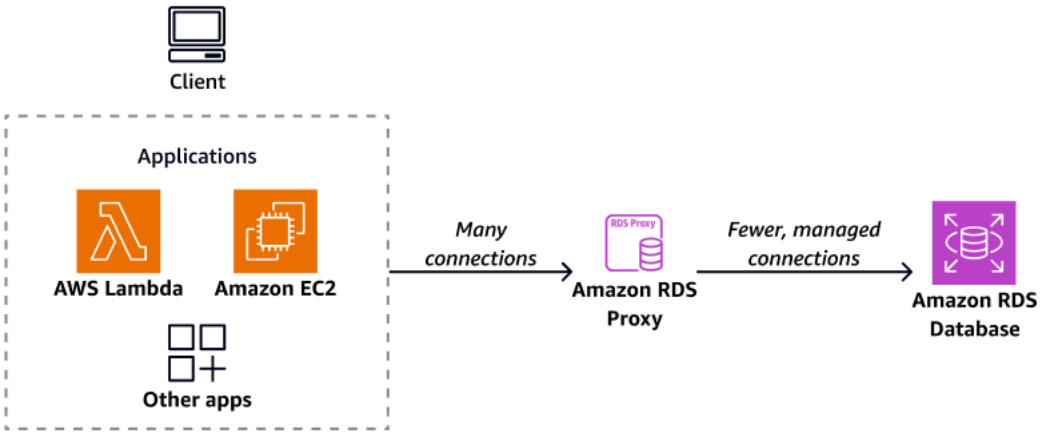
> **Note**
>
> Para las conmutaciones por error entre regiones manuales, puede realizar el proceso de conmutación por error siempre que el clúster de base de datos secundario de destino ejecute la misma versión principal y secundaria del motor que el clúster de base de datos principal. En este caso, no es necesario que los niveles de revisión coincidan.
> Si las versiones del motor requieren niveles de parches idénticos, puede realizar la conmutación por error manualmente por medio de los pasos que se indican en Ejecución de la conmutación por error manual para bases de datos globales de Aurora.

Motor de base de datos	Versiones secundarias del motor	Notas
Aurora MySQL	No hay versiones secundarias	Ninguna de las versiones de Aurora MySQL permiten una transición o conmutación por error administrada entre regiones si los niveles de parche de los clústeres de base de datos principal y secundario son distintos.
Aurora PostgreSQL	• Versión 15 o versiones principales superiores • Versión 14.5 o versiones secundarias superiores • Versión 13.8 o versiones secundarias superiores • Versión 12.12 o versiones secundarias superiores • Versión 11.17 o versiones secundarias superiores	En las versiones del motor enumeradas en la columna anterior, puede realizar una transición o conmutación por error administrada entre regiones desde un clúster de base de datos principal con un nivel de parche a un clúster de base de datos secundario que tenga un nivel de parche diferente. En las versiones secundarias anteriores a esas, solo puede realizar una transición o conmutación por error administrada entre regiones si los niveles de parche de los clústeres de base de datos principal y secundario coinciden. **Warning** Cuando actualice un clúster de la base de datos global a cualquiera de las siguientes versiones de revisión, no podrá realizar transiciones entre regiones ni conmutaci

Motor de base de datos	Versiones secundarias del motor	Notas
		ones por error hasta que todos los clústeres de la base de datos global ejecuten una de estas versiones de revisión o una más reciente. • Versiones de revisión 16.1.6, 16.2.4, 16.3.2 y 16.4.2 • Versiones de revisión 15.3.8, 15.4.9, 15.5.6, 15.6.4, y 15.8.2 • Versiones de revisión 14.8.8, 14.9.9, 14.10.6, 14.11 y 14.13.2

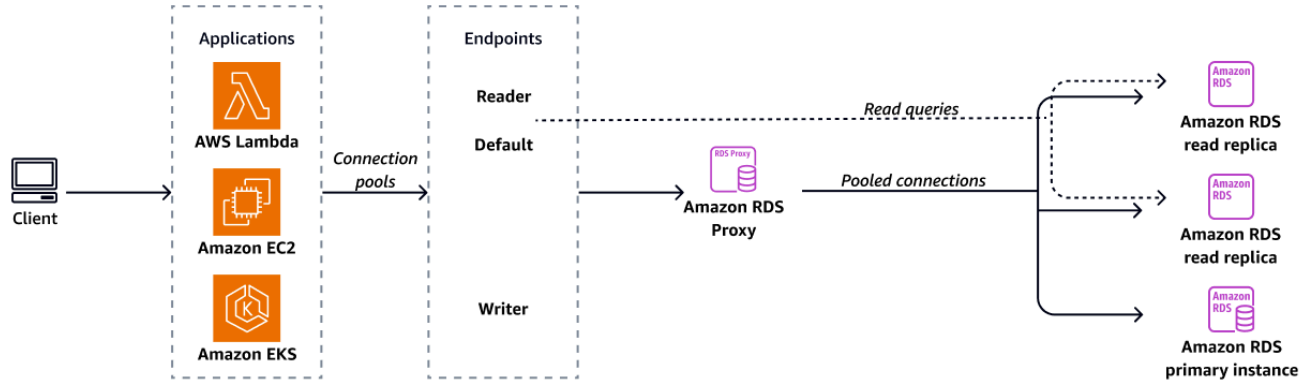
Amazon RDS Proxy para Aurora

Con Amazon RDS Proxy puede permitir a las aplicaciones agrupar y compartir conexiones de base de datos para mejorar su capacidad de escala. RDS Proxy hace que las aplicaciones sean más resistentes a los errores de base de datos al conectarse automáticamente a una instancia de base de datos en espera mientras se preservan las conexiones de las aplicaciones. Con RDS Proxy, puede aplicar la autenticación de AWS Identity and Access Management (IAM) a los clientes que se conectan al proxy, y el proxy puede conectarse a las bases de datos mediante la autenticación de bases de datos de IAM o las credenciales almacenadas en AWS Secrets Manager.



Con RDS Proxy puede gestionar aumentos imprevistos en el tráfico de base de datos. De lo contrario, estas sobrecargas podrían causar problemas debido a la suscripción excesiva de conexiones o a la creación de nuevas conexiones a un ritmo rápido. RDS Proxy establece un grupo de conexiones de base de datos y reutiliza las conexiones de este grupo. Este enfoque evita la sobrecarga de memoria y de CPU que supone abrir una nueva conexión de base de datos cada vez. Para proteger una base de datos frente a un exceso de suscripciones, puede controlar el número de conexiones de base de datos que se crean.

RDS Proxy pone en cola o limita las conexiones de aplicaciones que no se pueden atender de inmediato desde el grupo de conexiones. Aunque las latencias pueden aumentar, la aplicación puede seguir ajustando la escala sin fallar bruscamente ni sobrecargar la base de datos. Si las solicitudes de conexión superan los límites especificados, RDS Proxy rechaza las conexiones de aplicación (es decir, se desprende de la carga). Al mismo tiempo, mantiene un rendimiento predecible para la carga que RDS puede servir con la capacidad disponible.



Puede reducir la sobrecarga para procesar credenciales y establecer una conexión segura para cada nueva conexión. RDS Proxy puede gestionar parte de ese trabajo en nombre de la base de datos.

RDS Proxy es totalmente compatible con las versiones de motor admitidas. Puede habilitar RDS Proxy para la mayoría de las aplicaciones sin cambios de código. Para ver una lista de las versiones de motor admitidas, consulte Regiones y motores de base de datos Aurora para Amazon RDS Proxy.

Temas

- Disponibilidad en regiones y versiones
- Cuotas y limitaciones de RDS Proxy
- Planificación del lugar de uso de RDS Proxy
- Conceptos y terminología de RDS Proxy
- Introducción al proxy de RDS
- Administración de un RDS Proxy
- Trabajo con puntos de enlace del proxy de Amazon RDS
- Supervisión de las métricas de RDS Proxy con Amazon CloudWatch
- Trabajo con eventos de RDS Proxy
- Ejemplos de línea de comandos del proxy de RDS
- Solución de problemas de RDS Proxy
- Uso del proxy de RDS con AWS CloudFormation
- Uso de RDS Proxy con bases de datos globales de Aurora

Disponibilidad en regiones y versiones

Para obtener información sobre la compatibilidad de versiones del motor de base de datos y la disponibilidad de RDS Proxy en una determinada Región de AWS, consulte Regiones y motores de base de datos Aurora para Amazon RDS Proxy.

Cuotas y limitaciones de RDS Proxy

Las siguientes cuotas y limitaciones se aplican a RDS Proxy:

- Cada ID de Cuenta de AWS está limitado a 20 proxies. Si su aplicación requiere más proxies, solicite un aumento a través de la página Service Quotas dentro de la Consola de administración de AWS. En la página Service Quotas, seleccione Amazon Relational Database Service (Amazon RDS) y busque Proxies para solicitar un aumento de cuota. AWS puede aumentar automáticamente su cuota o, en espera de la revisión de su solicitud, mediante Soporte.
- Cada proxy puede tener hasta 200 secretos de Secrets Manager asociados, lo que limita las conexiones hasta a 200 cuentas de usuario diferentes cuando se utilizan secretos.
- Cada proxy tiene un punto de conexión predeterminado. También puede agregar hasta 20 puntos de conexión de proxy para cada proxy. Puede crear, ver, modificar y eliminar estos puntos de conexión.
- En un clúster de Aurora, todas las conexiones que utilizan el punto de conexión de proxy predeterminado se gestionan por la instancia de escritor Aurora. A fin de llevar a cabo el balanceo de carga para cargas de trabajo de lectura intensiva, puede crear un punto de enlace de solo lectura para un proxy. Ese punto de enlace pasa las conexiones al punto de enlace del lector del clúster. De esta manera, sus conexiones de proxy pueden aprovechar la escalabilidad de lectura Aurora. Para obtener más información, consulte Información general de los puntos de enlace de proxy.
- Puede utilizar el RDS Proxy con clústeres de Aurora Serverless v2 pero no con clústeres de Aurora Serverless v1.
- Su RDS Proxy debe estar en la misma nube privada virtual (VPC) que la base de datos. Aunque se puede acceder públicamente a la base de datos, no sucede lo mismo con el proxy. Por ejemplo, si va a crear prototipos de base de datos en un host local, no puede conectarse a su proxy, a menos que configure los requisitos de red necesarios para permitir la conexión al proxy. Esto es porque el host local está fuera de la VPC del proxy.

> **Note**
>
> Para los clústeres de bases de datos de Aurora, puede activar el acceso entre VPC. Para ello, cree un punto de conexión adicional para un proxy y especifique una VPC, subredes y grupos de seguridad diferentes con ese punto de conexión. Para obtener más información, consulte Acceso a las bases de datos de Aurora en todas las VPC.

- No se puede utilizar RDS Proxy con una VPC que tenga su tenencia establecida en `dedicated`.
- Para tipos de redes de puntos de conexión de IPv6, configure la VPC y las subredes para admitir solo IPv6. Para los tipos de red de conexión de destino de IPv4 e IPv6, configure la VPC y las subredes para que admitan el modo de doble pila.
- Si utiliza RDS Proxy con un clúster de base de datos de Aurora que tenga habilitada la autenticación de IAM, el proxy puede conectarse a la base de datos mediante autenticación de IAM o credenciales almacenadas en Secrets Manager. Los clientes que se conecten al proxy deben autenticarse con credenciales de IAM. Para obtener instrucciones de configuración detalladas, consulte Configuración de credenciales de base de datos para RDS Proxy y Configuración de la autenticación de IAM para RDS Proxy
- No se puede usar el proxy de RDS con DNS personalizados cuando se utiliza la validación de nombres de host SSL.
- Cada proxy se puede asociar con un único clúster de base de datos de destino. Sin embargo, puede asociar varios proxies con el mismo clúster de base de datos.
- Cualquier instrucción con un tamaño de texto superior a 16 KB hace que el proxy fije la sesión a la conexión actual.
- Algunas regiones tienen restricciones de zona de disponibilidad (AZ) que debe tener en cuenta al crear el proxy. La región Este de EE. UU. (Norte de Virginia) no admite RDS Proxy en la zona de disponibilidad `use1-az3`. La región Oeste de EE. UU. (Norte de California) no admite RDS Proxy en la zona de disponibilidad `usw1-az2`. Al seleccionar subredes al crear el proxy, asegúrese de no seleccionar subredes en las zonas de disponibilidad mencionadas anteriormente.
- Actualmente, RDS Proxy no admite claves de contexto de condición globales.

  Para obtener más información sobre las claves de condición globales, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.
- No se puede utilizar RDS Proxy con RDS Custom para SQL Server.

- Para reflejar cualquier modificación del grupo de parámetros de la base de datos en el proxy, es necesario reiniciar la instancia aunque decida aplicar los cambios inmediatamente. Es necesario reiniciar todo el clúster para los parámetros de clúster.
- Su proxy crea automáticamente el usuario de base de datos `rdsproxyadmin` cuando registra un destino de proxy. Este es un usuario protegido que es esencial para la funcionalidad del proxy. Debe evitar cualquier alteración del usuario de `rdsproxyadmin` de cualquier forma. Eliminar o modificar el usuario de `rdsproxyadmin` o sus permisos puede provocar la total falta de disponibilidad del proxy en la aplicación.

Para conocer las limitaciones adicionales de cada motor de base de datos, consulte las secciones siguientes:

- Limitaciones adicionales para Aurora MySQL
- Limitaciones adicionales para Aurora PostgreSQL

Limitaciones adicionales para Aurora MySQL

Las siguientes limitaciones adicionales se aplican a RDS Proxy con bases de datos de Aurora MySQL:

- La compatibilidad con RDS Proxy para la autenticación de `caching_sha2_password` requiere una conexión segura (TLS).
- Se sabe que `caching_sha2_password` presenta problemas de compatibilidad con RDS Proxy si se usan determinadas versiones del controlador go-sql.
- Cuando se utiliza el controlador C de MySQL 8.4, la API `mysql_stmt_bind_named_param` puede formar paquetes con un formato incorrecto si el recuento de parámetros supera el recuento de marcadores de posición en una instrucción preparada. Esto da como resultado respuestas incorrectas. Para obtener más información, consulte MySQL bug report.
- Actualmente, todos los proxies escuchan en el puerto 3306 para MySQL. Los proxies todavía se conectan a la base de datos mediante el puerto especificado en la configuración de la base de datos.
- No puede usar RDS Proxy con bases de datos MySQL autoadministradas en instancias EC2.
- No se puede usar RDS Proxy con una instancia de base de datos RDS para MySQL que tenga el parámetro `read_only` en su grupo de parámetros de base de datos establecido en 1.

- RDS Proxy no admite el modo comprimido de MySQL. Por ejemplo, no admite la compresión utilizada por las opciones `--compress` o `-C` del comando `mysql`.
- Las conexiones a bases de datos que procesan un comando `GET DIAGNOSTIC` pueden devolver información inexacta cuando RDS Proxy vuelve a utilizar la misma conexión de base de datos para ejecutar otra consulta. Esto puede ocurrir cuando RDS Proxy multiplexa las conexiones de bases de datos.
- Algunas instrucciones y funciones SQL, como `SET LOCAL`, pueden cambiar el estado de conexión sin producir una fijación. Para conocer el comportamiento de fijación más actual, consulte Cómo evitar la fijación de RDS Proxy.
- No se admite el uso de la función `ROW_COUNT()` en una consulta de varias instrucciones.
- RDS Proxy no admite aplicaciones cliente que no puedan gestionar varios mensajes de respuesta en un registro TLS.
- RDS Proxy no admite las contraseñas duales de MySQL.
- Es posible que el proxy de RDS no funcione según lo esperado al configurar el parámetro `init_connect` en el grupo de parámetros de la base de datos de RDS para establecer las variables de estado de la sesión. En su lugar, establezca la consulta de inicialización para que el proxy ejecute las instrucciones de inicialización de la sesión cuando utilice el proxy para conectarse a la base de datos.

> **Important**
>
> Para proxies asociados con bases de datos de MySQL, no establezca el parámetro de configuración `sql_auto_is_null` en `true` o un valor distinto de cero en la consulta de inicialización. Si lo hace, es posible que la aplicación se comporte incorrectamente.

Limitaciones adicionales para Aurora PostgreSQL

Las siguientes limitaciones adicionales se aplican a RDS Proxy con bases de datos de Aurora PostgreSQL:

- RDS Proxy no admite los filtros de fijación de sesión para PostgreSQL.
- Actualmente, todos los proxies escuchan en el puerto 5432 para PostgreSQL.

- Para PostgreSQL, RDS Proxy no admite actualmente la cancelación de una consulta por parte de un cliente mediante la emisión de `CancelRequest`. Este es el caso, por ejemplo, cuando se cancela una consulta de larga duración en una sesión psql interactiva con Ctrl + C.
- Los resultados de la función de PostgreSQL lastval no siempre son precisos. Como alternativa, utilice la instrucción INSERT con la cláusula `RETURNING`.
- Actualmente, RDS Proxy no admite el modo de replicación en streaming.
- La base de datos de `postgres` predeterminada debe existir en la instancia de RDS para PostgreSQL para que funcione RDS Proxy. No elimine esta base de datos aunque la aplicación utilice bases de datos diferentes.
- Si usa `ALTER ROLE` para cambiar el rol de usuario con `SET ROLE`, es posible que las conexiones posteriores con ese usuario al proxy no usen esta configuración de rol si esas conexiones se encuentran con alguna fijación. Para evitarlo, cuando utilice un proxy, emplee `SET ROLE` en la consulta de inicialización del proxy. Para obtener más información, consulte Consulta de inicialización en Creación de un proxy para Amazon Aurora.

> **Important**
>
> En el caso de los proxies existentes con bases de datos de PostgreSQL, si modifica la autenticación de la base de datos para utilizar únicamente `SCRAM`, el proxy dejará de estar disponible durante un máximo de 60 segundos. Para evitar este problema, lleve a cabo alguna de las siguientes operaciones:
>
> - Asegúrese de que la base de datos permita la autenticación `SCRAM` y `MD5`.
> - Para utilizar únicamente la autenticación `SCRAM`, cree un nuevo proxy, migre el tráfico de la aplicación al nuevo proxy y, a continuación, elimine el proxy previamente asociado a la base de datos.

Planificación del lugar de uso de RDS Proxy

Puede determinar cuáles de sus instancias de base de datos, clústeres y aplicaciones podrían beneficiarse más del uso de RDS Proxy. Para ello, tenga en cuenta estos factores:

- Cualquier clúster de base de datos que encuentre errores de "demasiadas conexiones" es un buen candidato para asociarse con un proxy. Esto suele caracterizarse por un valor alto de la métrica `ConnectionAttempts` de CloudWatch. El proxy permite a las aplicaciones abrir muchas

conexiones de cliente, mientras que el proxy administra un número menor de conexiones de larga duración a el clúster de base de datos.

- Para clústeres de base de datos que utilizan clases de instancias de AWS más pequeñas, como T2 o T3, el uso de un proxy puede ayudar a evitar condiciones de falta de memoria. También puede ayudar a reducir la sobrecarga de CPU para establecer conexiones. Estas condiciones pueden producirse cuando se trata de un gran número de conexiones.
- Puede monitorear ciertas métricas de Amazon CloudWatch para determinar si un clúster de base de datos se acerca a ciertos tipos de límite. Estos límites son para el número de conexiones y la memoria asociados a la administración de conexiones. También puede monitorear ciertas métricas de CloudWatch para determinar si un clúster de base de datos está controlando muchas conexiones de corta duración. Abrir y cerrar tales conexiones puede imponer una sobrecarga de rendimiento en su base de datos. Para obtener información sobre las métricas que se van a monitorizar, consulte Supervisión de las métricas de RDS Proxy con Amazon CloudWatch.
- AWS LambdaLas funciones de también pueden ser buenas candidatas para usar un proxy. Estas funciones hacen frecuentes conexiones cortas a la base de datos que aprovechan el grupo de conexiones que ofrece RDS Proxy. Puede aprovechar cualquier autenticación de IAM que ya tenga para funciones de Lambda, en lugar de administrar las credenciales de la base de datos en el código de la aplicación de Lambda.
- Las aplicaciones que suelen abrir y cerrar un gran número de conexiones de base de datos y no tienen mecanismos integrados de agrupación de conexiones son buenas candidatas para usar un proxy.
- Las aplicaciones que mantienen un gran número de conexiones abiertas durante largos períodos suelen ser buenas candidatas para usar un proxy. Las aplicaciones en sectores como el software como servicio (SaaS) o el comercio electrónico a menudo minimizan la latencia de las solicitudes de base de datos al dejar las conexiones abiertas. Con RDS Proxy, una aplicación puede mantener más conexiones abiertas que cuando se conecta directamente al clúster de base de datos.
- Es posible que no haya adoptado la autenticación de IAM y Secrets Manager debido a la complejidad de configurar dicha autenticación para todos los clústeres de base de datos. El proxy puede aplicar las directivas de autenticación para conexiones de cliente para aplicaciones concretas. Puede aprovechar cualquier autenticación de IAM que ya tenga para funciones de Lambda, en lugar de administrar las credenciales de la base de datos en el código de la aplicación de Lambda.
- RDS Proxy puede ayudar a que las aplicaciones sean más resilientes y transparentes ante los errores de base de datos. RDS Proxy evita las cachés del Sistema de nombres de dominio (DNS)

a fin de reducir los tiempos de conmutación por error hasta en un 66 % para las bases de datos de Aurora Multi-AZ. RDS Proxy también dirige automáticamente el tráfico a una nueva instancia de base de datos y conserva las conexiones de la aplicación. Esto hace que la conmutación por error sea más transparente para las aplicaciones.

Conceptos y terminología de RDS Proxy

Puede simplificar la administración de conexiones de los clústeres de bases de datos de Amazon Aurora usando RDS Proxy.

RDS Proxy controla el tráfico de red entre la aplicación cliente y la base de datos. Lo hace de una manera activa, comprendiendo primero el protocolo de base de datos. A continuación, ajusta su comportamiento en función de las operaciones SQL de la aplicación y los conjuntos de resultados de la base de datos.

RDS Proxy reduce la sobrecarga de memoria y CPU para la administración de conexiones en la base de datos. La base de datos necesita menos memoria y recursos de CPU cuando las aplicaciones abren muchas conexiones simultáneas. Tampoco requiere lógica en las aplicaciones para cerrar y volver a abrir conexiones que permanecen inactivas durante mucho tiempo. Del mismo modo, requiere menos lógica de aplicación para restablecer conexiones en caso de un problema de base de datos.

La infraestructura para RDS Proxy está altamente disponible e implementada en varias zonas de disponibilidad (AZ). El cálculo, la memoria y el almacenamiento de RDS Proxy son independientes de los clústeres de bases de datos de Aurora. Esta separación ayuda a reducir la sobrecarga en los servidores de bases de datos, de modo que puedan dedicar sus recursos a servir cargas de trabajo de base de datos. Los recursos informáticos de RDS Proxy no tienen servidor y se escalan automáticamente en función de la carga de trabajo de la base de datos.

Temas

- Información general de los conceptos de RDS Proxy
- Grupo de conexiones
- Seguridad de RDS Proxy
- Conmutación por error
- Transacciones

Información general de los conceptos de RDS Proxy

RDS Proxy gestiona la infraestructura para llevar a cabo la agrupación de conexiones y las demás características descritas en las siguientes secciones. Puede ver los proxies representados en la consola de RDS en la página Proxies.

Cada proxy gestiona las conexiones a un único clúster de base de datos de Aurora. El proxy determina automáticamente la instancia de escritor actual para los clústeres aprovisionados de Aurora.

Las conexiones que un proxy mantiene abiertas y disponibles para que las aplicaciones de base de datos puedan utilizar el grupo de conexiones.

De forma predeterminada, RDS Proxy puede reutilizar una conexión después de cada transacción en la sesión. Esta reutilización en el nivel de transacción se denomina multiplexación. Cuando RDS Proxy elimina temporalmente una conexión del grupo de conexiones para reutilizarla, esa operación se denomina préstamo de la conexión. Cuando sea seguro hacerlo, RDS Proxy devuelve esa conexión al grupo de conexiones.

En algunos casos, RDS Proxy no puede estar seguro de que sea seguro volver a utilizar una conexión de base de datos fuera de la sesión actual. En estos casos, mantiene la sesión en la misma conexión hasta que finalice la sesión. Este comportamiento de reserva se denomina fijación.

Un proxy tiene un punto de conexión predeterminado. Se conecta a este punto de conexión cuando trabaja con un clúster de bases de datos de Amazon Aurora. Lo hace en lugar de conectarse al punto de conexión de lectura y escritura que se conecta directamente al clúster. Los puntos de conexión para uso especial de un clúster de Aurora permanecen disponibles para su uso. Para los clústeres de bases de datos de Aurora, también puede crear puntos de conexión de lectura o escritura y de solo lectura adicionales. Para obtener más información, consulte Información general de los puntos de enlace de proxy.

Por ejemplo, aún puede conectarse al punto de enlace del clúster para conexiones de lectura y escritura sin agrupación de conexiones. Aún puede conectarse al punto de conexión del lector para conexiones de solo lectura con equilibrio de carga. Aún puede conectarse a los puntos de conexión de instancia para el diagnóstico y la resolución de problemas de instancias de base de datos específicas dentro de un clúster. Si utiliza otros servicios de AWS como AWS Lambda para conectarse a bases de datos de RDS, cambie la configuración de conexión para utilizar el punto de conexión del proxy. Por ejemplo, especifique el punto de conexión del proxy para permitir que las

funciones de Lambda accedan a la base de datos mientras aprovechan la funcionalidad del RDS Proxy.

Cada proxy contiene un grupo de destino. Este grupo de destino abarca el clúster de base de datos de Aurora que se puede conectar con el proxy. Para un clúster de Aurora, de forma predeterminada, el grupo de destino está asociado a todas las instancias de base de datos de ese clúster. De esta forma, el proxy puede conectarse a cualquier instancia de base de datos de Aurora que se promueva para ser la instancia de escritor en el clúster. El clúster de bases de datos de Aurora asociado con un proxy se denomina destino de ese proxy. Para mayor comodidad, al crear un proxy a través de la consola, RDS Proxy también crea el grupo de destino correspondiente y registra los destinos asociados automáticamente.

Una familia de motores es un conjunto relacionado de motores de base de datos que utilizan el mismo protocolo de base de datos. Elija la familia de motores para cada proxy que cree.

Grupo de conexiones

Cada proxy realiza la agrupación de conexiones por separado para la instancia de escritor y de lector de la base de datos de Aurora asociada. La agrupación de conexiones es una optimización que reduce la sobrecarga asociada a la apertura y el cierre de conexiones y al mantenimiento de muchas conexiones abiertas simultáneamente. Esta sobrecarga incluye la memoria necesaria para gestionar cada nueva conexión. También implica una sobrecarga de la CPU para cerrar cada conexión y abrir una nueva. Los ejemplos incluyen el protocolo de enlace Transport Layer Security/Secure Sockets Layer (TLS/SSL), autenticación, capacidades de negociación, etc. La agrupación de conexiones simplifica la lógica de la aplicación. No es necesario escribir código de aplicación para minimizar el número de conexiones abiertas simultáneas.

Cada proxy también realiza multiplexación de conexión, también conocida como reutilización de la conexión. Con la multiplexación, el proxy de RDS realiza todas las operaciones de una transacción mediante una conexión de base de datos subyacente. A continuación, RDS puede usar una conexión diferente para la siguiente transacción. Puede abrir muchas conexiones simultáneas al proxy y el proxy mantiene un número menor de conexiones abiertas a la instancia de base de datos o al clúster. Al hacerlo, se minimiza aún más la sobrecarga de memoria para las conexiones en el servidor de base de datos. Esta técnica también reduce la posibilidad de errores de «demasiadas conexiones».

Seguridad de RDS Proxy

El proxy de RDS utiliza los mecanismos de seguridad de RDS existentes, como TLS/SSL e AWS Identity and Access Management (IAM). Para obtener información general acerca de esas

características de seguridad, consulte Seguridad en Amazon Aurora. Además, asegúrese de familiarizarse con la forma en la que Aurora trabaja con autenticación, autorización y otras áreas de seguridad.

RDS Proxy puede actuar como una capa adicional de seguridad entre las aplicaciones cliente y la base de datos subyacente. Por ejemplo, puede conectarse al proxy mediante TLS 1.3, incluso si la instancia de base de datos subyacente admite una versión más antigua de TLS. Puede conectarse al proxy mediante un rol de IAM, incluso si el proxy se conecta a la base de datos mediante el método de autenticación de usuario y contraseña de base de datos. Mediante esta técnica, puede imponer requisitos de autenticación sólidos para las aplicaciones de base de datos sin un esfuerzo de migración costoso para las propias instancias de base de datos.

Puede usar los siguientes métodos de autenticación con RDS Proxy:

- Credenciales de base de datos
- Autenticación de IAM estándar
- Autenticación de IAM integral

Uso de IAM con RDS Proxy

RDS Proxy ofrece dos métodos de autenticación de IAM:

- Autenticación de IAM estándar: aplique la autenticación de IAM para las conexiones al proxy mientras el proxy se conecta a la base de datos mediante las credenciales almacenadas en Secrets Manager. Esto aplica la autenticación de IAM para el acceso a la base de datos incluso si las bases de datos utilizan la autenticación de contraseña nativa. El proxy recupera las credenciales de la base de datos de Secrets Manager y gestiona la autenticación en la base de datos en nombre de la aplicación.
- Autenticación de IAM integral: aplica la autenticación de IAM a las conexiones directamente desde las aplicaciones a la base de datos a través del proxy. La autenticación de IAM integral simplifica la configuración de seguridad y evita la administración de credenciales de bases de datos en Secrets Manager. Esta capa adicional de seguridad impone un control de acceso basado en IAM desde la aplicación de cliente a la base de datos.

Para usar la autenticación de IAM estándar, configure el proxy para que utilice los secretos de Secrets Manager para la autenticación y habilite la autenticación de IAM para las conexiones de los

clientes. Las aplicaciones se autentican en el proxy mediante IAM, mientras que el proxy se autentica en la base de datos con las credenciales recuperadas de Secrets Manager.

Para utilizar la autenticación de IAM integral, configure el proxy para que utilice la autenticación de IAM al establecer el esquema de autenticación predeterminado al crear o modificar el proxy.

Para la autenticación de IAM integral, debe actualizar el rol de IAM asociado al proxy para conceder el permiso `rds-db:connect`. Con la autenticación IAM integral, se elimina la necesidad de registrar usuarios individuales de bases de datos con el proxy a través de los secretos de Secrets Manager.

Uso de TLS/SSL con RDS Proxy

Puede conectarse a RDS Proxy con el protocolo TLS/SSL.

> **Note**
>
> El proxy de RDS utiliza certificados de AWS Certificate Manager (ACM). Si está utilizando RDS Proxy, no es necesario descargar certificados de Amazon RDS ni actualizar aplicaciones que usen conexiones RDS Proxy.

Para aplicar TLS a todas las conexiones entre el proxy y la base de datos, puede especificar la configuración Exigir Transport Layer Security al crear o modificar un proxy en la Consola de administración de AWS.

RDS Proxy puede también garantizar que la sesión utiliza TLS/SSL entre el cliente y el punto de enlace de RDS Proxy. Para que RDS Proxy lo haga, especifique el requisito en el lado del cliente. Las variables de sesión SSL no están establecidas para las conexiones SSL a una base de datos usando RDS Proxy.

- En el caso de Aurora MySQL, especifique el requisito en el lado del cliente con el parámetro `--ssl-mode` cuando ejecute el comando `mysql`.
- Para y Aurora PostgreSQL, especifique `sslmode=require` como parte de la cadena `conninfo` cuando ejecute el comando `psql`.

RDS Proxy admite las versiones 1.0, 1.1, 1.2 y 1.3 del protocolo TLS. Puede conectarse al proxy mediante una versión de TLS posterior a la que utiliza en la base de datos subyacente.

De forma predeterminada, los programas del cliente establecen una conexión cifrada con RDS Proxy, con un mayor control disponible gracias a la opción `--ssl-mode`. Desde el lado del cliente, RDS Proxy es compatible con todos los modos de SSL.

Para el cliente, los modos SSL son los siguientes:

PREFERRED

SSL es la primera opción, pero no es necesaria.

DISABLED

No se permite SSL.

REQUIRED

Obliga a usar SSL.

VERIFY_CA

Implemente SSL y verifique la entidad de certificación (CA).

VERIFY_IDENTITY

Obliga a usar SSL y comprueba CA y el nombre de host de CA.

Cuando se utiliza un cliente con `--ssl-mode` `VERIFY_CA` o `VERIFY_IDENTITY`, especifique la opción de `--ssl-ca` apuntando a una autoridad certificadora en formato `.pem`. Para usar el archivo `.pem`, descargue todos los PEM de CA raíz desde Amazon Trust Services y colóquelos en un solo archivo `.pem`.

RDS Proxy utiliza certificados comodín, que se aplican tanto a un dominio como a sus subdominios. Si utiliza el cliente `mysql` para conectarse con el modo SSL `VERIFY_IDENTITY`, actualmente deberá usar el comando `mysql` compatible con MySQL 8.0.

Conmutación por error

La conmutación por error es una característica de alta disponibilidad que reemplaza una instancia de base de datos por otra cuando la instancia original deja de estar disponible. Puede producirse una conmutación por error debido a un problema con una instancia de base de datos. También es posible que sea parte de los procedimientos normales de mantenimiento, como durante la actualización de una base de datos. La conmutación por error se aplica a los clústeres de bases de datos de Aurora con una o más instancias de lector además de la instancia de escritor.

La conexión a través de un proxy hace que las aplicaciones sean más resistentes a las conmutaciones por error de la base de datos. Cuando la instancia de base de datos original deja de estar disponible, RDS Proxy se conecta a la base de datos en espera sin perder las conexiones de aplicaciones inactivas. Esto le ayuda a acelerar y simplificar el proceso de conmutación por error. Esto es menos disruptivo para la aplicación que un problema típico de reinicio o base de datos.

Sin RDS Proxy, una conmutación por error implica una breve interrupción. Durante la interrupción, no puede realizar operaciones de escritura en esa base de datos en conmutación por error. Las conexiones de base de datos existentes se interrumpen y la aplicación debe volver a abrirlas. La base de datos está disponible para nuevas conexiones y operaciones de escritura cuando se promociona una instancia de base de datos de solo lectura para que tome el lugar de la que no está disponible.

Durante las conmutaciones por error de la base de datos, RDS Proxy continúa aceptando conexiones en la misma dirección IP y dirige automáticamente las conexiones a la nueva instancia de base de datos primaria. Los clientes que se conectan a través de RDS Proxy no son susceptibles a lo siguiente:

- Retrasos de propagación del sistema de nombres de dominio (DNS) en la conmutación por error.
- Almacenamiento en caché de DNS local.
- Tiempos de espera de conexión.
- Incertidumbre sobre qué instancia de base de datos es el escritor actual.
- Espera a la respuesta de una consulta de un escritor anterior que dejó de estar disponible sin cerrar las conexiones.

En el caso de las aplicaciones que mantienen su propio grupo de conexiones, pasar por RDS Proxy significa que la mayoría de las conexiones permanecen activas durante las conmutaciones por error u otras interrupciones. Solo se cancelan las conexiones que están en medio de una transacción o sentencia SQL. El proxy de RDS acepta inmediatamente nuevas conexiones. Cuando el escritor de la base de datos no está disponible, RDS Proxy pone en cola las solicitudes entrantes.

Para aplicaciones que no mantienen sus propios grupos de conexiones, RDS Proxy ofrece velocidades de conexión más rápidas y conexiones más abiertas. Descarga la costosa sobrecarga de reconexiones frecuentes de la base de datos. Lo hace reutilizando las conexiones de base de datos que se mantienen en el grupo de conexiones de RDS Proxy. Este enfoque es especialmente importante para las conexiones TLS, en las que los costos de instalación son importantes.

Transacciones

Todas las instrucciones dentro de una sola transacción siempre utilizan la misma conexión de base de datos subyacente. La conexión está disponible para su uso por parte de una sesión diferente cuando finaliza la transacción. El uso de la transacción como unidad de granularidad tiene las siguientes consecuencias:

- La reutilización de la conexión puede ocurrir después de cada instrucción individual cuando se ha activado la configuración `autocommit` de Aurora MySQL.
- Por el contrario, cuando el parámetro `autocommit` está desactivado, la primera instrucción que emita en una sesión comienza una nueva transacción. Por ejemplo, suponga que introduce una secuencia de `SELECT`, `INSERT`, `UPDATE` y otras instrucciones de lenguaje de manipulación de datos (DML). En este caso, la reutilización de la conexión no se producirá hasta que emita un`COMMIT`, `ROLLBACK` o finalice la transacción.
- La introducción de una instrucción de lenguaje de definición de datos (DDL) hace que la transacción finalice después de que se complete esa instrucción.

RDS Proxy detecta cuándo finaliza una transacción a través del protocolo de red utilizado por la aplicación cliente de base de datos. La detección de transacciones no se basa en palabras clave como `COMMIT` o `ROLLBACK` que aparecen en el texto de la instrucción SQL.

En algunos casos, RDS Proxy podría detectar una solicitud de base de datos que hace que sea poco práctico trasladar la sesión a una conexión diferente. En estos casos, desactiva la multiplexación para esa conexión el resto de la sesión. La misma regla se aplica si RDS Proxy no puede estar seguro de que la multiplexación sea práctica para la sesión. Esta operación se denomina fijación. Para obtener información sobre formas de detectar y minimizar la fijación, consulte Cómo evitar la fijación de RDS Proxy.

Introducción al proxy de RDS

Utilice la información de las siguientes páginas para configurar y administrar Amazon RDS Proxy para Aurora, así como para definir las opciones de seguridad relacionadas. Estas opciones de seguridad controlan quién puede acceder a cada proxy y cómo se conecta cada proxy a instancias de base de datos.

Si es la primera vez que utiliza RDS Proxy, le recomendamos que siga las páginas en el orden en que las presentamos.

Temas

- Configuración de requisitos previos de red para RDS Proxy
- Configuración de credenciales de base de datos para RDS Proxy
- Configuración de la autenticación de IAM para RDS Proxy
- Creación de un proxy para Amazon Aurora
- Visualización de un proxy
- Conexión a una base de datos mediante RDS Proxy

Configuración de requisitos previos de red para RDS Proxy

El uso de RDS Proxy requiere que tenga una nube privada virtual (VPC) común entre su clúster de base de datos de Aurora y RDS Proxy. Esta VPC debe tener un mínimo de dos subredes que se encuentren en diferentes zonas de disponibilidad. Tu cuenta puede poseer estas subredes o compartirlas con otras cuentas. Para obtener más información acerca del uso compartido de VPC, consulte Trabajar con VPC compartidas.

Para la compatibilidad con IPv6, se requiere una configuración de red adicional:

- Tipos de redes de puntos de conexión de IPv6: la VPC y las subredes deben estar configuradas para admitir IPv6. Esto incluye tener bloques de CIDR de IPv6 asignados a la VPC y las subredes.
- Tipos de redes de puntos de conexión de doble pila: la VPC y las subredes deben admitir direcciones IPv4 e IPv6.
- Tipos de redes de conexión de destino de IPv6: la base de datos debe estar configurada en modo de doble pila para admitir conexiones de IPv6 desde el proxy.

Los recursos de aplicaciones cliente, como Amazon EC2, Lambda o Amazon ECS, pueden estar en la misma VPC como el proxy. O pueden estar en una VPC independiente del proxy. Si se ha conectado correctamente a clústeres de bases de datos de Aurora, ya dispone de los recursos de red necesarios.

Temas

- Obtención de información sobre subredes.
- Planificación de la capacidad de direcciones IP

Obtención de información sobre subredes.

Si acaba de empezar a utilizar Aurora, puede obtener información sobre los conceptos básicos de la conexión a una base de datos siguiendo los procedimientos de Configuración del entorno para Amazon Aurora. También puede seguir el tutorial de Introducción a Amazon Aurora.

Para crear un proxy, debe proporcionar las subredes y la VPC en las que opera el proxy. En el siguiente ejemplo de Linux se muestran comandos de la AWS CLI con los que se examinan las VPC y las subredes que son propiedad de su Cuenta de AWS. En particular, se pasan los ID de subred como parámetros si crea un proxy utilizando la CLI.

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

En el siguiente ejemplo de Linux, se muestran comandos de la AWS CLI que sirven para determinar los ID de subred correspondientes a un clúster de base de datos de Aurora.

Para un clúster de Aurora, en primer lugar encuentra el ID de una de las instancias de base de datos asociadas. Puede extraer los ID de subred utilizados por esa instancia de base de datos. Para ello, examine los campos anidados que hay dentro de los atributos `DBSubnetGroup` y `Subnets` en la salida la instancia de descripción de la instancia de base de datos. Especifica algunos o todos esos ID de subred cuando establece un proxy para ese servidor de base de datos.

```
$ # Find the ID of any DB instance in the cluster.
$  aws rds describe-db-clusters --db-cluster-identifier my_cluster_id --query '*[].
[DBClusterMembers]||[0]||[0][*].DBInstanceIdentifier' --output text
```

```
my_instance_id
instance_id_2
instance_id_3
```

Una vez encontrado el identificador de la instancia de base de datos, examine la VPC asociada para encontrar sus subredes. En el siguiente ejemplo de Linux se muestra cómo.

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0]||[Subnets]||[0]||[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]|[0]|[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
subnet_id_4
subnet_id_5
subnet_id_6
```

Planificación de la capacidad de direcciones IP

RDS Proxy ajusta automáticamente su capacidad en función de la configuración de las instancias de base de datos registradas en él. Para las instancias aprovisionadas, esto se determina por el tamaño de la instancia y, en el caso de las instancias de Aurora Serverless v2, por la capacidad máxima de la ACU. Determinadas operaciones también pueden requerir una capacidad adicional, como el aumento del tamaño de una base de datos registrada u operaciones de mantenimiento internas de RDS Proxy. Durante estas operaciones, es posible que el proxy necesite más direcciones IP para aprovisionar la capacidad adicional. Estas direcciones adicionales permiten que su proxy escale sin afectar a su carga de trabajo. La falta de direcciones IP gratuitas en las subredes impide que un proxy se amplíe. Esto puede provocar latencias de consulta más altas o errores en la conexión del cliente. RDS le notifica mediante un evento `RDS-EVENT-0243` cuando no hay suficientes direcciones IP libres en sus subredes. Para obtener información acerca de este evento, consulte Trabajo con eventos de RDS Proxy.

Reserve el número mínimo siguiente de direcciones IP libres en las subredes para el proxy, en función de los tamaños de las clases de instancias de bases de datos.

Clase de instancia de base de datos	Direcciones IP gratuitas mínimas
db.*.xlarge o menor	10
db.*.24xlarge	15
db.*.24xlarge	25
db.*.24xlarge	45
db.*.24xlarge	60
db.*.24xlarge	75
db.*.24xlarge	110

Para Aurora Serverless v2, reserve el número mínimo siguiente de direcciones IP libres en las subredes para el proxy, en función de la capacidad de ACU máxima.

`Maximum ACU Capacity`	Direcciones IP gratuitas mínimas
16 o menos	10
32	15
64	25
96	30
128	40
160	50
192	55
224	65
256	75

> **Note**
>
> RDS Proxy no consume más de 215 direcciones IP para cada proxy en una VPC.

RDS Proxy requiere un mínimo de 10 direcciones IP para la base de datos de Aurora. Estos números de direcciones IP recomendados son estimaciones para un proxy con solo el punto de conexión predeterminado. Para cada punto de conexión personalizado adicional, le recomendamos que reserve tres direcciones IP más. Para cada instancia de lector de Aurora, le recomendamos que reserve direcciones IP adicionales tal como se especifica en la tabla en función de las ACU máximas de dicho lector para el destino de Aurora Serverless v2 o el tamaño de la instancia de base de datos para un destino aprovisionado.

Para estimar las direcciones IP necesarias para un proxy asociado a un clúster de base de datos de Aurora con:

- 1 instancia de escritor aprovisionada de tamaño `db.r5.8xlarge` y 1 instancia de lector aprovisionada de tamaño `db.r5.2xlarge`.
- El proxy adjunto a este clúster tiene un punto de conexión predeterminado y un punto de conexión personalizado con el rol de solo lectura.

En este caso, el proxy necesita aproximadamente 63 direcciones IP libres (45 para la instancia de escritor, 15 para la instancia de lector y 3 para el punto de conexión personalizado adicional).

Para estimar las direcciones IP necesarias para un proxy asociado a un clúster de base de datos de Aurora que tiene:

- 1 instancia escritor de Aurora Serverless v2 con una capacidad máxima de 256 ACU y 1 instancia de lector v2 sin servidor con una capacidad máxima de 192 ACU.
- El proxy que está adjunto a este clúster tiene el punto de conexión predeterminado y 1 punto de conexión personalizado con la función de solo lectura.

En este caso, el proxy necesita aproximadamente 133 direcciones IP libres (75 para la instancia de escritor, 55 para la instancia de lector y 3 para el punto de conexión personalizado adicional).

Para estimar las direcciones IP necesarias para un proxy asociado a un clúster de Aurora que tiene:

- 1 instancia de escritor aprovisionada con un tamaño de instancia de base de datos de db.r5.4xlarge y 1 instancia de lector v2 sin servidor con una capacidad máxima de 64 ACU.
- El proxy que está adjunto a este clúster tiene el punto de conexión predeterminado y 1 punto de conexión personalizado con la función de solo lectura.

En este caso, el proxy necesita aproximadamente 53 direcciones IP libres (25 para la instancia de escritor, 25 para la instancia de lector y 3 para el punto de conexión personalizado adicional).

Para estimar las direcciones IP necesarias para un proxy asociado a un clúster de base de datos de Aurora que tiene:

- 1 instancia de escritor aprovisionada de tamaño db.r5.24xlarge y 3 instancias de lector aprovisionadas de tamaño db.r5.8xlarge.
- El proxy que está adjunto a este clúster de base de datos tiene el punto final predeterminado y un punto de conexión personalizado con la función de solo lectura.

En este caso, el proxy necesita 215 direcciones IP libres. Aunque los cálculos sugieren 248 IP (110 + (3*45) + 3), RDS Proxy no consume más de 215 direcciones IP por cada proxy de una VPC.

Configuración de credenciales de base de datos para RDS Proxy

RDS Proxy en Amazon RDS utiliza AWS Secrets Manager para almacenar y administrar las credenciales de la base de datos de forma segura. En lugar de incrustar credenciales en la aplicación, se asocia un proxy con un secreto de Secrets Manager que contiene los detalles de autenticación necesarios. Cree un secreto independiente de Secrets Manager para cada cuenta de usuario de base de datos a la que se conecta el proxy en el clúster de bases de datos de Aurora.

Otra opción, puede configurar RDS Proxy para utilizar la autenticación de IAM integral, lo que elimina la necesidad de almacenar las credenciales de la base de datos en Secrets Manager. RDS Proxy utiliza la autenticación de IAM para las conexiones de cliente a proxy y de proxy a base de datos. Esto proporciona una solución de autenticación basada en IAM totalmente integrada que no requiere administrar secretos ni contraseñas. Para obtener información sobre cómo agregar un usuario de base de datos de IAM nuevo, consulte Creación de cuentas de base de datos utilizando autenticación de IAM.

Temas

- Creación de secretos para utilizar con RDS Proxy

Creación de secretos para utilizar con RDS Proxy

Antes de crear un proxy, debe crear al menos un secreto que almacene las credenciales de la base de datos.

Consola

Creación de un secreto

1. Abra la consola de Secrets Manager enhttps://console.aws.amazon.com/secretsmanager/.
2. Elija Almacenar un secreto nuevo.
3. Elija Credenciales para base de datos de Amazon RDS.
4. Introduzca un nombre de usuario y una contraseña. Las credenciales que introduzca deben coincidir con las credenciales de un usuario de base de datos que exista en la base de datos de RDS asociada. RDS Proxy utiliza estas credenciales para autenticar y establecer conexiones con la base de datos en nombre de las aplicaciones.

   Si no hay coincidencia, puede actualizar el secreto para que coincida con la contraseña de la base de datos. Hasta que no actualice el secreto, los intentos de conexión a través del proxy mediante ese secreto no se realizarán correctamente, pero las conexiones que utilicen otros secretos válidos seguirán funcionando.

   > **Note**
   >
   > En el caso de RDS para SQL Server, RDS Proxy requiere un secreto que distinga entre mayúsculas y minúsculas en Secrets Manager, independientemente de la configuración de intercalación de la instancia de base de datos. Si la aplicación permite nombres de usuario con distintas mayúsculas, como “Admin” y “admin”, deberá crear secretos distintos para cada uno. RDS Proxy no admite la autenticación de nombre de usuario sin distinción entre mayúsculas y minúsculas entre el cliente y el proxy.
   > Para obtener más información sobre colación en SQL Server, consulte la documentación de Microsoft SQL Server.

5. En Base de datos, seleccione la base de datos de Amazon RDS a la que accederá el secreto.
6. Rellene otras configuraciones para el secreto y elija Guardar. Para obtener instrucciones completas, consulte Creación de un secreto de AWS Secrets Manager en la Guía del usuario de AWS Secrets Manager.

AWS CLI

Cuando cree un proxy a través de la AWS CLI, especifique los nombres de recursos de Amazon (ARN) de los secretos correspondientes. Lo hace para todas las cuentas de usuario de base de datos a las que puede acceder el proxy. En la Consola de administración de AWS, elija los secretos por sus nombres descriptivos.

- Para crear un secreto de Secrets Manager para utilizarlo con RDS Proxy, utilice el comando create-secret:

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- También puede crear una clave personalizada para cifrar su secreto de Secrets Manager. El siguiente comando crea una clave de ejemplo.

```
aws kms create-key --description "test-key" --policy '{
  "Id":"kms-policy",
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Sid":"Enable IAM User Permissions",
        "Effect":"Allow",
        "Principal":{"AWS":"arn:aws:iam::account_id:root"},
        "Action":"kms:*","Resource":"*"
      },
      {
        "Sid":"Allow access for Key Administrators",
        "Effect":"Allow",
        "Principal":
          {
            "AWS":
              ["$USER_ARN","arn:aws:iam:account_id::role/Admin"]
          },
        "Action":
          [
            "kms:Create*",
            "kms:Describe*",
```

```
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms:Delete*",
            "kms:TagResource",
            "kms:UntagResource",
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion"
          ],
        "Resource":"*"
      },
      {
        "Sid":"Allow use of the key",
        "Effect":"Allow",
        "Principal":{"AWS":"$ROLE_ARN"},
        "Action":["kms:Decrypt","kms:DescribeKey"],
        "Resource":"*"
      }
    ]
}'
```

Por ejemplo, los siguientes comandos crean secretos de Secrets Manager para dos usuarios de bases de datos:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

Para crear estos secretos cifrados con su clave de AWS KMS personalizada, use los siguientes comandos:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
```

```
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

Para ver los secretos que posee la cuenta de AWS, utilice el comando list-secrets:

```
aws secretsmanager list-secrets
```

Si crea un proxy mediante la CLI, se pasan los nombres de recursos de Amazon (ARN) de uno o más secretos al parámetro `--auth`. En el siguiente ejemplo se muestra cómo preparar un informe si solo se tiene el nombre y el ARN de cada secreto que es propiedad de la cuenta de AWS. En este ejemplo se utiliza el parámetro `--output table`, disponible en la versión 2 de la AWS CLI. Si está utilizando la versión 1 de la AWS CLI, use `--output text`.

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

Para confirmar que el secreto contiene las credenciales correctas en el formato adecuado, utilice el comando get-secret-value. Reemplace *your_secret_name* por el nombre corto o ARN del secreto.

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

La salida contiene una línea con un valor codificado en JSON similar al siguiente:

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

Configuración de la autenticación de IAM para RDS Proxy

Para configurar la autenticación de AWS Identity and Access Management (IAM) para RDS Proxy en Amazon RDS, cree y configure una política de IAM que conceda los permisos necesarios.

En este tema se proporcionan los pasos para configurar la autenticación de IAM para RDS Proxy, incluida la creación de la política de IAM necesaria y la asociación a un rol de IAM.

> **Tip**
>
> Este procedimiento solo es necesario si desea crear un rol de IAM propio. De lo contrario, RDS puede crear automáticamente el rol requerido cuando configure el proxy, por lo que puede omitir estos pasos.

Requisitos previos

Antes de configurar la autenticación de IAM para RDS Proxy, asegúrese de que tiene lo siguiente:

- AWS Secrets Manager: al menos un secreto almacenado que contenga credenciales de base de datos. Para obtener instrucciones para crear secretos, consulte the section called “Configuración de credenciales de base de datos”.

  Esto no es obligatorio si utiliza la autenticación de IAM integral.
- Permisos de IAM: un rol o usuario de IAM con permisos para crear y administrar políticas, roles y secretos de IAM en AWS Secrets Manager.

Creación de una política de IAM para la autenticación de IAM integral

Cuando se utiliza la autenticación de IAM integral, RDS Proxy se conecta a la base de datos mediante la autenticación de IAM en lugar de recuperar las credenciales de Secrets Manager. Para ello, es necesario configurar el rol de IAM con permisos `rds-db:connect` para las cuentas de base de datos que desee utilizar con el proxy.

Para autenticar RDS Proxy en la base de datos mediante IAM, cree un rol de IAM con una política que conceda los permisos de conexión a la base de datos necesarios.

Consola

Creación de un rol para la autenticación de IAM integral con el proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Cree una política de permisos para el rol. Para conocer los pasos generales, consulte Creación de políticas de IAM (consola).

   Pegue esta política en el editor JSON y realice los siguientes cambios:

- Sustituya su propio ID de cuenta.
- Sustituya `us-east-2` por el lugar donde debe residir el proxy.
- Sustituya los ID de los recursos de la base de datos y los nombres de usuario por los que desee utilizar. El formato del ID de recurso difiere entre las instancias de RDS y Aurora clusters.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}
```

3. Cree el rol y asocie la política de permisos a este. Para conocer los pasos generales, consulte Creación de un rol para delegar permisos a un servicio de AWS.

   En Tipo de entidad de confianza, elija Servicio de AWS. En Caso de uso, seleccione RDS y elija RDS: agregar rol a la base de datos para el caso de uso.

4. En Políticas de permisos, elija la política que creó.
5. En Seleccionar entidades de confianza, introduzca la siguiente política de confianza para el rol:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Para crear el rol mediante la AWS CLI, envíe la siguiente solicitud:

```
aws iam create-role \
  --role-name my_e2e_iam_role_name \

  --assume-role-policy-document '{"Version":"2012-10-17",
     "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

A continuación, asocie la política al rol:

```
aws iam put-role-policy \
  --role-name my_e2e_iam_role_name \
  --policy-name e2e_iam_db_connect_policy \
  --policy-document '{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}'
```

Una vez configurados el rol de IAM y los permisos para la autenticación de IAM integral, puede crear ahora un proxy con `DefaultAuthScheme` configurado en `IAM_AUTH`. Este proxy se autentica

directamente en la base de datos mediante IAM sin requerir los secretos de Secrets Manager. Para obtener instrucciones, consulte the section called “Creación de un proxy”.

Cuando utilice la autenticación de IAM integral, asegúrese de que los usuarios de la base de datos estén configurados para la autenticación de IAM tal y como se describe en Creación de cuentas de base de datos utilizando autenticación de IAM.

Creación de una política de IAM para acceder a Secrets Manager

Para permitir que RDS Proxy recupere las credenciales de la base de datos de Secrets Manager, cree un rol de IAM con una política que conceda los permisos necesarios.

Consola

Creación de un rol que acceda a los secretos que desea utilizar con el proxy

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Cree una política de permisos para el rol. Para conocer los pasos generales, consulte Creación de políticas de IAM (consola).

   Pegue esta política en el editor JSON y realice los siguientes cambios:

   - Sustituya su propio ID de cuenta.
   - Sustituya `us-east-2` por la región donde residirá el proxy.
   - Sustituya los nombres secretos por los que ha creado. Para obtener más información, consulte Especificación de claves de KMS en instrucciones de política de IAM.
   - Sustituya el ID de clave de KMS por el que utilizó para cifrar los secretos de Secrets Manager, ya sea la clave predeterminada o una clave propia.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
```

```
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_1",
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:111122223333:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-
east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. Cree el rol y asocie la política de permisos a este. Para conocer los pasos generales, consulte Creación de un rol para delegar permisos a un servicio de AWS.

   En Tipo de entidad de confianza, elija Servicio de AWS. En Caso de uso, seleccione RDS y elija RDS: agregar rol a la base de datos para el caso de uso.

4. En Políticas de permisos, elija la política que creó.

5. En Seleccionar entidades de confianza, introduzca la siguiente política de confianza para el rol:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

AWS CLI

Para crear el rol mediante la AWS CLI, envíe la siguiente solicitud:

```
aws iam create-role \
  --role-name my_role_name \
  --assume-role-policy-document '{"Version": "2012-10-17",
      "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

A continuación, asocie la política al rol:

```
aws iam put-role-policy \
  --role-name my_role_name \
  --policy-name secret_reader_policy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}'
```

Una vez configurados el rol de IAM y los permisos, puede crear un proxy y asociarlo a este rol. Esto permite al proxy recuperar de forma segura las credenciales de la base de datos de AWS Secrets Manager y habilitar la autenticación de IAM para las aplicaciones. Para obtener instrucciones, consulte the section called “Creación de un proxy”.

Creación de un proxy para Amazon Aurora

Puede utilizar Amazon RDS Proxy para mejorar la escalabilidad, la disponibilidad y la seguridad de las aplicaciones de bases de datos agrupando las conexiones y administrando las conmutaciones por error de las bases de datos de forma más eficiente. Este tema le guía por el proceso de creación de un proxy. Antes de empezar, asegúrese de que la base de datos cumpla los requisitos previos necesarios, incluidos los permisos de IAM y la configuración de la VPC.

Puede asociar un proxy con un clúster de base de datos de Aurora MySQL o Aurora PostgreSQL.

Consola

Para crear un proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Proxies.
3. Elija Create proxy (Crear proxy).
4. Configure los siguientes ajustes para el proxy.

Ajuste	Descripción
Familia de motores	El protocolo de red de base de datos reconoce el proxy cuando interpreta el tráfico de red hacia y desde la base de datos. **Note** Para usar Aurora PostgreSQL, asegúrese de retener la base de datos de `postgres` en la instancia. Consulte Se ha eliminado la solución de problemas de una base de datos de `postgres`.

Ajuste	Descripción
Identificador de proxy	Un nombre que sea único dentro del ID de cuenta de AWS y de la región de AWS actual.
Tiempo de espera de inactividad de conexión de cliente	El proxy cierra la conexión de un cliente si permanece inactiva durante un periodo determinado. De forma predeterminada, esto es 1800 segundos (30 minutos). Una conexión se considera inactiva cuando la aplicación no envía una nueva solicitud dentro del plazo especificado después de completar la solicitud anterior. El proxy mantiene abierta la conexión de base de datos subyacente y la devuelve al grupo de conexiones, de modo que esté disponible para nuevas conexiones de clientes. Para eliminar de forma proactiva las conexiones obsoletas, reduzca el tiempo de espera de conexión de cliente inactivo. Para minimizar los costos de conexión durante los picos de carga de trabajo, aumente el tiempo de espera.
Base de datos	Clúster de base de datos de Aurora para acceder a través de este proxy. La lista solo incluye instancias de base de datos y clústeres con motores de base de datos compatibles, versiones de motor y otras configuraciones. Si la lista está vacía, cree una nueva instancia de base de datos o clúster que sea compatible con RDS Proxy. Para ello, siga el procedimiento en Creación de un clúster de base de datos de Amazon Aurora. A continuación, intente volver a crear el proxy.
Conexiones máximas de grupo de conexión	Un valor entre 1 y 100 para definir el porcentaje del límite de `max_connections` que RDS Proxy puede utilizar. Si solo tiene la intención de utilizar un proxy con esta instancia de base de datos o clúster, establezca este valor en 100. Para obtener más información sobre cómo RDS Proxy usa esta configuración, consulte the section called “MaxConnectionsPercent”.

Ajuste	Descripción
Filtros de fijación de sesión	Impide que RDS Proxy fije determinados estados de sesión detectados, lo que evita las medidas de seguridad predeterm inadas para las conexiones de multiplexación. Actualmente, PostgreSQL no admite esta configuración y la única opción disponible es `EXCLUDE_VARIABLE_SETS` . Si se habilita, es posible que las variables de sesión de una conexión afecten a otras, lo que puede provocar errores o problemas de correcció n si las consultas dependen de variables de sesión establecidas fuera de la transacción actual. Utilice esta opción solo después de confirmar que las aplicaciones pueden compartir conexiones de bases de datos de forma segura. Los siguientes patrones se consideran seguros: • Instrucciones `SET` donde no hay ningún cambio en el valor efectivo de la variable de sesión. En otras palabras, no hay ningún cambio en la variable de sesión. • Cambia el valor de la variable de sesión y ejecuta una instrucción en la misma transacción. Para obtener más información, consulte Cómo evitar la fijación de RDS Proxy.
Tiempo de espera de préstamo de conexión	Si espera que el proxy utilice todas las conexiones de base de datos disponibles, establezca el tiempo de espera antes de que devuelva un error de tiempo de espera. Puede especificar hasta cinco minutos. Esta configuración solo se aplica cuando el proxy ha alcanzado el número máximo de conexiones y todas están en uso.

Ajuste	Descripción
Consulta de inicialización	Añada una consulta de inicialización o modifique la actual (opcional). Puede especificar una o más instrucciones de SQL para que el proxy se ejecute al abrir cada nueva conexión de base de datos. Normalmente, el ajuste se utiliza con instrucciones de SET para garantizar que cada conexión tenga una configuración idéntica. Asegúrese de que la consulta que añada sea válida. Para incluir varias variables en una única instrucción de SET, utilice separadores de coma. Por ejemplo: `SET variable1 =value1, variable2 =value2` Para varias instrucciones, utilice punto y coma como separador. **Important** Dado que se puede acceder a la consulta de inicialización como parte de la configuración del grupo de destino, no está protegida por métodos de autenticación o criptográficos. Cualquier persona con acceso para ver o administrar la configuración del grupo de destino del proxy puede ver la consulta de inicialización. No debe agregar información confidencial, como contraseñas o claves de cifrado de larga duración, a esta opción.
AWS Identity and Access Management (Rol de (IAM	Un rol de IAM con permiso para acceder a los secretos de Secrets Manager, que representan las credenciales de las cuentas de usuario de la base de datos que puede usar el proxy. Otra opción, puede crear un rol de IAM nuevo desde la Consola de administración de AWS.

Ajuste	Descripción
Secretos de Secrets Manager	Cree o elija secretos de Secrets Manager que representen las credenciales de las cuentas de los usuarios de bases de datos que pueden usar el proxy. Si el esquema de autenticación predeterminado está establecido en Ninguno, este campo es obligatorio. Cuando el esquema de autenticación predeterminado se establece en la autenticación de IAM, este campo pasa a ser opcional y se marca como tal en la consola. Puede elegir uno o más secretos del menú desplegable o crear uno nuevo mediante el enlace Crear un nuevo secreto.
Tipo de autenticación de cliente	El tipo de autenticación que utiliza el proxy para las conexione s de los clientes. Su elección se aplica a todos los secretos de Secrets Manager que asocie a este proxy. Si tiene que especificar un tipo de autenticación de cliente diferente para cada secreto, cree su proxy mediante la AWS CLI o la API. Especifique esta opción solo cuando la conexión del cliente utilice las credenciales de la base de datos para la autentica ción.
Autenticación de IAM	Especifique Obligatorio o No Permitido para la autenticación de IAM para las conexiones al proxy. Su elección se aplica a todos los secretos de Secrets Manager que asocie a este proxy. Si tiene que especificar un tipo de autenticación de IAM diferente para cada secreto, cree su proxy mediante la AWS CLI o la API.

Ajuste	Descripción
Esquema de autenticación predeterminado	Elija el tipo de autenticación predeterminado que utiliza el proxy para las conexiones del cliente al proxy y las conexiones del proxy a la base de datos subyacente. Dispone de las opciones siguientes: • Ninguno (predeterminado): el proxy recupera las credenciales de la base de datos de los secretos de Secrets Manager. • Autenticación de IAM: el proxy utiliza la autenticación de IAM para conectarse a la base de datos, lo que permite la autenticación de IAM integral. Al seleccionar la autenticación de IAM, aparece una alerta de información que le recuerda que debe habilitar la autenticación de las bases de datos de IAM para las bases de datos de la configuración del grupo de destino. **Note** Esta opción solo es compatible con familias de motor de MySQL, PostgreSQL y MariaDB.

Ajuste	Descripción
Cuentas de base de datos para la autenticación de IAM	Este campo solo aparece cuando el esquema de autenticación predeterminado está establecido en la autenticación de IAM y el rol de Identity and access management (IAM) está establecido en Crear un rol de IAM. Asigne un nombre a las cuentas de usuario de la base de datos para que el proxy las utilice con la autenticación de IAM. Este campo es obligatorio. Especifique varias cuentas de la siguiente manera: • Escribiendo la escritura de un nombre de usuario de la base de datos para agregarlo como etiqueta • Usando nombres de usuario de bases de datos específicos (por ejemplo, `db_user`, `jane_doe`) • Usando patrones comodín para varios usuarios (por ejemplo, `db_test_*`) Cada cuenta aparece como una etiqueta extraíble que puede eliminar haciendo clic en el icono X. La consola utiliza estos valores para crear los permisos `rds-db:connect` adecuados en la política del rol de IAM.
Requerir seguridad de capa de transporte	Aplica TLS/SSL para todas las conexiones de cliente. El proxy utiliza la misma configuración de cifrado para su conexión a la base de datos subyacente, si la conexión del cliente se cifra o no se cifra.

Ajuste	Descripción
Tipo de red de conexión de destino	La versión de IP que utiliza el proxy para conectarse a la base de datos de destino. Puede elegir entre las siguientes opciones: • IPv4: el proxy se conecta a la base de datos mediante direcciones IPv4. • IPv6: el proxy se conecta a la base de datos mediante direcciones IPv6. El valor predeterminado es IPv4. Para utilizar IPv6, la base de datos debe admitir el modo de pila doble. El modo de doble pila no está disponible para conexiones de destino.
Tipo de red de punto de conexión	La versión de IP del punto de conexión del proxy que los clientes utilizan para conectarse al proxy. Puede elegir entre las siguientes opciones: • IPv4: el punto de conexión del proxy utiliza solo direcciones IPv4. • IPv6: el punto de conexión del proxy utiliza solo direcciones IPv6. • Doble pila: el punto de conexión del proxy admite direcciones IPv4 y IPv6. El valor predeterminado es IPv4. Para usar IPv6 o de doble pila, la VPC y las subredes deben estar configuradas para admitir el tipo de red seleccionado.
Subredes	Este campo se rellena previamente con todas las subredes asociadas a la VPC. Puede eliminar cualquier subred que no sea necesaria para el proxy, pero debe dejar al menos dos subredes. Para los tipos de redes de punto de conexión de IPv6 o de doble pila, asegúrese de que las subredes seleccionadas admitan el tipo de red elegido.

Ajuste	Descripción
VPC security group (Grupo de seguridad de VPC	Elija un grupo de seguridad de VPC existente o cree uno nuevo desde la Consola de administración de AWS. Configure las reglas de entrada para permitir que las aplicaciones accedan al proxy y las reglas de salida para permitir el tráfico desde los destinos de base de datos. **Note** El grupo de seguridad debe permitir conexiones desde el proxy a la base de datos. Sirve para la entrada de las aplicaciones al proxy y para la salida del proxy a la base de datos. Por ejemplo, si utiliza el mismo grupo de seguridad para la base de datos y el proxy, asegúrese de que los recursos dentro de ese grupo de seguridad puedan comunicarse entre sí. Cuando se utiliza una VPC compartida, evite el uso del grupo de seguridad predeterminado para la VPC o uno asociado a otra cuenta. En su lugar, elija un grupo de seguridad que pertenezca a la cuenta. Si no existe ninguno, créelo. Para obtener más información, consulte Trabajar con VPC compartidas. RDS despliega un proxy en varias zonas de disponibilidad para garantizar una alta disponibilidad. Para habilitar la comunicac ión entre zonas de disponibilidad, la lista de control de acceso (ACL) a la red de la subred del proxy debe permitir la salida específica en el puerto del motor y la entrada en todos los puertos. Para obtener más información acerca de las ACL de red de, consulte Controlar el tráfico hacia las subredes utilizand o las ACL de red. Si la ACL de red del proxy y la de destino son idénticas, debe añadir una regla de entrada del protocolo TCP en la que el Origen esté configurado en el CIDR de la VPC. También debe añadir una regla de salida del protocolo

Ajuste	Descripción
	TCP específica del puerto del motor en la que el Destino esté configurado en el CIDR de la VPC.
Activación del registro mejorado	Habilite esta configuración para solucionar problemas de compatibilidad de proxy o rendimiento. Cuando está habilitado, RDS Proxy registra información sobre el rendimiento detallada para ayudarle a depurar el comportamiento de SQL o el rendimiento y la escalabilidad de la conexión del proxy. Habilite esta configuración solo para la depuración y asegúrese de que existan las medidas de seguridad adecuadas para proteger la información confidencial en los registros. Para minimizar la sobrecarga, RDS Proxy desactiva automáticamente esta configuración 24 horas después de la activación. Úsela temporalmente para solucionar problemas específicos.

5. Elija Create proxy (Crear proxy).

AWS CLI

Para crear un proxy utilizando el comando AWS CLI, llame al comando create-db-proxy con los siguientes parámetros requeridos:

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--vpc-subnet-ids`

El valor `--engine-family` distingue entre mayúsculas y minúsculas.

Example

Para Linux, macOS o Unix:

```
aws rds create-db-proxy \
    --db-proxy-name proxy_name \
```

```
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
    [--default-auth-scheme { NONE | IAM_AUTH }] \
    [--auth ProxyAuthenticationConfig_JSON_string] \
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] \
    [--target-connection-network-type { IPV4 | IPV6 }] \
    [--tags comma_separated_list]
```

Para Windows:

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--default-auth-scheme { NONE | IAM_AUTH }] ^
    [--auth ProxyAuthenticationConfig_JSON_string] ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] ^
    [--target-connection-network-type { IPV4 | IPV6 }] ^
    [--tags comma_separated_list]
```

A continuación se muestra un ejemplo del valor JSON para la opción `--auth`. Este ejemplo aplica un tipo de autenticación de cliente diferente a cada secreto.

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },
```

```
  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },

  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

El parámetro `--endpoint-network-type` especifica la versión de IP del punto de conexión del proxy que los clientes utilizan para conectarse al proxy. Los valores válidos son:

- `IPV4`: el punto de conexión del proxy utiliza solo direcciones IPv4 (predeterminado).
- `IPV6`: el punto de conexión del proxy utiliza solo direcciones IPv6.
- `DUAL`: el punto de conexión del proxy admite direcciones IPv4 y IPv6.

El parámetro `--target-connection-network-type` especifica la versión de IP que utiliza el proxy para conectarse a la base de datos de destino. Los valores válidos son:

- `IPV4`: el proxy se conecta a la base de datos mediante direcciones IPv4 (predeterminado).
- `IPV6`: el proxy se conecta a la base de datos mediante direcciones IPv6.

Para usar los tipos de redes de punto de conexión de IPv6 o de doble pila, la VPC y las subredes deben estar configuradas para admitir el tipo de red seleccionado. Para utilizar el tipo de red de conexión de destino de IPv6, la base de datos debe admitir el modo de doble pila.

> **Tip**
>
> Si aún no conoce los ID de subred que utilizar para el parámetro `--vpc-subnet-ids`, consulte Configuración de requisitos previos de red para RDS Proxy para ver ejemplos de cómo encontrarlos.

> **Note**
>
> Este grupo de seguridad debe permitir el acceso a la base de datos a la que se conecta el proxy. El mismo grupo de seguridad se utiliza para la entrada de las aplicaciones al proxy y para la salida del proxy a la base de datos. Por ejemplo, supongamos que utiliza el mismo grupo de seguridad para la base de datos y el proxy. En este caso, asegúrese de especificar que los recursos de ese grupo de seguridad pueden comunicarse con otros recursos del mismo grupo de seguridad.
>
> Cuando se utiliza una VPC compartida, no se puede utilizar el grupo de seguridad predeterminado para la VPC o uno que pertenezca a otra cuenta. Elija un grupo de seguridad que pertenezca a su cuenta. Si no existe uno, créelo. Para obtener más información acerca de esta limitación, consulte Trabajar con VPC compartidas.

Para crear las asociaciones adecuadas para el proxy, utilice también el comando register-db-proxy-targets. Especifique el nombre de grupo de destino de `default`. El proxy de RDS crea automáticamente un grupo de destino con este nombre cuando crea cada proxy.

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

API de RDS

Para crear un proxy de RDS, llame a la operación de la API de Amazon RDS CreateDBProxy. Transfiera un parámetro con la estructura de datos AuthConfig .

El proxy de RDS crea automáticamente un grupo de destino denominado `default` cuando crea cada proxy. Se asocia un clúster de bases de datos de Aurora con el grupo de destino llamando a la función RegisterDBProxyTargets.

> **Important**
>
> Al seleccionar la autenticación de IAM para el esquema de autenticación predeterminado:
>
> - Debe habilitar la autenticación de la base de datos de IAM en las instancias o clústeres de la base de datos de destino para que el proxy pueda conectarse correctamente.
> - Si elige Crear rol de IAM, es obligatorio el campo Cuentas de base de datos para la autenticación de IAM.
> - Si selecciona un rol de IAM existente, la consola no actualiza automáticamente el rol con los permisos de conexión de base de datos. Compruebe que el rol tenga los permisos `rds-db:connect` necesarios.

Visualización de un proxy

Después de crear uno o varios proxies de RDS, puede verlos y administrarlos en la Consola de administración de AWS, la AWS CLI o la API de RDS. Puede revisar sus detalles de configuración, supervisar el rendimiento y determinar qué proxies modificar o eliminar según sea necesario.

Para permitir que las aplicaciones de base de datos enruten el tráfico a través de un proxy, debe especificar el punto de conexión del proxy en la cadena de conexión.

Consola

Visualización de un proxy en la consola

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Proxies.
3. Seleccione el nombre del proxy para ver sus detalles.
4. En la página de detalles, la sección Grupos de destino muestra cómo se enlaza el proxy a un clúster de bases de datos de Aurora específico. Puede navegar a la página del grupo de destino predeterminado para obtener una vista más detallada de esta asociación, incluida la configuración definida durante la creación del proxy. Esta incluye el porcentaje máximo de conexión, el tiempo de espera de préstamo de la conexión, la familia de motores y los filtros de fijación de sesión.

CLI

Para consultar el proxy mediante la CLI, utilice el comando describe-db-proxies. De forma predeterminada, la solicitud devuelve todos los proxies propiedad de la cuenta de AWS. Para ver los detalles de un solo proxy, especifique su nombre con el parámetro `--db-proxy-name`.

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

Para consultar otra información asociada con el proxy, utilice los comandos que se muestran a continuación.

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

Utilice la siguiente secuencia de comandos para ver más detalles acerca de las cosas que están asociadas con el proxy:

1. Para obtener una lista de proxies, ejecute describe-db-proxies.
2. Para mostrar parámetros de conexión como el porcentaje máximo de conexiones que puede utilizar el proxy, ejecute describe-db-proxy-target-groups `--db-proxy-name`. Utilice el nombre del proxy como el valor del parámetro.
3. Para consultar los detalles del clúster de bases de datos de Aurora con asociación con el grupo de destino devuelto, ejecute describe-db-proxy-targets.

API de RDS

Para ver los proxies mediante la API de RDS, utilice la operación DescribeDBProxies . Devuelve valores del tipo de datos DBProxy.

Para consultar los detalles de la configuración de conexión del proxy, utilice los identificadores de proxy de este valor devuelto con la operación DescribeDBProxyTargetGroups. Devuelve valores del tipo de datos DBProxyTargetGroup.

Para consultar la instancia de RDS o el clúster de bases de datos de Aurora asociado con el proxy, utilice la operación DescribeDBProxyTargets. Devuelve valores del tipo de datos DBProxyTarget.

Conexión a una base de datos mediante RDS Proxy

Se conecta a un clúster de base de datos de Aurora o a un clúster que usa Aurora Serverless v2 a través de un proxy, generalmente de la misma manera que se conecta directamente a la base de datos. La principal diferencia es que se especifica el punto de conexión del proxy en lugar del punto de conexión del clúster. De forma predeterminada, todas las conexiones del proxy tienen capacidad de lectura o escritura y utilizan la instancia de escritor. Si normalmente utiliza el punto de conexión del lector para conexiones de solo lectura, puede crear un punto de conexión de solo lectura adicional para el proxy. Puede usar ese punto de conexión de la misma manera. Para obtener más información, consulte Información general de los puntos de enlace de proxy.

Temas

- Conexión a una base de datos mediante credenciales de base de datos
- Conexión a una base de datos mediante autenticación de IAM
- Consideraciones para conectarse a PostgreSQL

Conexión a una base de datos mediante credenciales de base de datos

Siga los siguientes pasos para conectarse a un proxy mediante credenciales de base de datos:

1. Buscar el punto de enlace del proxy. En la Consola de administración de AWS, puede encontrar el punto de enlace en la página de detalles del proxy correspondiente. Con la AWS CLI, puede usar el comando describe-db-proxies. El siguiente ejemplo muestra cómo.

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
        },
        {
```

```
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. Especifique el punto de conexión como parámetro del host en la cadena de conexión de la aplicación cliente. Por ejemplo, especifique el punto de enlace del proxy como el valor para la opción `mysql -h` o la opción `psql -h`.
3. Proporcione el mismo nombre de usuario y contraseña de la base de datos como suele hacer.

Conexión a una base de datos mediante autenticación de IAM

Cuando utilice la autenticación de IAM con RDS Proxy, tendrá dos opciones de autenticación entre el cliente y el proxy:

- Configure los usuarios de la base de datos para que se autentiquen con nombres de usuario y contraseñas normales. El RDS Proxy recupera el nombre de usuario y las credenciales de contraseña de Secrets Manager. La conexión desde RDS Proxy a la base de datos subyacente no pasa a través de IAM.
- También puede utilizar la autenticación de IAM integral, que se conecta a la base de datos a través del proxy mediante IAM sin necesidad de credenciales de base de datos.

Para conectarse a RDS Proxy utilizando la autenticación de IAM, utilice el mismo procedimiento general de conexión que para la autenticación de IAM con un clúster de bases de datos de Aurora. Para obtener más información acerca del uso de la IAM, consulte Seguridad en Amazon Aurora. Si utiliza la autenticación de IAM integral, proporcione el complemento de autenticación de IAM al usuario de base de datos. Consulte Creación de cuentas de base de datos utilizando autenticación de IAM.

Las principales diferencias en el uso de IAM para RDS Proxy incluyen las siguientes:

- Con la autenticación de IAM estándar, los usuarios de la base de datos disponen de credenciales habituales en la base de datos. Se configuran los secretos de Secrets Manager que contienen

estos nombres de usuario y contraseñas y se autoriza al RDS Proxy para recuperar las credenciales de Secrets Manager. La autenticación IAM se aplica a la conexión entre el programa cliente y el proxy. A continuación, el proxy se autentica en la base de datos utilizando las credenciales de nombre de usuario y contraseña recuperadas de Secrets Manager.

- Con la autenticación de IAM integral, no es necesario configurar los secretos de Secrets Manager para las credenciales de la base de datos. La autenticación de IAM se aplica a la conexión entre el cliente y el proxy y el proxy y la base de datos.
- En lugar del punto de enlace de instancia, clúster o lector, se especifica el punto de enlace del proxy. Para obtener detalles sobre el punto de enlace del proxy, consulte Conexión a al clúster de bases de datos con la autenticación de IAM.
- Asegúrese de que usa Transport Layer Security (TLS)/Capa de sockets seguros (SSL) cuando se conecte a un proxy mediante la autenticación de IAM.

Puede conceder acceso al proxy a un usuario específico modificando la política de IAM. Ejemplo:

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

Tip

Al configurar la autenticación de IAM para las conexiones de RDS Proxy, siga estas instrucciones importantes para evitar problemas de conexión:

- No conceda el rol de `rds_iam` mientras mantenga la autenticación de contraseña general para el mismo usuario o rol de la base de datos.
- Recuerde que, mientras los clientes se conectan a RDS Proxy mediante la autenticación de IAM, RDS Proxy siempre se conecta a la base de datos mediante la autenticación de contraseña a través de Secrets Manager.
- Si la conexión se termina o se vuelve a conectar con frecuencia, elimine cualquier concesión de `rds_iam` existente del usuario o rol y utilice solo la autenticación con contraseña.
- Asegúrese de que la política de contraseñas cumpla con los requisitos de caracteres seguros de SCRAM-SHA-256.

La combinación de métodos de autenticación de IAM y contraseña para el mismo usuario de la base de datos puede provocar inestabilidad de conexión.

Consideraciones para conectarse a PostgreSQL

Si crea un nuevo usuario de base de datos de PostgreSQL para conectarse a RDS Proxy, asegúrese de conceder al usuario el privilegio `CONNECT` en la base de datos. Sin esto, el usuario no puede establecer una conexión. Para obtener más información, consulte the section called “Cómo añadir un nuevo usuario de base de datos a una base de datos PostgreSQL al usar RDS Proxy”.

Cuando un cliente comienza una conexión a una base de datos de PostgreSQL, envía un mensaje de inicio. Este mensaje incluye pares de cadenas de nombres y valores de parámetros. Para obtener detalles, consulte `StartupMessage` en Formatos de mensaje de PostgreSQL en la documentación de PostgreSQL.

Al conectarse a través de un proxy de RDS, el mensaje de inicio puede incluir los siguientes parámetros reconocidos actualmente:

- `user`
- `database`

El mensaje de inicio también puede incluir los siguientes parámetros de tiempo de ejecución adicionales:

- `application_name`
- `client_encoding`
- `DateStyle`
- `TimeZone`
- `extra_float_digits`
- `search_path`

Para obtener más información acerca de la mensajería de PostgreSQL, consulte el Protocolo Frontend/Backend en la documentación de PostgreSQL.

Para PostgreSQL, si usa JDBC, recomendamos lo siguiente para evitar la fijación:

- Establezca el parámetro de conexión JDBC `assumeMinServerVersion` en al menos `9.0` para evitar la fijación. Esto evita que el controlador JDBC realice un viaje de ida y vuelta adicional durante el inicio de la conexión cuando se ejecuta `SET extra_float_digits = 3`.

- Establezca el parámetro de conexión JDBC `ApplicationName` en *`any/your-application-name`* para evitar la fijación. Al hacerlo, se evita que el JDBC driver realice un viaje de ida y vuelta adicional durante el inicio de la conexión cuando se ejecuta `SET application_name = "PostgreSQL JDBC Driver"`. Tenga en cuenta que el parámetro JDBC es `ApplicationName` pero el parámetro `StartupMessage` de PostgreSQL es `application_name`.

Para obtener más información, consulte Cómo evitar la fijación de RDS Proxy. Para obtener más información acerca de la conexión mediante JDBC, consulte Conexión a la base de datos en la documentación de PostgreSQL.

Administración de un RDS Proxy

En esta sección, se proporciona información sobre cómo administrar el funcionamiento y la configuración de RDS Proxy. Estos procedimientos ayudan a su aplicación a hacer más eficiente el uso de las conexiones de base de datos y a lograr la máxima reutilización de la conexión. Cuanto más pueda aprovechar la reutilización de la conexión, más sobrecarga de la CPU y la memoria podrá evitar. Esto, a su vez, reduce la latencia de la aplicación y permite que la base de datos dedique más recursos al procesamiento de solicitudes de la aplicación.

Temas

- Modificación de un RDS Proxy
- Cómo añadir un nuevo usuario de base de datos al usar RDS Proxy
- Paso de la autenticación de IAM estándar a la autenticación de IAM integral para RDS Proxy
- Observaciones sobre la conexión de RDS Proxy
- Cómo evitar la fijación de RDS Proxy
- Eliminación de un RDS Proxy

Modificación de un RDS Proxy

Puede cambiar determinadas configuraciones asociadas a un proxy después de crearlos. Para ello, modifique el propio proxy, su grupo de destino asociado o ambos. Cada proxy tiene un grupo de destino asociado.

Consola de administración de AWS

> **Important**
>
> Los valores de los campos Client authentication type (Tipo de autenticación de cliente) y IAM authentication (Autenticación de IAM) se aplican a todos los secretos de Secrets Manager asociados a este proxy. Para especificar valores diferentes para cada secreto, modifique su proxy mediante la AWS CLI o la API.

Para modificar la configuración de un proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Proxies.
3. En la lista de proxies, elija el proxy cuya configuración desea modificar o vaya a su página de detalles.
4. Para Actions (Acciones), elija Modify (Modificar).
5. Introduzca o elija las propiedades que desea modificar. Puede modificar lo siguiente:

   - Proxy identifier (Identificador de proxy: escriba un nuevo identificador para cambiar el nombre del proxy.
   - Idle client connection timeout (Tiempo de espera de inactividad de conexión de cliente): especifique un período de tiempo de espera de conexión de cliente inactiva.
   - IAM role (Rol de IAM): cambie el rol de IAM utilizado para recuperar los secretos de Secrets Manager.

     > **Note**
     >
     > No puede crear un nuevo rol de IAM si establece el esquema de autenticación predeterminado en la autenticación de IAM.

   - Secrets Manager secrets (Secretos de Secrets Manager): agregue o elimine secretos de Secrets Manager. Estos secretos corresponden a nombres de usuario y contraseñas de la base de datos.
   - Tipo de autenticación de cliente: cambie el tipo de autenticación de las conexiones del cliente al proxy.

- IAM Authentication (Autenticación de IAM): requiera o no permita la autenticación de IAM para las conexiones al proxy.
- Esquema de autenticación predeterminado: cambie el esquema de autenticación predeterminado que utiliza el proxy para las conexiones del cliente al proxy y las conexiones del proxy a la base de datos subyacente.
- Require Transport Layer Security (Requerir Transport Layer Security): active o desactive el requisito de Transport Layer Security (TLS).
- VPC security group (Grupo de seguridad de VPC): agregue o quite grupos de seguridad de VPC para que los utilice el proxy.
- Enable enhanced logging (Habilitar el registro optimizado): habilite o deshabilite el registro mejorado.

6. Elija Modify.

Si no ha encontrado la configuración mostrada que desea cambiar, utilice el procedimiento siguiente para actualizar el grupo de destino del proxy. El grupo de destino asociado con un proxy controla la configuración relacionada con las conexiones de base de datos físicas. Cada proxy tiene un grupo de destino asociado llamado `default`, que se crea automáticamente junto con el proxy. No se puede cambiar el nombre del grupo de destino predeterminado.

Solo puede modificar el grupo de destino desde la página de detalles del proxy, no desde la lista de la página Proxies.

Para modificar la configuración de un grupo de destino de proxy

1. En la página Proxies, vaya a la página de detalles de un proxy.
2. En Target groups (Grupos de destino), elija el enlace `default`. Actualmente, todos los proxies tienen un único grupo de destino denominado `default`.
3. En la página de detalles del grupo de destino default (predeterminado) elija Modify (Modificar).
4. Elija nuevas configuraciones para las propiedades que puede modificar:
   - Base de datos: elija un clúster de Aurora diferente.
   - Connection pool maximum connections (Conexiones máximas de grupo de conexión): ajuste el porcentaje de conexiones disponibles máximas que puede utilizar el proxy.
   - Session pinning filters (Filtros de fijación de sesión): (opcional) elija un filtro de fijación de sesión. De este modo se eluden las medidas de seguridad predeterminadas para multiplexar

las conexiones de bases de datos en las conexiones del cliente. Actualmente, la configuración no es compatible con PostgreSQL. La única opción es `EXCLUDE_VARIABLE_SETS`.

Si se habilita esta configuración, es posible que las variables de sesión de una conexión afecten a otras conexiones. Esto puede provocar errores o problemas de corrección si las consultas dependen de valores de variables de sesión establecidos fuera de la transacción actual. Considere la posibilidad de utilizar esta opción después de comprobar que sea seguro que sus aplicaciones compartan conexiones de bases de datos en las conexiones del cliente.

Los siguientes patrones pueden considerarse seguros:

- Instrucciones `SET` en las que no hay ningún cambio en el valor de la variable de sesión efectiva, es decir, no hay ningún cambio en la variable de sesión.
- Cambia el valor de la variable de sesión y ejecuta una instrucción en la misma transacción.

Para obtener más información, consulte Cómo evitar la fijación de RDS Proxy.

- Connection borrow timeout (Tiempo de espera de préstamo de conexión): ajuste el intervalo de tiempo de espera de préstamo de la conexión. Esta configuración se aplica cuando el número máximo de conexiones ya se está utilizando para el proxy. La configuración determina cuánto tiempo espera el proxy a que una conexión esté disponible antes de devolver un error de tiempo de espera.
- Consutla de inicialización. Añada una consulta de inicialización o modifique la actual (opcional). Puede especificar una o más instrucciones de SQL para que el proxy se ejecute al abrir cada nueva conexión de base de datos. Normalmente, el ajuste se utiliza con instrucciones de `SET` para garantizar que cada conexión tenga una configuración idéntica. Asegúrese de que la consulta que añada sea válida. Para incluir varias variables en una única instrucción de `SET`, utilice separadores de coma. Por ejemplo:

```
SET variable1=value1, variable2=value2
```

Para varias instrucciones, utilice punto y coma como separador.

No puede cambiar ciertas propiedades, como el identificador del grupo de destino y el motor de base de datos.

5. Elija Modify target group (Modificar grupo de destino).

AWS CLI

Para modificar un proxy mediante la AWS CLI, utilice los comandos modify-db-proxy, modify-db-proxy-target-group, deregister-db-proxy-targets y register-db-proxy-targets.

Con el comando `modify-db-proxy`, puede cambiar propiedades como las siguientes:

- El conjunto de secretos de Secrets Manager utilizados por el proxy.
- Si se requiere TLS.
- El tiempo de espera del cliente inactivo.
- Si se debe registrar información adicional de instrucciones de SQL para la depuración.
- El rol de IAM utilizado para recuperar secretos de Secrets Manager.
- Los grupos de seguridad utilizados por el proxy.
- El esquema de autenticación predeterminado asociado al proxy.

En el ejemplo siguiente se muestra cómo cambiar el nombre de un proxy existente.

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

Para modificar la configuración relacionada con la conexión o cambiar el nombre del grupo de destino, utilice el comando `modify-db-proxy-target-group`. Actualmente, todos los proxies tienen un único grupo de destino denominado `default`. Cuando trabaja con este grupo de destino, debe especificar el nombre del proxy y `default` para el nombre del grupo de destino. No se puede cambiar el nombre del grupo de destino predeterminado.

En el ejemplo siguiente se muestra cómo comprobar primero la configuración de `MaxIdleConnectionsPercent` de un proxy y, a continuación, cambiarla mediante el grupo de destino.

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
```

```
                "ConnectionBorrowTimeout": 120,
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

Con los comandos `deregister-db-proxy-targets` y `register-db-proxy-targets`, puede cambiar a qué clústeres de bases de datos de Aurora está asociado el proxy a través de su grupo de destino. Actualmente, cada proxy puede conectarse a un clúster de base de datos de Aurora. El grupo de destino realiza un seguimiento de los detalles de conexión de todas las instancias de base de datos en un clúster de Aurora.

El ejemplo siguiente comienza con un proxy asociado a un clúster de Aurora MySQL denominado `cluster-56-2020-02-25-1399`. El ejemplo muestra cómo cambiar el proxy para que pueda conectarse a un clúster diferente denominado `provisioned-cluster`.

Cuando se trabaja con un clúster de bases de datos de Aurora, se especifica la opción `--db-cluster-identifier`.

El siguiente ejemplo modifica un proxy Aurora MySQL. Un proxy Aurora PostgreSQL tiene el puerto 5432.

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
    ]
}
```

```
aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": []
}

aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

API de RDS

Para modificar un proxy mediante la API de RDS, utilice las operaciones ModifyDBProxy, ModifyDBProxyTargetGroup, DeregisterDBProxyTargets y RegisterDBProxyTargets.

Con `ModifyDBProxy`, puede cambiar propiedades como las siguientes:

- El conjunto de secretos de Secrets Manager utilizados por el proxy.
- Si se requiere TLS.

- El tiempo de espera del cliente inactivo.
- Si se debe registrar información adicional de instrucciones de SQL para la depuración.
- El rol de IAM utilizado para recuperar secretos de Secrets Manager.
- Los grupos de seguridad utilizados por el proxy.

Con `ModifyDBProxyTargetGroup`, puede modificar la configuración relacionada con la conexión. Actualmente, todos los proxies tienen un único grupo de destino denominado `default`. Cuando trabaja con este grupo de destino, debe especificar el nombre del proxy y `default` para el nombre del grupo de destino. No se puede cambiar el nombre del grupo de destino predeterminado.

Con `DeregisterDBProxyTargets` y `RegisterDBProxyTargets`, puede cambiar con qué clúster de Aurora está asociado el proxy a través de su grupo de destino. Actualmente, cada proxy puede conectarse a un clúster de bases de datos de Aurora. El grupo de destino hace un seguimiento de los detalles de conexión de las instancias de base de datos en un clúster de Aurora.

Cómo añadir un nuevo usuario de base de datos al usar RDS Proxy

En algunos casos, podría agregar un nuevo usuario de base de datos a un clúster de Aurora asociado a un proxy. Continúe en función de si utiliza la autenticación estándar con secretos de Secrets Manager o la autenticación de IAM integral.

Si utiliza la autenticación de IAM estándar, siga estas instrucciones:

1. Cree un nuevo secreto de Secrets Manager mediante el procedimiento descrito en Configuración de credenciales de base de datos para RDS Proxy.
2. Actualice el rol de IAM para conceder acceso al RDS Proxy al nuevo secreto de Secrets Manager. Para ello, actualice la sección de recursos de la política del rol de IAM.
3. Modifique el proxy de RDS para añadir el nuevo secreto de Secrets Manager en Secretos de Secrets Manager.
4. Si el nuevo usuario toma el lugar de uno existente, actualice las credenciales almacenadas en el secreto de Secrets Manager del proxy para el usuario existente.

Si utiliza la autenticación de IAM integral, debe crear el usuario de la base de datos y configurar los permisos de IAM. Para hacerlo, realice estos pasos:

1. Cree un nuevo usuario de base de datos en la base de datos que coincida con el nombre de usuario o rol de IAM que desee utilizar para la autenticación.

2. Asegúrese de que el usuario de la base de datos esté configurado con el complemento de autenticación de IAM en la base de datos. Consulte Creación de cuentas de base de datos utilizando autenticación de IAM.
3. Actualice la política de IAM para conceder el permiso `rds-db:connect` al usuario o rol de IAM, tal y como se describe en Creación de una política de IAM para la autenticación de IAM integral.
4. Asegúrese de que el proxy esté configurado para usar la autenticación de IAM como esquema de autenticación predeterminado.

Con la autenticación de IAM integral, no es necesario administrar las credenciales de la base de datos en secretos de Secrets Manager, ya que las credenciales de IAM se utilizan para la autenticación del cliente al proxy y del proxy a la base de datos.

Cómo añadir un nuevo usuario de base de datos a una base de datos PostgreSQL al usar RDS Proxy

Al agregar un nuevo usuario a su base de datos de PostgreSQL, si ha ejecutado el siguiente comando:

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

Otorgue al usuario `rdsproxyadmin` el privilegio `CONNECT` para que pueda supervisar las conexiones en la base de datos de destino.

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

También puede permitir que otros usuarios de la base de datos de destino realicen comprobaciones de estado cambiando `rdsproxyadmin` por el usuario de la base de datos del comando anterior.

Cambio de la contraseña de un usuario de base de datos al usar RDS Proxy

En algunos casos, puede cambiar la contraseña de un usuario de base de datos en un clúster de Aurora asociado a un proxy. Si es así, actualice el secreto de Secrets Manager correspondiente con la nueva contraseña.

Si utiliza la autenticación de IAM integral, no necesita actualizar ninguna contraseña en secretos de Secrets Manager.

Paso de la autenticación de IAM estándar a la autenticación de IAM integral para RDS Proxy

Si actualmente utiliza la autenticación de IAM estándar para RDS Proxy, en la que los clientes se autentican en el proxy mediante IAM pero el proxy se conecta a la base de datos mediante secretos, puede migrar a la autenticación de IAM integral, en la que tanto las conexiones de cliente a proxy como de proxy a base de datos utilizan la autenticación de IAM.

Cambio a autenticación integral de IAM

1. Actualización de los permisos del rol de IAM de RDS Proxy

   Cree una política de permisos de proxy actualizada que incluya Secrets Manager y permisos de `rds:db-connect`:

```
# Create updated proxy permission policy
cat > updated-proxy-policy.json # EOF
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetSecretsValue",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-1234f"
      ]
    },
    {
      "Sid": "RdsDBConnect",
      "Action": [
        "rds-db:connect"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:rds-db:us-east-1:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
jane_doe"
      ]
```

```
    }
  ]
}
```

Actualice la política del rol del proxy:

```
aws iam put-role-policy \
            --role-name RDSProxyRole \
            --policy-name UpdatedProxyPermissions \
            --policy-document file://updated-proxy-policy.json
```

2. Modificación del RDS Proxy para habilitar la autenticación integral de IAM

```
aws rds modify-db-proxy \
  --db-proxy-name my-database-proxy \
  --default-auth-scheme IAM_AUTH \
  --region us-east-1
```

Compruebe que el estado de RDS Proxy sea disponible y `DefaultAuthScheme` sea `IAM_AUTH` antes de continuar para garantizar que no haya ningún tiempo de inactividad durante la migración.

```
aws rds describe-db-proxies --db-proxy-name my-database-proxy --region us-east-1
```

Resultado previsto:

```
{
  "DBProxies": [
    {
      "DBProxyName": "my-database-proxy",
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-0123456789abcdef",
      "Status": "available",
      ...
      "DefaultAuthScheme": "IAM_AUTH"
    }
  ]
}
```

3. Habilitación de la autenticación de IAM en la base de datos

```
aws rds modify-db-cluster \
  --db-cluster-identifier my-database-cluster \
  --enable-iam-database-authentication \
  --region us-east-1
```

4. Configuración del usuario de la base de datos para la autenticación de IAM

   Para Aurora PostgreSQL:

   ```
   GRANT rds_iam TO jane_doe;
   ```

   Para Aurora MySQL:

   ```
   ALTER USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
   ALTER USER 'jane_doe'@'%' REQUIRE SSL;
   ```

5. No es necesario realizar cambios en el código de aplicación del cliente. El proceso de conexión sigue siendo el mismo:

   Para Aurora PostgreSQL:

   ```
   # Generate authentication token
   export PGPASSWORD=$(aws rds generate-db-auth-token \
     --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    \
     --port 5432 \
     --username jane_doe \
     --region us-east-1)

   # Connect to database through proxy
   psql "host=my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    port=5432 user=jane_doe dbname=postgres password=$PGPASSWORD sslmode=require
    sslrootcert=us-east-1-bundle.pem"
   ```

   Para Aurora MySQL:

   ```
   # Generate authentication token
   export MYSQL_PWD=$(aws rds generate-db-auth-token \
     --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    \
   ```

```
  --port 3306 \
  --username jane_doe \
  --region us-east-1)

# Connect to database through proxy
mysql -h my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com \
  -P 3306 \
  -u jane_doe \
  --ssl-ca=us-east-1-bundle.pem \
  --enable-cleartext-plugin
```

Observaciones sobre la conexión de RDS Proxy

Configuración de los valores de conexión

Para ajustar la agrupación de conexiones de RDS Proxy, puede modificar la siguiente configuración:

- IdleClientTimeout
- MaxConnectionsPercent
- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

Puede especificar cuánto tiempo puede estar inactiva una conexión del cliente antes de que el proxy la cierre. El valor predeterminado es 1800 segundos (30 minutos).

Una conexión de cliente se considera inactiva cuando la aplicación no envía una nueva solicitud dentro del plazo especificado después de completar la solicitud anterior. La conexión de base de datos subyacente permanece abierta y se devuelve al grupo de conexiones. Por lo tanto, está disponible para reutilizarla para nuevas conexiones de cliente. Si quiere que el proxy elimine de forma proactiva las conexiones obsoletas, considere la posibilidad de reducir el tiempo de espera de conexión del cliente inactivo. Si la carga de trabajo establece conexiones frecuentes con el proxy, considere la posibilidad de aumentar el tiempo de espera de la conexión del cliente inactivo para ahorrar el costo del establecimiento de conexiones.

Esta configuración se representa mediante el campo Idle client connection timeout (Tiempo de espera de la conexión de cliente inactivo) en la consola de RDS y la configuración

`IdleClientTimeout` en la AWS CLI y la API. Para obtener información sobre cómo cambiar el valor del campo Idle client connection timeout (Tiempo de espera de la conexión de cliente inactivo) en la consola de RDS, consulte Consola de administración de AWS. Para obtener información sobre cómo cambiar el valor de la configuración `IdleClientTimeout`, consulte el comando de la CLI modify-db-proxy o la operación de la API ModifyDBProxy.

MaxConnectionsPercent

Puede limitar el número de conexiones que un proxy de RDS puede establecer con la base de datos de destino. Especifique el límite como porcentaje de las conexiones máximas disponibles para la base de datos. Esta configuración se representa mediante el campo Connection pool maximum connections (Conexiones máximas de grupo de conexión) en la consola de RDS y la configuración `MaxConnectionsPercent` en la AWS CLI y la API.

El valor `MaxConnectionsPercent` se expresa como un porcentaje de la configuración de `max_connections` para el clúster de base de datos de Aurora que usa el grupo de destino. El proxy no crea todas estas conexiones por adelantado. Esta configuración permite que el proxy establezca estas conexiones a medida que la carga de trabajo las necesita.

Por ejemplo, en una base de datos registrada donde `max_connections` está establecido en 1000 y `MaxConnectionsPercent` está establecido en 95, el proxy de RDS establece 950 conexiones como límite máximo para las conexiones simultáneas a esa base de datos de destino.

Un efecto secundario habitual de que la carga de trabajo alcance el número máximo de conexiones a bases de datos permitidas es el aumento de la latencia general de las consultas, junto con un aumento de la métrica `DatabaseConnectionsBorrowLatency`. Puede supervisar las conexiones a bases de datos que se utilizan actualmente y el total permitido comparando las métricas `DatabaseConnections` y `MaxDatabaseConnectionsAllowed`.

Al configurar este parámetro, tenga en cuenta las siguientes prácticas recomendadas:

- Deje suficiente margen de conexión para los cambios en el patrón de carga de trabajo. Se recomienda configurar el parámetro al menos un 30 % por encima del uso supervisado máximo reciente. Dado que el proxy de RDS redistribuye las cuotas de conexión a las bases de datos entre varios nodos, los cambios en la capacidad interna pueden requerir al menos un 30 % de margen para conexiones adicionales para evitar un aumento de la latencia de préstamos.
- El proxy de RDS reserva una cantidad determinada de conexiones para la supervisión activa para facilitar una conmutación por error rápida, el enrutamiento del tráfico y las operaciones internas. La métrica `MaxDatabaseConnectionsAllowed` no incluye estas conexiones reservadas.

Representa el número de conexiones disponibles para atender la carga de trabajo y puede ser inferior al valor derivado de la configuración de `MaxConnectionsPercent`.

Los valores `MaxConnectionsPercent` mínimos recomendados son los siguientes:

- db.t3.small: 100
- db.t3.medium: 55
- db.t3.large: 35
- db.r3.large o superior: 20

Si hay varias instancias de destino registradas en el proxy de RDS, como un clúster de Aurora con nodos lectores, defina el valor mínimo en función de la instancia registrada más pequeña.

Para obtener información sobre cómo cambiar el valor del campo Connection pool maximum connections (Conexiones máximas de grupo de conexión) en la consola de RDS, consulte Consola de administración de AWS. Para obtener información sobre cómo cambiar el valor de la configuración de `MaxConnectionsPercent`, consulte el comando de la CLI modify-db-proxy-target-group o la operación de la API ModifyDBProxyTargetGroup.

> **Important**
>
> Si el clúster de base de datos forma parte de una base de datos global con el reenvío de escritura activado, reduzca el valor `MaxConnectionsPercent` del proxy según la cuota asignada para el reenvío de escritura. La cuota de reenvío de escritura se establece en el parámetro del clúster de base de datos `aurora_fwd_writer_max_connections_pct`. Para obtener información sobre el reenvío de escritura, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

Para obtener información sobre los límites de conexión de base de datos, consulte Número máximo de conexiones a una instancia de base de datos Aurora MySQL y Número máximo de conexiones a una instancia de base de datos de Aurora PostgreSQL.

MaxIdleConnectionsPercent

Puede controlar el número de conexiones de base de datos inactivas que RDS Proxy puede mantener en el grupo de conexiones. De forma predeterminada, RDS Proxy considera que una

conexión de base de datos en su grupo está inactiva cuando no ha habido actividad en la conexión durante cinco minutos.

El valor `MaxIdleConnectionsPercent` se expresa como un porcentaje de la configuración de `max_connections` para el grupo de destino de la instancia de base de datos de RDS. El valor predeterminado es del 50 por ciento de `MaxConnectionsPercent` y el límite superior es el valor de `MaxConnectionsPercent`. Por ejemplo, si `MaxConnectionsPercent` es 80, el valor predeterminado de `MaxIdleConnectionsPercent` es 40.

Con un valor alto, el proxy deja un alto porcentaje de conexiones de base de datos inactivas abiertas. Con un valor bajo, el proxy cierra un alto porcentaje de conexiones de base de datos inactivas. Si sus cargas de trabajo son impredecibles, considere establecer un valor alto para `MaxIdleConnectionsPercent`. De este modo, RDS Proxy puede adaptarse a los aumentos de actividad sin abrir muchas conexiones de base de datos nuevas.

Esta configuración se representa mediante la configuración `MaxIdleConnectionsPercent` de `DBProxyTargetGroup` en la AWS CLI y la API. Para obtener información sobre cómo cambiar el valor de la configuración de `MaxIdleConnectionsPercent`, consulte el comando de la CLI modify-db-proxy-target-group o la operación de la API ModifyDBProxyTargetGroup.

Para obtener información sobre los límites de conexión de base de datos, consulte Número máximo de conexiones a una instancia de base de datos Aurora MySQL y Número máximo de conexiones a una instancia de base de datos de Aurora PostgreSQL.

ConnectionBorrowTimeout

Puede elegir cuánto tiempo espera RDS Proxy a que una conexión a la base de datos del grupo de conexiones esté disponible para su uso antes de devolver un error de tiempo de espera. El valor predeterminado es de 120 segundos. Esta configuración se aplica cuando el número de conexiones está al máximo, por lo que no hay conexiones disponibles en el grupo de conexiones. Esto también se aplica si no hay ninguna instancia de base de datos adecuada disponible para gestionar la solicitud, como cuando se esté realizando una operación de conmutación por error. Con esta configuración, puede establecer el mejor periodo de espera para la aplicación sin cambiar el tiempo de espera de la consulta en el código de la aplicación.

Esta configuración se representa mediante el campo Connection borrow timeout (Tiempo de espera de préstamo de conexión) en la consola de RDS o en la configuración de `ConnectionBorrowTimeout` de `DBProxyTargetGroup` de la AWS CLI o de la API. Para obtener información sobre cómo cambiar el valor del campo Connection borrow timeout

(Tiempo de espera de préstamo de conexión) en la consola de RDS, consulte Consola de administración de AWS. Para obtener información sobre cómo cambiar el valor de la configuración de `ConnectionBorrowTimeout`, consulte el comando de la CLI modify-db-proxy-target-group o la operación de la API ModifyDBProxyTargetGroup.

Conexiones de cliente y base de datos

Las conexiones de la aplicación a RDS Proxy se conocen como conexiones de cliente. Las conexiones desde un proxy a la base de datos son conexiones de base de datos. Cuando se utiliza RDS Proxy, las conexiones de cliente terminan en el proxy, mientras que las conexiones de la base de datos se administran en RDS Proxy.

La agrupación de conexiones en la aplicación puede ofrecer la ventaja de reducir el establecimiento de conexiones recurrentes entre la aplicación y RDS Proxy.

Tenga en cuenta los siguientes elementos de configuración antes de implementar un grupo de conexiones en la aplicación:

- Duración máxima de la conexión del cliente: RDS Proxy impone una vida útil máxima de 24 horas para las conexiones del cliente. Este valor no se puede configurar. Configure su grupo con una vida útil máxima de conexión inferior a 24 horas para evitar caídas inesperadas en la conexión del cliente.
- Tiempo de espera de inactividad en la conexión de cliente: RDS Proxy impone un tiempo máximo de inactividad para las conexiones del cliente. Configure su grupo con un valor de tiempo de espera de conexión inactiva inferior al tiempo de espera de la conexión de cliente para RDS Proxy, a fin de evitar caídas inesperadas en la conexión.

El número máximo de conexiones del cliente configuradas en el grupo de conexiones de la aplicación no tiene por qué limitarse a la configuración max_connections de RDS Proxy

La agrupación de conexiones de clientes prolonga la vida útil de las conexiones del cliente. Si las conexiones se bloquean, agrupar las conexiones de cliente podría reducir la eficiencia de la multiplexación. Las conexiones del cliente que están bloqueadas pero inactivas en el grupo de conexiones de la aplicación siguen teniendo una conexión de base de datos e impiden que otras conexiones del cliente reutilicen la conexión a la base de datos. Revise los registros de proxy para comprobar si las conexiones se fijan.

> **Note**
>
> RDS Proxy cierra las conexiones de base de datos un poco después de 24 horas, cuando ya no están en uso. El proxy realiza esta acción independientemente del valor de configuración máxima de conexiones inactivas.

Cómo evitar la fijación de RDS Proxy

La multiplexación es más eficiente cuando las solicitudes de base de datos no dependen de la información de estado de solicitudes anteriores. En ese caso, RDS Proxy puede reutilizar una conexión en la conclusión de cada transacción. Algunos ejemplos de dicha información de estado incluyen la mayoría de las variables y parámetros de configuración que puede cambiar a través de instrucciones `SET` o `SELECT`. Las transacciones SQL en una conexión de cliente pueden multiplexar entre conexiones de base de datos subyacentes de forma predeterminada.

Las conexiones al proxy pueden entrar en un estado conocido como fijación. Cuando se fija una conexión, cada transacción posterior utiliza la misma conexión de base de datos subyacente hasta que finaliza la sesión. Otras conexiones de cliente tampoco pueden reutilizar esa conexión de base de datos hasta que finaliza la sesión. La sesión finaliza cuando se interrumpe la conexión del cliente.

RDS Proxy fija automáticamente una conexión de cliente a una conexión de base de datos específica cuando detecta un cambio de estado de sesión que no es apropiado para otras sesiones. La fijación reduce la eficacia de la reutilización de la conexión. Si todas o casi todas las conexiones experimentan asignación, plantéese modificar el código de la aplicación o la carga de trabajo para reducir las condiciones que provocan la fijación.

Por ejemplo, su aplicación cambia una variable de sesión o un parámetro de configuración. En este caso, las instrucciones posteriores pueden depender de que la nueva variable o parámetro esté en vigor. Por lo tanto, cuando RDS Proxy procesa solicitudes para cambiar las variables de sesión o los parámetros de configuración, fija esa sesión a la conexión de base de datos. De esta forma, el estado de la sesión permanece en vigor para todas las transacciones posteriores en la misma sesión.

Para algunos motores de base de datos, esta regla no se aplica a todos los parámetros que se pueden establecer. RDS Proxy realiza un seguimiento de ciertas sentencias y variables. Por lo tanto, RDS Proxy no fija la sesión cuando las modifica. En ese caso, RDS Proxy solo reutiliza la conexión para otras sesiones que tengan los mismos valores para esa configuración. Para obtener las listas de

instrucciones y variables rastreadas para Aurora MySQL, consulte Qué seguimiento hace RDS Proxy para bases de datos de Aurora MySQL.

Qué seguimiento hace RDS Proxy para bases de datos de Aurora MySQL

A continuación se presentan las instrucciones MySQL de las que RDS Proxy realiza un seguimiento:

- DROP DATABASE
- DROP SCHEMA
- USE

A continuación se presentan las variables MySQL de las que RDS Proxy realiza un seguimiento:

- `AUTOCOMMIT`
- `AUTO_INCREMENT_INCREMENT`
- `CHARACTER SET (or CHAR SET)`
- `CHARACTER_SET_CLIENT`
- `CHARACTER_SET_DATABASE`
- `CHARACTER_SET_FILESYSTEM`
- `CHARACTER_SET_CONNECTION`
- `CHARACTER_SET_RESULTS`
- `CHARACTER_SET_SERVER`
- `COLLATION_CONNECTION`
- `COLLATION_DATABASE`
- `COLLATION_SERVER`
- `INTERACTIVE_TIMEOUT`
- `NAMES`
- `NET_WRITE_TIMEOUT`
- `QUERY_CACHE_TYPE`
- `SESSION_TRACK_SCHEMA`
- `SQL_MODE`
- `TIME_ZONE`

- `TRANSACTION_ISOLATION (or TX_ISOLATION)`
- `TRANSACTION_READ_ONLY (or TX_READ_ONLY)`
- `WAIT_TIMEOUT`

Minimizar la fijación

El ajuste del rendimiento para RDS Proxy implica intentar maximizar la reutilización de la conexión en el nivel de transacción (multiplexación) minimizando la fijación.

Puede minimizar la fijación, puede realizar lo siguiente:

- Evite las solicitudes de base de datos innecesarias que puedan provocar la fijación.
- Establezca variables y parámetros de configuración de forma coherente en todas las conexiones. De esta forma, es más probable que las sesiones posteriores reutilicen conexiones que tengan esa configuración particular.

  Sin embargo, en el caso de PostgreSQL, el establecimiento de una variable conduce a la fijación de sesión.
- Para una base de datos de familia de motores MySQL, aplique un filtro de sesión de fijación al proxy. Puede eximir a ciertos tipos de operaciones de asignar la sesión si sabe que hacerlo no afecta al correcto funcionamiento de la fijación.
- Consulte la frecuencia con la que se produce la fijación mediante la supervisión de la métrica de Amazon CloudWatch `DatabaseConnectionsCurrentlySessionPinned`. Para obtener información sobre esta y otras métricas de CloudWatch, consulte Supervisión de las métricas de RDS Proxy con Amazon CloudWatch.
- Si utiliza instrucciones `SET` para realizar una inicialización idéntica para cada conexión de cliente, puede hacerlo sin dejar de mantener la multiplexación en el nivel de transacción. En este caso, mueva las instrucciones que configuran el estado de la sesión inicial a la consulta de inicialización utilizada por un proxy. Esta propiedad es una cadena que contiene una o varias instrucciones SQL, separadas por punto y coma.

  Por ejemplo, puede definir una consulta de inicialización para un proxy que establezca determinados parámetros de configuración. A continuación, RDS Proxy aplica esa configuración cada vez que configura una nueva conexión para ese proxy. Puede eliminar las instrucciones `SET` correspondientes de su código de aplicación, para que no interfieran con la multiplexación en el nivel de transacción.

Para obtener métricas acerca de la frecuencia con la que se produce la fijación de un proxy, consulte Supervisión de las métricas de RDS Proxy con Amazon CloudWatch.

Condiciones que provocan la fijación de todas las familias de motores

El proxy fija la sesión a la conexión actual en las siguientes situaciones en las que la multiplexación podría provocar un comportamiento inesperado:

- Cualquier instrucción con un tamaño de texto superior a 16 KB hace que el proxy fije la sesión.

Condiciones que provocan la fijación para Aurora MySQL

Para PostgreSQL, las siguientes interacciones también producen fijación:

- Las instrucciones de bloqueo de tabla explícitas `LOCK TABLE`, `LOCK TABLES` o `FLUSH TABLES WITH READ LOCK` hacen que el proxy fije la sesión.
- Creación de bloqueos con nombre mediante`GET_LOCK`provoca que el proxy instale la sesión.
- El establecimiento de una variable de usuario o una variable de sistema (con algunas excepciones) hace que el proxy fije la sesión. Si esta situación reduce demasiado la reutilización de la conexión, puede elegir que las operaciones `SET` no provoquen la fijación. Para obtener información acerca de cómo hacerlo estableciendo la propiedad de los filtros de fijación de sesión, consulte Creación de un proxy para Amazon Aurora y Modificación de un RDS Proxy.
- La creación de una tabla temporal hace que el proxy fije la sesión. De esta forma, el contenido de la tabla temporal se conserva a lo largo de la sesión con independencia de los límites de la transacción.
- La llamada a las funciones `ROW_COUNT`, `FOUND_ROWS` y `LAST_INSERT_ID` a veces provoca fijación.

  Es posible que las circunstancias exactas en las que estas funciones provocan fijación difieran entre versiones de Aurora MySQL que son compatibles con MySQL 5.7.
- Las instrucciones preparadas hacen que el proxy fije la sesión. Esta regla se aplica si la instrucción preparada utiliza texto SQL o el protocolo binario.
- El proxy de RDS no fija las conexiones cuando se utiliza SET LOCAL.
- La llamada a procedimientos almacenados y funciones almacenadas no causa fijación. El proxy de RDS no detecta ningún cambio de estado de sesión resultante de dichas llamadas. Asegúrese

de que su aplicación no cambie el estado de la sesión dentro de las rutinas almacenadas si confía en que ese estado de sesión vaya a persistir en las transacciones. Por ejemplo, RDS Proxy no es compatible actualmente con un procedimiento almacenado que crea una tabla temporal que persiste a través de todas las transacciones.

Si tiene conocimientos especializados sobre el comportamiento de la aplicación, puede omitir el comportamiento de fijación de determinadas instrucciones de aplicación. Para ello, elija la opción Session pinning filters (Filtro de fijación de sesión) al crear el proxy. Actualmente, puede desactivar la fijación de sesión para establecer variables de sesión y valores de configuración.

Condiciones que provocan la fijación para Aurora PostgreSQL

Para PostgreSQL, las siguientes interacciones también producen fijación:

- Uso de comandos `SET`.
- Uso de los comandos `PREPARE`, `DISCARD`, `DEALLOCATE` o `EXECUTE` para gestionar las instrucciones preparadas.
- Creación de secuencias, tablas o vistas temporales.
- Declaración de cursores.
- Descartar el estado de la sesión.
- Escucha en un canal de notificación.
- Carga de un módulo de biblioteca como `auto_explain`.
- Manipulación de secuencias mediante el uso de funciones como `nextval` y `setval`.
- Interacción con bloqueos mediante el uso de funciones como `pg_advisory_lock` y `pg_try_advisory_lock`.

> **Note**
>
> RDS Proxy no fija los bloqueos consultivos en la transacción, específicamente `pg_advisory_xact_lock`, `pg_advisory_xact_lock_shared`, `pg_try_advisory_xact_lock` y `pg_try_advisory_xact_lock_shared`.

- Cómo configurar un parámetro o restablecerlo a su valor predeterminado. En concreto, el uso de comandos `SET` y `set_config` para asignar valores predeterminados a las variables de sesión.
- La llamada a procedimientos almacenados y funciones almacenadas no causa fijación. El proxy de RDS no detecta ningún cambio de estado de sesión resultante de dichas llamadas. Asegúrese

de que su aplicación no cambie el estado de la sesión dentro de las rutinas almacenadas si confía en que ese estado de sesión vaya a persistir en las transacciones. Por ejemplo, RDS Proxy no es compatible actualmente con un procedimiento almacenado que crea una tabla temporal que persiste a través de todas las transacciones.

Eliminación de un RDS Proxy

Puede eliminar un proxy cuando ya no lo necesite. O podría eliminar un proxy si pone la instancia de base de datos o el clúster asociado con él fuera de servicio.

Consola de administración de AWS

Para eliminar un proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Proxies.
3. Elija el proxy que desea eliminar de la lista.
4. Elija Delete Proxy (Eliminar proxy).

AWS CLI

Para eliminar un proxy de base de datos, utilice el comando de la AWS CLI delete-db-proxy. Para quitar asociaciones relacionadas, utilice también el comando deregister-db-proxy-targets.

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
    [--db-cluster-identifiers cluster_id]
```

API de RDS

Para eliminar un proxy de base de datos, llame a la función de la API de Amazon RDS DeleteDBProxy. Para eliminar elementos relacionados y asociaciones, llame también a las funciones DeleteDBProxyTargetGroup y DeregisterDBProxyTargets.

Trabajo con puntos de enlace del proxy de Amazon RDS

Los puntos de conexión de RDS Proxy proporcionan formas flexibles y eficientes de administrar las conexiones de bases de datos, lo que mejora la escalabilidad, la disponibilidad y la seguridad. Con los puntos de conexión del proxy, puede:

- Simplificar la supervisión y la resolución de problemas: use varios puntos de conexión para realizar un seguimiento y administrar conexiones de diferentes aplicaciones de forma independiente.
- Mejorar la escalabilidad de lectura: aproveche los puntos de conexión de lector con los clústeres de bases de datos de Aurora para distribuir el tráfico de lectura de manera eficiente, lo que optimiza el rendimiento para las cargas de trabajo con muchas consultas.
- Habilitar el acceso entre VPC: conecte bases de datos entre VPC mediante puntos de conexión entre VPC, lo que permite que recursos como las instancias de Amazon EC2 de una VPC accedan a las bases de datos de otra.

Temas

- Información general de los puntos de enlace de proxy
- Limitaciones para los puntos de conexión de proxy
- Usar puntos de enlace del lector con los clústeres de Aurora
- Acceso a las bases de datos de Aurora en todas las VPC
- Creación de un punto de enlace de proxy
- Visualización de puntos de enlace de proxy
- Modificación de un punto de enlace de proxy
- Eliminación de un punto de enlace de proxy

Información general de los puntos de enlace de proxy

Trabajar con puntos de conexión de RDS Proxy implica los mismos tipos de procedimientos que con los puntos de conexión del lector y clústeres de base de datos de Aurora. Si no está familiarizado

con los puntos de enlace de Aurora, puede encontrar más información en Conexiones de puntos de conexión de Amazon Aurora.

Cuando utiliza RDS Proxy con un clúster de Aurora, el punto de conexión predeterminado admite operaciones de lectura y escritura. Esto significa que enruta todas las solicitudes a la instancia de escritura del clúster, lo que contribuye a su límite de `max_connections`. Para distribuir mejor el tráfico, puede crear puntos de conexión adicionales de lectura/escritura o de solo lectura para el proxy, lo que permite una administración de carga de trabajo más eficiente y una mejor escalabilidad.

Utilice un punto de conexión de solo lectura con el proxy para gestionar consultas de lectura, de igual manera que lo haría con un punto de conexión de lector para un clúster aprovisionado de Aurora. Este enfoque maximiza la escalabilidad de lectura de Aurora mediante la distribución de consultas entre una o más instancias de base de datos de lector. Al utilizar un punto de conexión de solo lectura y agregar más instancias de lector según sea necesario, puede aumentar el número de consultas y conexiones simultáneas que el clúster puede gestionar.

> **Tip**
>
> Cuando crea un proxy para un clúster de Aurora mediante la Consola de administración de AWS, puede tener el proxy de RDS para crear un punto de conexión del lector automáticamente. Para obtener información acerca de los beneficios de un punto de enlace del lector, consulte Usar puntos de enlace del lector con los clústeres de Aurora.

Cuando cree un punto de conexión del proxy, puede asociarlo con una nube privada virtual (VPC) diferente de la que utiliza la VPC del proxy. Esto le permite conectarse al proxy desde otra VPC, como una utilizada por una aplicación diferente dentro de la organización.

Para obtener información acerca de los límites asociados a los puntos de enlace de proxy, consulte Limitaciones para los puntos de conexión de proxy.

RDS Proxy registra el prefijo de cada entrada con el nombre del punto de conexión del proxy asociado. Este puede ser el nombre especificado para un punto de conexión definido por el usuario o el nombre `default` especial para el punto de conexión de lectura o escritura predeterminado de un proxy.

Cada punto de conexión de proxy tiene su propio conjunto de métricas de CloudWatch. Supervise las métricas de todos los puntos de conexión de proxy, un punto de conexión específico o de todos

los puntos de conexión de lectura o escritura o de solo lectura de un proxy. Para obtener más información, consulte Supervisión de las métricas de RDS Proxy con Amazon CloudWatch.

Un punto de enlace del proxy utiliza el mismo mecanismo de autenticación que su proxy asociado. El proxy de RDS configura automáticamente permisos y autorizaciones para el punto de enlace definido por el usuario, de acuerdo con las propiedades del proxy asociado.

Para obtener información sobre cómo funcionan los puntos de conexión proxy para los clústeres de bases de datos de una base de datos global de Aurora, consulte Cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales.

Limitaciones para los puntos de conexión de proxy

Los puntos de conexión de RDS Proxy tienen las siguientes limitaciones:

- El punto de conexión predeterminado del proxy de RDS no se puede modificar.
- El número máximo de puntos de enlace definidos por el usuario para un proxy es 20. Por lo tanto, un proxy puede tener hasta 21 puntos de enlace: el punto de enlace predeterminado, más 20 que cree.
- Cuando asocia puntos de enlace adicionales con un proxy, RDS Proxy determina automáticamente qué instancias de base de datos del clúster se utilizarán para cada punto de enlace. No puede elegir instancias específicas tal y como lo hace con los puntos de conexión personalizados de Aurora.
- Para los tipos de redes de punto de conexión de IPv6 o de doble pila, la VPC y las subredes deben estar configuradas para admitir el tipo de red seleccionado.

Al crear un proxy, RDS crea automáticamente un punto de conexión de VPC para una comunicación segura entre las aplicaciones y la base de datos. El punto de conexión de VPC es visible y se puede acceder a él desde la consola de Amazon VPC.

Al agregar un nuevo punto de conexión de proxy, se aprovisiona un punto de conexión de interfaz de AWS PrivateLink. Si agrega uno o más puntos de conexión al proxy, incurrirá en cargos adicionales. Para obtener más información, consulte Precios de proxy de RDS.

Usar puntos de enlace del lector con los clústeres de Aurora

Puede crear y conectarse a puntos de enlace de solo lectura denominados puntos de enlace del lector cuando usa RDS Proxy con los clústeres de Aurora. Estos puntos de enlace del lector

ayudan a mejorar la escalabilidad de lectura de sus aplicaciones que requieren un uso intensivo de consultas. Los puntos de enlace del lector también ayudan a mejorar la disponibilidad de las conexiones si una instancia de base de datos de lector del clúster deja de estar disponible.

> **Note**
>
> Cuando especifica que un punto de enlace nuevo es de solo lectura, RDS Proxy requiere que el clúster de Aurora tenga una o más instancias de base de datos de lector. En algunos casos, puede cambiar el destino del proxy a un clúster de Aurora que contenga a un solo escritor. Si lo hace, se producirá un error en cualquier solicitud al punto de conexión del lector. Las solicitudes también fallan si el destino del proxy es una instancia de RDS en lugar de un clúster de Aurora.
>
> Si un clúster de Aurora tiene instancias de lector, pero esas instancias no están disponibles, RDS Proxy espera para enviar la solicitud en lugar de devolver un error inmediatamente. Si no hay instancias de lector disponibles dentro del periodo de tiempo de espera de préstamo de conexión, la solicitud fallará.

Cómo los puntos de enlace del lector ayudan a la disponibilidad de las aplicaciones

En algunos casos, es posible que una o más instancias de lector del clúster no estén disponibles. Si es así, las conexiones que utilizan un punto de enlace del lector de un proxy de base de datos se pueden recuperar más rápidamente que las que utilizan el punto de enlace del lector de Aurora. El proxy de RDS enruta las conexiones solo a las instancias de lector disponibles en el clúster. No hay retraso debido al almacenamiento en caché de DNS cuando una instancia deja de estar disponible.

Si la conexión es multiplexada, RDS Proxy dirige las consultas posteriores a una instancia de base de datos de lector diferente sin interrupciones en su aplicación. Durante el cambio automático a una instancia de lector nueva, el proxy de RDS verifica el retraso de reproducción de las instancias de lector antiguas y nuevas. El proxy de RDS se asegura de que la instancia de lector nueva esté actualizada con los mismos cambios que la instancia de lector anterior. De esta manera, su aplicación nunca ve datos obsoletos cuando RDS Proxy cambia de una instancia de base de datos de lector a otra.

Si la conexión está anclada, la siguiente consulta en la conexión devuelve un error. Sin embargo, la aplicación puede volver a conectarse inmediatamente al mismo punto de enlace. El proxy de RDS enruta la conexión a una instancia de base de datos de lector diferente que se encuentra en estado

`available`. Cuando se vuelve a conectar manualmente, RDS Proxy no comprueba el retraso de reproducción entre las instancias de lector antiguas y nuevas.

Si su clúster de Aurora no tiene instancias de lector disponibles, RDS Proxy comprueba si esta condición es temporal o permanente. El comportamiento en cada caso es el siguiente:

- Supongamos que su clúster tiene una o más instancias de base de datos de lector, pero ninguna de ellas está en el estado `Available`. Por ejemplo, todas las instancias de lector pueden estar reiniciándose o encontrando problemas. En ese caso, los intentos de conectarse a un punto de enlace del lector esperan a que una instancia de lector esté disponible. Si no hay instancias de lector disponibles dentro del periodo de tiempo de espera de préstamo de conexión, se produce un error en el intento de conexión. Si una instancia de lector está disponible, el intento de conexión se lleva a cabo correctamente.
- Supongamos que su clúster no tiene instancias de base de datos de lector. En ese caso, RDS Proxy devuelve un error inmediatamente si intenta conectarse a un punto de enlace del lector. Para resolver este problema, agregue una o más instancias de lector al clúster antes de conectarse al punto de enlace del lector.

Cómo los puntos de enlace del lector ayudan a la escalabilidad de las consultas

Los puntos de enlace del lector de un proxy ayudan con la escalabilidad de consulta de Aurora de las siguientes maneras:

- A medida que agrega instancias de lector a su clúster de Aurora, RDS Proxy puede enrutar conexiones nuevas a cualquier punto de enlace del lector a las diferentes instancias de lector. De esta forma, las consultas realizadas con una conexión de punto de enlace de lector no ralentizan las consultas realizadas con otra conexión de punto de enlace del lector. Las consultas se ejecutan en instancias de base de datos independientes. Cada instancia de base de datos tiene sus propios recursos informáticos, caché de búfer y demás.
- Cuando sea práctico, RDS Proxy utiliza la misma instancia de base de datos de lector para todos los problemas de consultas mediante una conexión de punto de enlace del lector en particular. De esta manera, un conjunto de consultas relacionadas en las mismas tablas puede aprovechar el almacenamiento en caché, la optimización del plan y demás, en una instancia de base de datos particular.
- Si una instancia de base de datos de lector deja de estar disponible, el efecto en la aplicación depende de si la sesión está multiplexada o anclada. Si la sesión está multiplexada, RDS Proxy enruta las consultas posteriores a una instancia de base de datos de lector diferente sin acciones

por su parte. Si la sesión está anclada, la aplicación recibe un error y debe volver a conectarse. Puede volver a conectarse al punto de enlace del lector inmediatamente y RDS Proxy enruta la conexión a una instancia de base de datos de lector disponible. Para obtener más información acerca de la multiplexación y el anclaje de sesiones de proxy, consulte Información general de los conceptos de RDS Proxy.

- Cuantas más instancias de base de datos de lector tenga en el clúster, más conexiones simultáneas podrá realizar con los puntos de conexión de lector. Por ejemplo, supongamos que el clúster tiene cuatro instancias de base de datos de lector, cada una configurada para permitir 200 conexiones simultáneas. Supongamos que su proxy está configurado para usar el 50 % de las conexiones máximas. Aquí, el número máximo de conexiones que puede realizar a través de los puntos de enlace del lector en el proxy es 100 (50 % de 200) para el lector 1. También son 100 para el lector 2, y así sucesivamente, para un total de 400. Si se duplica el número de instancias de la base de datos del lector del clúster a ocho, el número máximo de conexiones a través de los puntos de conexión del lector también se duplicará (hasta 800).

Ejemplos de uso de puntos de enlace del lector

El siguiente ejemplo de Linux muestra cómo puede confirmar que está conectado a un clúster de Aurora MySQL a través de un punto de enlace del lector. La configuración de `innodb_read_only` está habilitada. Los intentos de realizar operaciones de escritura como las instrucciones de `CREATE DATABASE` fallan. Además, puede confirmar que está conectado a una instancia de base de datos de lector mediante la verificación del nombre de la instancia de base de datos con el uso de la variable de `aurora_server_id`.

> **Tip**
>
> No confíe solo en comprobar el nombre de la instancia de base de datos para determinar si la conexión es de lectura o escritura o de solo lectura. Recuerde que las instancias de base de datos en un clúster de Aurora puede cambiar los roles entre escritor y lector cuando se producen conmutaciones por error.

```
$ mysql -h endpoint-demo-reader.endpoint.proxy-demo.us-east-1.rds.amazonaws.com -u
 admin -p
...
mysql> select @@innodb_read_only;
+--------------------+
```

```
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
mysql> create database shouldnt_work;
ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it
 cannot execute this statement

mysql> select @@aurora_server_id;
+---------------------------------------+
| @@aurora_server_id                    |
+---------------------------------------+
| proxy-reader-endpoint-demo-instance-3 |
+---------------------------------------+
```

El siguiente ejemplo muestra cómo la conexión a un punto de enlace del lector de proxy puede seguir funcionando incluso cuando se elimina la instancia de base de datos del lector. En este ejemplo, el clúster de Aurora tiene dos instancias de lector, `instance-5507` y `instance-7448`. La conexión con el punto de enlace del lector comienza mediante una de las instancias de lector. Durante el ejemplo, esta instancia de lector se elimina mediante un comando `delete-db-instance`. Para consultas posteriores, el proxy de RDS cambia a una instancia de lector diferente.

```
$ mysql -h reader-demo.endpoint.proxy-demo.us-east-1.rds.amazonaws.com
  -u my_user -p
...
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-5507      |
+--------------------+

mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+

mysql> select count(*) from information_schema.tables;
+----------+
| count(*) |
```

```
+----------+
|      328 |
+----------+
```

Mientras que la sesión de `mysql` sigue funcionando, el siguiente comando elimina la instancia de lector a la que está conectado el punto de enlace del lector.

```
aws rds delete-db-instance --db-instance-identifier instance-5507 --skip-final-snapshot
```

Las consultas en la sesión de `mysql` sigue trabajando sin necesidad de volver a conectarse. El proxy de RDS cambia a una instancia de base de datos de lector diferente automáticamente.

```
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-7448      |
+--------------------+

mysql> select count(*) from information_schema.TABLES;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Acceso a las bases de datos de Aurora en todas las VPC

De forma predeterminada, los componentes de su pila de tecnología de Aurora están todos en la misma Amazon VPC. Por ejemplo, supongamos que una aplicación que se ejecuta en una instancia de Amazon EC2 se conecta a un clúster de bases de datos de Aurora. En este caso, el servidor de la aplicación y la base de datos deben estar dentro de la misma VPC.

Con RDS Proxy, puede configurar el acceso a un clúster de bases de datos de Aurora en una VPC a partir de recursos de otra VPC, como las instancias de EC2. Por ejemplo, la organización puede tener varias aplicaciones que tengan acceso a los mismos recursos de base de datos. Cada aplicación puede estar en su propia VPC.

A fin de habilitar el acceso entre VPC, cree un nuevo punto de enlace para el proxy. El propio proxy reside en la misma VPC que el clúster de bases de datos de Aurora. Sin embargo, el punto de enlace

en VPC reside en la otra VPC, junto con otros recursos, como las instancias EC2. El punto de enlace en VPC está asociado con subredes y grupos de seguridad de la misma VPC que EC2 y otros recursos. Estas asociaciones permiten conectarse al punto de enlace desde las aplicaciones que, de lo contrario, no pueden acceder a la base de datos debido a las restricciones de la VPC.

Los siguientes pasos explican cómo crear y acceder a un punto de enlace en VPC a través de RDS Proxy:

1. Cree dos VPC o elija dos VPC que ya utilice para el trabajo en Aurora . Cada VPC debe tener sus propios recursos de red asociados, como una puerta de enlace de Internet, tablas de enrutamiento, subredes y grupos de seguridad. Si solo tiene una VPC, puede consultar Introducción a Amazon Aurora para conocer los pasos a fin de configurar otra VPC para que use Aurora con éxito. También puede examinar la VPC existente en la consola de Amazon EC2 para ver los tipos de recursos que conectar entre sí.
2. Cree un proxy de base de datos asociado con el clúster de bases de datos de Aurora al que desea conectarse. Siga el procedimiento indicado en Creación de un proxy para Amazon Aurora.
3. En la página de Details (Detalles) para su proxy en la consola de RDS, en la pestaña de Proxy endpoints (Puntos de enlace de proxy), elija Create endpoint (Crear punto de enlace). Siga el procedimiento indicado en Creación de un punto de enlace de proxy.
4. Elija si desea que el punto de enlace en VPC sea de lectura y escritura o de solo lectura.
5. En lugar de aceptar el valor predeterminado de la misma VPC que el clúster de bases de datos de Aurora, elija una VPC diferente. Esta VPC debe estar en la misma región de AWS que la VPC donde reside el proxy.
6. Ahora, en lugar de aceptar los valores predeterminados para subredes y grupos de seguridad de la misma VPC que el clúster de bases de datos de Aurora, haga nuevas selecciones. Estos se basen en las subredes y los grupos de seguridad de la VPC que eligió.
7. No es necesario cambiar las opciones de configuración de los secretos de Secrets Manager. Las mismas credenciales funcionan para todos los puntos de enlace de proxy, independientemente de la VPC en la que se encuentre cada punto de enlace. Del mismo modo, al utilizar la autenticación de IAM, la configuración y los permisos de IAM funcionan de manera uniforme en todos los puntos de conexión del proxy, incluso cuando los puntos de conexión están en diferentes VPC. No se requiere ninguna configuración de IAM adicional por punto de conexión.
8. Espere a que el punto de enlace nuevo alcance el estado de Available (Disponible).
9. Anote el nombre completo del punto de enlace. Este es el valor que termina en *`Region_name`*`.rds.amazonaws.com` que proporciona como parte de la cadena de conexión para la aplicación de base de datos.

10 Acceda al punto de enlace nuevo desde un recurso en la misma VPC que el punto de enlace. Una forma sencilla de probar este proceso es crear una instancia de EC2 nueva en esta VPC. A continuación, puede iniciar sesión en la instancia de EC2 y ejecutar los comandos `mysql` o `psql` para conectarse mediante el valor de punto de conexión en la cadena de conexión.

Creación de un punto de enlace de proxy

Para crear un punto de conexión proxy, siga estas instrucciones:

Consola

Para crear un punto de enlace de proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Proxies.
3. Haga clic en el nombre del proxy para el que desea crear un punto de enlace nuevo.

   Aparecerá la página de detalles de ese proxy.
4. En la sección de Proxy endpoints (Puntos de enlace de proxy), elija Crear endpoint proxy (Crear punto de enlace de proxy).

   Aparecerá la ventana de Create proxy endpoint (Crear punto de enlace de proxy).
5. Para Proxy endpoint name (Nombre del punto de enlace de proxy), escriba un nombre descriptivo de su elección.
6. Para Target role (Rol de destino), elija si desea que el punto de enlace sea de lectura o escritura o de solo lectura.

   Las conexiones que utilizan puntos de conexión de lectura/escritura pueden realizar cualquier tipo de operación, como instrucciones de lenguaje de definición de datos (DDL), instrucciones de lenguaje de manipulación de datos (DML) y consultas. Estos puntos de conexión siempre se conectan a la instancia principal del clúster de Aurora. Puede utilizar puntos de enlace de lectura o escritura para operaciones generales de bases de datos cuando solo utiliza un punto de enlace único en la aplicación. También puede utilizar puntos de enlace de lectura o escritura para operaciones administrativas, aplicaciones de procesamiento de transacciones en línea (OLTP) y trabajos de extracción, transformación y carga (ETL).

Las conexiones que utilizan un punto de enlace de solo lectura solo pueden realizar consultas. Cuando hay varias instancias de lector en el clúster de Aurora, RDS Proxy puede utilizar una instancia de lector diferente para cada conexión al punto de conexión. De esta manera, una aplicación que requiere un uso intensivo de consultas puede aprovechar la capacidad de clúster de Aurora. Puede agregar más capacidad de consulta al clúster con la adición de más instancias de base de datos de lector. Estas conexiones de solo lectura no imponen sobrecargas en la instancia principal del clúster. De esta manera, sus consultas de informes y análisis no ralentizan las operaciones de escritura de sus aplicaciones de OLTP.

7. En Nube privada virtual (VPC), elija el valor predeterminado para acceder al punto de conexión desde las mismas instancias de EC2 u otros recursos que normalmente utiliza para acceder al proxy o a su base de datos asociada. Para configurar el acceso entre VPC para este proxy, elija una VPC distinta de la predeterminada. Para obtener más información sobre el acceso a través de VPC, consulte Acceso a las bases de datos de Aurora en todas las VPC.
8. Para Tipo de red de punto de conexión, elija la versión de IP del punto de conexión de proxy. Las opciones disponibles son:

   - IPv4: el punto de conexión del proxy utiliza solo direcciones IPv4 (predeterminado).
   - IPv6: el punto de conexión del proxy utiliza solo direcciones IPv6.
   - Doble pila: el punto de conexión del proxy admite direcciones IPv4 y IPv6.

   Para usar IPv6 o de doble pila, la VPC y las subredes deben estar configuradas para admitir el tipo de red seleccionado.
9. En Subnets (Subredes), RDS Proxy rellena las mismas subredes que el proxy asociado de forma predeterminada. Para restringir el acceso al punto de conexión para que solo una parte del intervalo de direcciones de la VPC pueda conectarse a él, quite una o varias subredes.
10. En VPC security group (Grupos de seguridad de la VPC), puede elegir un grupo de seguridad existente o crear uno nuevo. De forma predeterminada, el proxy de RDS rellena el mismo grupo o grupos de seguridad que el proxy asociado. Si las reglas entrantes y salientes del proxy son apropiadas para este punto de conexión, puede dejar la opción predeterminada.

    Si decide crear un grupo de seguridad nuevo, especifique un nombre para el grupo de seguridad en esta página. A continuación, edite la configuración del grupo de seguridad desde la consola de EC2.
11. Elija Create proxy endpoint (Crear punto de enlace de proxy).

AWS CLI

Para crear un punto de enlace de proxy, utilice el comando de AWS CLI create-db-proxy-endpoint.

Incluya los siguientes parámetros obligatorios:

- `--db-proxy-name` *`value`*
- `--db-proxy-endpoint-name` *`value`*
- `--vpc-subnet-ids` *`list_of_ids`*. Separe los ID de subred con espacios. No se especifica el ID de la VPC en sí.

También puede incluir los siguientes parámetros opcionales:

- `--target-role { READ_WRITE | READ_ONLY }`. El valor predeterminado de este parámetro es `READ_WRITE`. El valor de `READ_ONLY` solo afecta a los clústeres aprovisionados de Aurora que contengan una o más instancias de base de datos de lector. Cuando el proxy está asociado a un clúster de Aurora que solo contiene una instancia de base de datos de escritura, no puede especificar `READ_ONLY`. Para obtener más información sobre el uso previsto de puntos de conexión de solo lectura con clústeres de Aurora, consulte Usar puntos de enlace del lector con los clústeres de Aurora .
- `--vpc-security-group-ids` *`value`*. Separe los ID de grupo de seguridad con espacios. Si omite este parámetro, el proxy de RDS utiliza el grupo de seguridad predeterminado de la VPC. El proxy de RDS determina la VPC en función de los ID de subred que especifique para el parámetro `--vpc-subnet-ids`.
- `--endpoint-network-type { IPV4 | IPV6 | DUAL }`. Este parámetro especifica la versión de IP del punto de conexión del proxy. El valor predeterminado es `IPV4`. Para usar `IPV6` o `DUAL`, la VPC y las subredes deben estar configuradas para admitir el tipo de red seleccionado.

Example

En el ejemplo siguiente se crea un punto de enlace de proxy denominado `my-endpoint`.

Para Linux, macOS o Unix:

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
```

```
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id \
  --endpoint-network-type DUAL
```

Para Windows:

```
aws rds create-db-proxy-endpoint ^
  --db-proxy-name my-proxy ^
  --db-proxy-endpoint-name my-endpoint ^
  --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
  --target-role READ_ONLY ^
  --vpc-security-group-ids security_group_id ^
  --endpoint-network-type DUAL
```

API de RDS

Para crear un punto de conexión proxy, utilice la acción CreateDBProxyEndpoint de la API de RDS.

Visualización de puntos de enlace de proxy

Para ver los puntos de conexión proxy existentes, siga estas instrucciones:

Consola

Para ver los detalles de un punto de enlace de proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Proxies.
3. En la lista, elija el proxy cuyo punto de enlace desea ver. Haga clic en el nombre del proxy para ver su página de detalles.
4. En la sección Proxy endpoints (Puntos de enlace de proxy), elija el punto de enlace que desea ver. Haga clic en su nombre para ver la página de detalles.
5. Examine los parámetros cuyos valores le interesan. Puede comprobar propiedades como las siguientes:

   - Si el punto de enlace es de lectura o escritura o de solo lectura.
   - La dirección de punto de enlace que se utiliza en una cadena de conexión de base de datos.
   - La VPC, las subredes y los grupos de seguridad asociados con el punto de enlace.

AWS CLI

Para ver uno o más puntos de conexión de proxy, utilice el comando de AWS CLI describe-db-proxy-endpoints.

Puede incluir los siguientes parámetros opcionales:

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

En el siguiente ejemplo se describe el punto de enlace de proxy de `my-endpoint`.

Example

Para Linux, macOS o:Unix

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

En:Windows

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

API de RDS

Para describir uno o más puntos de enlace de proxy, utilice la operación de API de RDS DescribeDBProxyEndpoints.

Modificación de un punto de enlace de proxy

Para modificar los puntos de conexión proxy, siga estas instrucciones:

Consola

Para modificar uno o varios puntos de enlace de proxy

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

2. En el panel de navegación, seleccione Proxies.
3. En la lista, elija el proxy cuyo punto de enlace desea modificar. Haga clic en el nombre del proxy para ver su página de detalles.
4. En la sección Proxy endpoints (Puntos de enlace de proxy), elija el punto de enlace que desea modificar. Puede seleccionarlo en la lista o hacer clic en su nombre para ver la página de detalles.
5. En la página de detalles del proxy, en la sección de Proxy endpoints (Puntos de enlace de proxy), elija Edit (Editar). O en la página de detalles del punto de conexión de proxy, en Acciones, elija Editar.
6. Cambie los valores de los parámetros que desee modificar.
7. Elija Guardar cambios.

AWS CLI

Para modificar un punto de conexión de proxy, utilice el comando de AWS CLI modify-db-proxy-endpoint con los siguientes parámetros requeridos:

- `--db-proxy-endpoint-name`

Especifique los cambios en las propiedades del punto de enlace mediante uno o varios de los siguientes parámetros:

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`. Separe los ID de grupo de seguridad con espacios.

En el ejemplo siguiente se cambia el nombre del punto de enlace de proxy de `my-endpoint` a `new-endpoint-name`.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
  --new-db-proxy-endpoint-name new-endpoint-name
```

Para Windows:

```
aws rds modify-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

API de RDS

Para modificar un punto de enlace de proxy, utilice la operación de API de RDS ModifyDBProxyEndpoint.

Eliminación de un punto de enlace de proxy

Para eliminar un punto de conexión para su proxy, siga estas instrucciones:

> **Note**
>
> No puede eliminar el punto de conexión de proxy predeterminado que RDS Proxy crea automáticamente para cada proxy.
> Cuando elimina un proxy, RDS Proxy elimina automáticamente todos los puntos de enlace asociados.

Consola

Para eliminar un punto de enlace de proxy mediante Consola de administración de AWS

1. En el panel de navegación, seleccione Proxies.
2. En la lista, elija el proxy cuyo punto de enlace desea establecer como punto de enlace. Haga clic en el nombre del proxy para ver su página de detalles.
3. En la sección Proxy endpoints (Puntos de enlace de proxy), elija el punto de enlace que desea eliminar. Puede seleccionar uno o varios puntos de enlace de la lista o hacer clic en el nombre de un punto de enlace único para ver la página de detalles.
4. En la página de detalles del proxy, en la sección de Proxy endpoints (Puntos de enlace de proxy), elija Delete (Eliminar). O en la página de detalles del punto de conexión de proxy, en Acciones, elija Eliminar.

AWS CLI

Para eliminar un punto de enlace de proxy, ejecute el comando delete-db-proxy-endpoint con los siguientes parámetros requeridos:

- `--db-proxy-endpoint-name`

El siguiente comando elimina el punto de enlace de proxy denominado `my-endpoint`.

Para Linux, macOS o:Unix

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

En:Windows

```
aws rds delete-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint
```

API de RDS

Para eliminar un punto de enlace de proxy con la API de RDS, ejecute la operación DeleteDBProxyEndpoint. Especifique el nombre del punto de enlace de proxy para el parámetro `DBProxyEndpointName`.

Supervisión de las métricas de RDS Proxy con Amazon CloudWatch

Puede monitorear el proxy de RDS mediante Amazon CloudWatch. CloudWatch recopila y procesa los datos sin procesar de los proxies en métricas legibles y casi en tiempo real. Para buscar estas métricas en la consola de CloudWatch, seleccione Metrics (Métricas), a continuación, RDS y, a continuación, Per-Proxy Metrics (Métricas por proxy). Para obtener más información, consulte Uso de métricas de Amazon CloudWatch en la Guía del usuario de Amazon CloudWatch.

> **Note**
>
> RDS publica estas métricas para cada instancia Amazon EC2 subyacente asociada con un proxy. Es posible que más de una instancia EC2 sirva un proxy único. Utilice estadísticas de CloudWatch para agregar los valores de un proxy en todas las instancias asociadas.

Es posible que algunas de estas métricas no estén visibles hasta después de la primera conexión correcta a través de un proxy.

En los registros de RDS Proxy, cada entrada tiene el prefijo del nombre del punto de enlace de proxy asociado. Este nombre puede ser el especificado para un punto de conexión definido por el usuario, o el nombre especial `default` para el punto de conexión predeterminado de un proxy que realiza las solicitudes de lectura/escritura.

Todas las métricas de RDS Proxy están en el grupo `proxy`.

Cada punto de enlace de proxy tiene sus propias métricas de CloudWatch. Puede monitorear el uso de cada punto de enlace de proxy de forma independiente. Para obtener más información acerca de los puntos de enlace de proxy, consulte Trabajo con puntos de enlace del proxy de Amazon RDS.

Puede agregar los valores de cada métrica mediante uno de los siguientes conjuntos de dimensiones. Por ejemplo, mediante el uso del conjunto de dimensiones de `ProxyName`, puede analizar todo el tráfico de un proxy determinado. Mediante el uso de los otros conjuntos de dimensiones, puede dividir las métricas de diferentes maneras. Puede dividir las métricas en función de los diferentes puntos de enlace o bases de datos de destino de cada proxy, o el tráfico de lectura o escritura y de solo lectura a cada base de datos.

- Conjunto de dimensiones 1 : `ProxyName`
- Conjunto de dimensiones 2 : `ProxyName`, `EndpointName`
- Conjunto de dimensiones 3 : `ProxyName`, `TargetGroup`, `Target`
- Conjunto de dimensiones 4 : `ProxyName`, `TargetGroup`, `TargetRole`

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
`AvailabilityPercentage`	El porcentaje de tiempo para el que el grupo de destino estaba disponible en el rol indicado por la dimensión. Se informa	1 minuto	Dimension set 4

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
	de esta métrica cada minuto. La estadísti ca más útil para esta métrica es Sum.		
ClientCon nections	El número actual de conexiones de cliente. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es Sum.	1 minuto	Dimension set 1, Dimension set 2
ClientCon nectionsClosed	El número de conexiones de cliente cerradas. La estadísti ca más útil para esta métrica es Sum.	1 minuto o más	Dimension set 1, Dimension set 2
ClientCon nectionsI nSetup	El número actual de conexiones de cliente abiertas, pero que no han completad o la configuración. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es Suma.	1 minuto	Dimension set 1, Dimension set 2

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
`ClientConnectionsNoTLS`	Número actual de conexiones de cliente sin Transport Layer Security (TLS). Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es `Sum`.	1 minuto	Dimension set 1, Dimension set 2
`ClientConnectionsReceived`	El número de solicitudes de conexión de cliente recibidas. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 2
`ClientConnectionsSetupFailedAuth`	El número de intentos de conexión de cliente que produjeron un error debido a una autenticación o TLS mal configurada. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 2

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
`ClientConnectionsSetupSucceeded`	El número de conexiones de cliente establecidas correctamente con cualquier mecanismo de autenticación con o sin TLS. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 2
`ClientConnectionsTLS`	El número actual de conexiones de cliente con TLS. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es `Sum`.	1 minuto	Dimension set 1, Dimension set 2
`DatabaseConnectionRequests`	El número de solicitudes para crear una conexión de base de datos. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionRequestsWithTLS`	El número de solicitudes para crear una conexión de base de datos con TLS. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 3, Dimension set 4

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
DatabaseConnections	El número actual de conexiones de base de datos. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es Sum.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
DatabaseConnectionsBorrowLatency	El tiempo en microsegundos que tarda el proxy que se monitorea en obtener una conexión de base de datos. La estadística más útil para esta métrica es Sum.	1 minuto o más	Dimension set 1, Dimension set 2
DatabaseConnectionsCurrentlyBorrowed	El número actual de conexiones de base de datos en estado de préstamo. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es Sum.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
`DatabaseConnectionsCurrentlyInTransaction`	El número actual de conexiones de base de datos en una transacción. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlySessionPinned`	El número actual de conexiones de base de datos fijadas actualmente debido a operaciones en solicitudes de cliente que cambian el estado de la sesión. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsSetupFailed`	El número de solicitudes de conexión de base de datos que produjeron un error. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 3, Dimension set 4

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
`DatabaseConnectionsSetupSucceeded`	El número de conexiones de base de datos establecidas correctamente con o sin TLS. La estadística más útil para esta métrica es `Sum`.	1 minuto o más	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsWithTLS`	El número actual de conexiones de base de datos con TLS. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`MaxDatabaseConnectionsAllowed`	El número máximo de conexiones de base de datos permitidas. Se informa de esta métrica cada minuto. La estadística más útil para esta métrica es `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`QueryDatabaseResponseLatency`	El tiempo en microsegundos que la base de datos tardó en responder a la consulta. La estadística más útil para esta métrica es `Average`.	1 minuto o más	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
QueryRequests	El número de consultas recibidas . Una consulta que incluye varias instrucciones se cuenta como una consulta. La estadística más útil para esta métrica es Sum.	1 minuto o más	Dimension set 1, Dimension set 2
QueryRequestsNoTLS	El número de consultas recibidas de conexiones que no son TLS. Una consulta que incluye varias instrucciones se cuenta como una consulta. La estadística más útil para esta métrica es Sum.	1 minuto o más	Dimension set 1, Dimension set 2
QueryRequestsTLS	El número de consultas recibidas de conexiones TLS. Una consulta que incluye varias instrucciones se cuenta como una consulta. La estadística más útil para esta métrica es Sum.	1 minuto o más	Dimension set 1, Dimension set 2

Métrica	Descripción	Período de validez	Conjunto de dimensiones de CloudWatch
`QueryResponseLatency`	El tiempo en microsegundos entre obtener una solicitud de consulta y que el proxy la responda. La estadística más útil para esta métrica es `Average`.	1 minuto o más	Dimension set 1, Dimension set 2

Puede encontrar registros de actividad del proxy de RDS en CloudWatch en la Consola de administración de AWS. Cada proxy tiene una entrada en la página Log groups (Grupos de registro).

> **Important**
>
> Estos registros están destinados al consumo humano para solucionar problemas y no para el acceso mediante programación. El formato y el contenido de los registros están sujetos a cambios.
>
> En particular, los registros más antiguos no contienen prefijos que indiquen el punto de enlace de cada solicitud. En los registros más recientes, cada entrada tiene el prefijo del nombre del punto de enlace de proxy asociado. Este nombre puede ser el que especificó para un punto de enlace definido por el usuario, o el nombre especial `default` para las solicitudes que utilizan el punto de enlace predeterminado de un proxy.

Trabajo con eventos de RDS Proxy

Un evento indica un cambio en un entorno, como un entorno de AWS, un servicio o aplicación de un socio de software como servicio (SaaS). O bien, puede ser una de sus propias aplicaciones o servicios personalizados. Por ejemplo, Amazon Aurora genera un evento al crear o modificar una instancia de RDS Proxy. Amazon Aurora envía eventos a Amazon EventBridge casi en tiempo real. A continuación, encontrará una lista de eventos de RDS Proxy a los que puede suscribirse y un ejemplo de evento de RDS Proxy.

Para obtener más información acerca de cómo trabajar con eventos, consulte lo siguiente:

- Para obtener instrucciones acerca de cómo ver los eventos mediante la Consola de administración de AWS, la AWS CLI o la API de RDS, consulte Consulta de eventos de Amazon RDS.
- Para obtener información sobre cómo configurar Amazon Aurora para enviar eventos a EventBridge, consulte Creación de una regla que se desencadena en función de un evento Amazon Aurora.

Eventos de RDS Proxy

En la siguiente tabla se muestran las categorías de eventos y una lista de los eventos que pueden producirse cuando el tipo de origen es una instancia de RDS Proxy.

Categoría	ID de evento de RDS	Mensaje	Notas
configuration change	RDS-EVENT-0204	RDS ha modificado el proxy de la base de datos *nombre*.	Ninguna
configuration change	RDS-EVENT-0207	RDS ha modificado el punto de conexión del proxy de la base de datos *nombre*.	Ninguna
configuration change	RDS-EVENT-0213	RDS ha detectado la adición de la instancia de base de datos y la ha agregado automáticamente al grupo de destino del proxy de la base de datos *nombre*.	Ninguna
configuration change	RDS-EVENT-0214	RDS ha detectado la eliminación de la instancia de base de datos *nombre* y la ha borrado automátic	Ninguna

Categoría	ID de evento de RDS	Mensaje	Notas
		amente del grupo de destino *nombre* del proxy de la base de datos *nombre*.	
configuration change	RDS-EVENT-0215	RDS ha detectado la eliminación del clúster de base de datos *nombre* y la ha borrado automátic amente del grupo de destino *nombre* del proxy de la base de datos *nombre*.	Ninguna
creación	RDS-EVENT-0203	RDS ha creado el proxy de la base de datos *nombre*.	Ninguna
creación	RDS-EVENT-0206	RDS ha creado el punto de conexión *nombre* del para el proxy de la base de datos *nombre*.	Ninguna
deletion	RDS-EVENT-0205	RDS ha eliminado el proxy de la base de datos *nombre*.	Ninguna
deletion	RDS-EVENT-0208	RDS ha eliminado el punto de conexión *nombre* del proxy de la base de datos *nombre*.	Ninguna

Categoría	ID de evento de RDS	Mensaje	Notas
failure	RDS-EVENT-0243	RDS no ha podido aprovisionar capacidad para el proxy *nombre* porque no hay suficientes direcciones IP disponibles en las subredes: *nombre*. Para resolver el problema, asegúrese de que sus subredes tengan el número mínimo de direcciones IP sin usar, tal como se recomienda en la documentación de RDS Proxy.	Para determinar el número recomendado para la clase de instancia, consulte Planificación de la capacidad de direcciones IP.
failure	RDS-EVENT-0275	RDS ha limitado algunas conexiones al proxy de base de datos *nombre*. El número de solicitudes de conexión simultáneas del cliente al proxy ha superado el límite.	Ninguna

A continuación, se muestra un ejemplo de un evento de RDS Proxy en formato JSON. El evento muestra que RDS modificó el punto de conexión denominado `my-endpoint` de la instancia de RDS Proxy denominada `my-rds-proxy`. El ID de evento es RDS-EVENT-0207.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
```

```
  "region": "us-east-1",
  "resources": [
     "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```

Ejemplos de línea de comandos del proxy de RDS

Para ver cómo interactúan con RDS Proxy las combinaciones de comandos de conexión y las instrucciones SQL, consulte los siguientes ejemplos.

Ejemplos

- Preserving Connections to a MySQL Database Across a Failover
- Adjusting the max_connections Setting for an Aurora DB Cluster

Example Conservación de las conexiones en una base de datos de MySQL a través de una conmutación por error

En este ejemplo de MySQL se muestra cómo las conexiones abiertas continúan funcionando durante una conmutación por error. Un ejemplo es cuando se reinicia una base de datos o deja de estar disponible debido a un problema. En este ejemplo se utiliza un proxy denominado `the-proxy` y un clúster de bases de datos de Aurora con instancias de base de datos `instance-8898` y `instance-9814`. Cuando el comando `failover-db-cluster` se ejecuta desde la línea de comandos de Linux, la instancia de escritor a la que está conectado el proxy cambia a una instancia de base de datos diferente. Puede ver que la instancia de base de datos asociada con el proxy cambia mientras la conexión permanece abierta.

```
$ mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p
```

```
Enter password:
...

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ # Initially, instance-9814 is the writer.
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-8898 is the writer.
$ fg
mysql -h the-proxy.proxy-demo.us.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-8898      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-9814 is the writer again.
$ fg
mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
```

```
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)
+---------------+---------------+
| Variable_name | Value         |
+---------------+---------------+
| hostname      | ip-10-1-3-178 |
+---------------+---------------+
1 row in set (0.02 sec)
```

Example Ajuste de la configuración max_connections para un clúster de bases de datos de Aurora

En este ejemplo se muestra cómo se puede ajustar la configuración `max_connections` de un clúster de bases de datos de Aurora MySQL. Para ello, cree su propio grupo de parámetros de clúster de bases de datos basado en la configuración predeterminada de parámetros para clústeres compatibles con MySQL 5.7. Especifique un valor para la configuración `max_connections`, anulando la fórmula que establece el valor predeterminado. Asocie el grupo de parámetros del clúster de bases de datos con el clúster de bases de datos.

```
export REGION=us-east-1
export CLUSTER_PARAM_GROUP=rds-proxy-mysql-57-max-connections-demo
export CLUSTER_NAME=rds-proxy-mysql-57

aws rds create-db-parameter-group --region $REGION \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-parameter-group-name $CLUSTER_PARAM_GROUP \
  --description "Aurora MySQL 5.7 cluster parameter group for RDS Proxy demo."

aws rds modify-db-cluster --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP

echo "New cluster param group is assigned to cluster:"
aws rds describe-db-clusters --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --query '*[*].{DBClusterParameterGroup:DBClusterParameterGroup}'

echo "Current value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"
```

```
echo -n "Enter number for max_connections setting: "
read answer

aws rds modify-db-cluster-parameter-group --region $REGION --db-cluster-parameter-
group-name $CLUSTER_PARAM_GROUP \
  --parameters "ParameterName=max_connections,ParameterValue=$
$answer,ApplyMethod=immediate"

echo "Updated value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"
```

Solución de problemas de RDS Proxy

A continuación, puede encontrar ideas de solución de problemas para algunos problemas de RDS Proxy comunes e información sobre registros de CloudWatch para RDS Proxy.

En los registros de RDS Proxy, cada entrada tiene el prefijo del nombre del punto de enlace de proxy asociado. Este nombre puede ser el especificado para un punto de conexión definido por el usuario. O puede ser el nombre especial `default` para el punto de conexión predeterminado de un proxy que lleva a cabo solicitudes de lectura/escritura. Para obtener más información acerca de los puntos de enlace de proxy, consulte Trabajo con puntos de enlace del proxy de Amazon RDS.

Temas

- Verificación de la conectividad para un proxy
- Problemas y soluciones comunes
- Solución de problemas de RDS Proxy con RDS para MySQL
- Solución de problemas de RDS Proxy con RDS para PostgreSQL

Verificación de la conectividad para un proxy

Puede utilizar los siguientes comandos para comprobar que todos los componentes, como el proxy, la base de datos y las instancias de computación de la conexión, se pueden comunicar entre sí.

Examine el propio proxy usando el comando describe-db-proxies. Examine también el grupo de destino asociado mediante el comando describe-db-proxy-target-groups. Compruebe que los detalles

de los destinos coincidan con el clúster de Aurora que desea asociar con el proxy. Utilice comandos como los siguientes.

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

Para confirmar que el proxy puede conectarse a la base de datos subyacente, examine los destinos especificados en los grupos de destino mediante el comando describe-db-proxy-targets. Utilice un comando como el siguiente.

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

El resultado del comando describe-db-proxy-targets incluye un campo `TargetHealth`. Puede examinar los campos `State`, `Reason` y `Description` dentro de `TargetHealth` para comprobar si el proxy puede comunicarse con la instancia de base de datos subyacente.

- Un valor `State` de `AVAILABLE` indica que el proxy puede conectarse a la instancia de base de datos.
- Un valor `State` de `UNAVAILABLE` indica un problema de conexión temporal o permanente. En este caso, examine los campos `Reason` y `Description`. Por ejemplo, si `Reason` tiene un valor de `PENDING_PROXY_CAPACITY`, intente conectarse de nuevo después de que el proxy finalice su operación de escalado. Si `Reason` tiene un valor de `UNREACHABLE`, `CONNECTION_FAILED` o `AUTH_FAILURE`, utilice la explicación del campo `Description` que le ayudará a diagnosticar el problema.
- El campo `State` es posible que tenga un valor de `REGISTERING` durante un breve tiempo antes de cambiar a `AVAILABLE` o `UNAVAILABLE`.

Si el siguiente comando Netcat (`nc`) se ejecuta correctamente, puede acceder al punto de enlace del proxy desde la instancia de EC2 u otro sistema en el que haya iniciado sesión. Este comando notifica un error si no está en la misma VPC que el proxy y la base de datos asociada. Es posible que pueda iniciar sesión directamente en la base de datos sin estar en la misma VPC. Sin embargo, no puede iniciar sesión en el proxy a menos que esté en la misma VPC.

```
nc -zx MySQL_proxy_endpoint 3306

nc -zx PostgreSQL_proxy_endpoint 5432
```

Puede utilizar los siguientes comandos para asegurarse de que la instancia de EC2 tenga las propiedades requeridas. Algo especialmente importante es que la VPC para la instancia de EC2 debe ser la misma que la VPC para la instancia de base de datos de RDSel clúster de Aurora donde se conecta el proxy.

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

Examine los secretos de Secrets Manager utilizados para el proxy.

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

Asegúrese de que el campo `SecretString` que muestra `get-secret-value` está codificado como una cadena JSON que incluye los campos `username` y `password`. En el ejemplo siguiente se muestra el formato del campo `SecretString`.

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

Al resolver problemas de autenticación de IAM, verifique lo siguiente:

- La base de datos tiene habilitada la autenticación de IAM.
- El proxy está configurado con el esquema de autenticación correcto.
- Las políticas de IAM del rol de IAM proporcionadas al proxy conceden el permiso `rds-db:connect` necesario a la base de datos correspondiente y su nombre de usuario.
- Para la autenticación de IAM integral, existen usuarios de bases de datos que coinciden con los nombres de usuario o rol de IAM.
- SSL/TLS está habilitado para la conexión.

Problemas y soluciones comunes

En esta sección, se describen algunos problemas comunes y posibles soluciones al utilizar RDS Proxy.

Después de ejecutar el comando de la CLI `aws rds describe-db-proxy-targets`, si en la descripción `TargetHealth` se indica `Proxy does not have any registered credentials`, verifique lo siguiente:

- Hay credenciales registradas para que el usuario acceda al proxy.
- El rol de IAM para acceder al secreto de Secrets Manager utilizado por el proxy es válido.

Es posible que encuentre los siguientes eventos de RDS al crear o conectarse a un proxy de base de datos.

Categoría	ID de evento de RDS	Descripción
error	RDS-EVENT-0243	RDS no ha podido aprovisionar capacidad para el proxy porque no hay suficientes direcciones IP disponibles en las subredes. Para resolver el problema, asegúrese de que sus subredes tengan el número mínimo de direcciones IP sin usar. Para determinar el número recomendado para la clase de instancia, consulte Planificación de la capacidad de direcciones IP.
error	RDS-EVENT-0275	RDS ha limitado algunas conexiones al proxy de base de datos *nombre*. El número de solicitudes de conexión simultáneas del cliente al proxy ha superado el límite.

Es posible que encuentre los siguientes problemas al crear un nuevo proxy o al conectarse a un proxy.

Error	Causas o soluciones provisionales
`403: The security token included in the request is invalid`	Seleccione un rol de IAM existente en lugar de elegir crear uno nuevo.

Solución de problemas de RDS Proxy con RDS para MySQL

Es posible que encuentre los siguientes problemas al conectarse a un proxy MySQL.

Error	Causas o soluciones provisionales
`ERROR 1040 (HY000): Connections rate limit exceeded (`*`limit_value`*` )`	La tasa de solicitudes de conexión del cliente al proxy ha superado el límite.
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	El número de solicitudes simultáneas con autenticación de IAM desde el cliente al proxy ha superado el límite.
`ERROR 1040 (HY000): Number simultaneous connections exceeded (`*`limit_value`*` )`	El número de solicitudes de conexión simultáneas del cliente al proxy ha superado el límite.

Error	Causas o soluciones provisionales
`ERROR 1045 (28000): Access denied for user 'DB_USER'@'%' (usi password: YES)`	El secreto de Secrets Manager utilizado por el proxy no coincide con el nombre de usuario y la contraseña de un usuario de base de datos existente. Actualice las credenciales en el secreto de Secrets Manager o asegúrese de que el usuario de la base de datos existe y tiene la misma contraseña que en el secreto.
`ERROR 1105 (HY000): Unknown error`	Se ha producido un error desconocido.
`ERROR 1231 (42000): Variable ''charact er_set_cl ient'' can't be set to the value of value`	El valor establecido para el parámetro `character_set_client` no es válido. Por ejemplo, el valor `ucs2` no es válido porque puede bloquear el servidor MySQL.
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	Ha habilitado la configuración Exigir Transport Layer Security en el proxy pero su conexión incluyó el parámetro `ssl-mode=DISABLED` en el cliente de MySQL. Haga una de estas dos operaciones: • Desactive la configuración Exigir Transport Layer Security para el proxy. • Conéctese a la base de datos mediante la configuración mínima de `ssl-mode=REQUIRED` en el cliente de MySQL.
`ERROR 2026 (HY000): SSL connection error: Internal Server Error`	Error en el protocolo de enlace TLS al proxy. Algunas posibles razones incluyen las siguientes: • Se requiere SSL, pero el servidor no lo admite. • Se ha producido un error interno del servidor. • Se ha producido un protocolo de enlace erróneo.

Error	Causas o soluciones provisionales
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	Se agotó el tiempo de espera del proxy mientras esperaba a adquirir una conexión a la base de datos. Algunas posibles razones incluyen las siguientes: • El proxy no puede establecer una conexión con la base de datos porque se ha llegado al máximo de conexiones. • El proxy no puede establecer una conexión a base de datos porque la base de datos no está disponible.

Solución de problemas de RDS Proxy con RDS para PostgreSQL

Es posible que encuentre los siguientes problemas al conectarse a un proxy PostgreSQL.

Error	Causa	Solución
`ERROR 28000: IAM authentication is allowed only with SSL connections.`	El usuario ha intentado conectarse a la base de datos mediante la autenticación de IAM con la configuración `sslmode=disable` en el cliente PostgreSQL.	El usuario necesita conectarse a la base de datos utilizando la configuración mínima de `sslmode=require` en el cliente PostgreSQL. Para obtener más información, consulte la documentación de Soporte de SSL de PostgreSQL.
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the credentials for this role and try again.`	No hay ningún secreto de Secrets Manager para este rol.	Agregue un secreto de Secrets Manager para este rol. Para obtener más información, consulte Configuración de la autenticación de IAM para RDS Proxy.
`ERROR 28000: RDS supports only`	El cliente de base de datos que se utiliza para conectarse al proxy utiliza un	Si no utiliza la autenticación IAM, utilice la autenticación de contraseñas MD5 o SCRAM.

Error	Causa	Solución
`IAM, MD5, or SCRAM authentication.`	mecanismo de autenticación que actualmente no admite el proxy.	
`ERROR 28000: A user name is missing from the connectio n startup packet. Provide a user name for this connection.`	El cliente de base de datos que se utiliza para conectars e al proxy no envía un nombre de usuario al intentar establecer una conexión.	Asegúrese de definir un nombre de usuario al configurar una conexión con el proxy utilizando el cliente PostgreSQL de su elección.
`ERROR 28000: IAM is allowed only with SSL connections.`	Un cliente intentó conectars e mediante la autenticación de IAM, pero SSL no estaba habilitado.	Habilite SSL en el cliente PostgreSQL.
`ERROR 28000: This RDS Proxy requires TLS connections.`	El usuario habilitó la opción Exigir Transport Layer Security pero intentó conectarse con `sslmode=d isable` en el cliente de PostgreSQL.	Para corregir este error, realice alguna de las siguiente s acciones: • Desactive la opción del proxy Exigir Transport Layer Security. • Conectarse a la base de datos mediante la configuración mínima de `sslmode=allow` en el cliente PostgreSQL.

Error	Causa	Solución
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	Este error es posible que se deba a las siguientes razones: • El cliente proporcionó el nombre de usuario de IAM incorrecto. • El cliente proporcionó un token de autorización de IAM incorrecto para el usuario. • El cliente está utilizando una política de IAM que no tiene los permisos necesarios. • El cliente proporcionó un token de autorización de IAM caducado para el usuario.	Para corregir este error, haga lo siguiente: 1. Confirme que existe el usuario de IAM proporcio nado. 2. Confirme que el token de autorización de IAM pertenece al usuario de IAM proporcionado. 3. Confirme que la política de IAM tiene los permisos adecuados para RDS. 4. Compruebe la validez del token de autorización de IAM utilizado.
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	La contraseña de este rol no coincide con el secreto de Secrets Manager.	Compruebe el secreto de este rol en Secrets Manager para ver si la contraseña es la misma que la que se está utilizando en su cliente PostgreSQL.
`ERROR 28P01: The IAM authentication failed for the role` *`role_name`* `. Check the IAM token for this role and try again.`	Hay un problema con el token de IAM utilizado para la autenticación de IAM.	Genere un nuevo token de autenticación y úselo en una nueva conexión.

Error	Causa	Solución
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	El cliente PostgreSQL utilizado para conectarse al proxy utiliza un protocolo anterior a 3.0.	Utilice un cliente PostgreSQL más reciente que sea compatible con el protocolo de mensajería 3.0. Si está utilizando la CLI `psql` de PostgreSQL, utilice una versión mayor o igual a 7.4.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	El cliente PostgreSQL utilizado para conectarse al proxy está intentando utilizar el modo de replicación de streaming, que actualmente no es compatible con el proxy RDS.	Desactive el modo de replicación de streaming en el cliente de PostgreSQL que se utiliza para conectarse.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option` *`option_name`* `.`	A través del mensaje de inicio, el cliente PostgreSQL utilizado para conectarse al proxy solicita una opción que actualmente no es compatible con el proxy RDS.	Desactive la opción que se muestra como no compatible con el mensaje anterior en el cliente de PostgreSQL que se utiliza para conectarse.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	El número de solicitudes simultáneas con autenticación de IAM desde el cliente al proxy ha superado el límite.	Reduzca la velocidad en la que se establecen las conexiones que utilizan la autenticación de IAM desde un cliente PostgreSQL.
`ERROR 53300: The maximum number of client connections to the proxy exceeded` *`number_value`* `.`	El número de solicitudes de conexión simultáneas del cliente al proxy ha superado el límite.	Reduzca el número de conexiones activas de clientes PostgreSQL a este proxy RDS.

Error	Causa	Solución
`ERROR 53300: Rate of connection to proxy exceeded` *`number_value`* `.`	La tasa de solicitudes de conexión del cliente al proxy ha superado el límite.	Reduzca la velocidad en la que se establecen las conexiones de un cliente PostgreSQL.
`ERROR XX000: Unknown error.`	Se ha producido un error desconocido.	Contacte con AWS Support para que investiguemos el problema.
`ERROR 08000: Timed-out waiting to acquire database connection.`	El proxy ha agotado el tiempo de espera para adquirir una conexión de base de datos en el tiempo especificado por la configuración `ConnectionBorrowTimeout` . Algunas posibles razones incluyen las siguientes: • El proxy no puede establecer una conexión con la base de datos porque se ha alcanzado el número máximo de conexiones. • El proxy no puede establecer una conexión con la base de datos porque esta no está disponible o porque el establecimiento de la conexión con la base de datos tarda más que valor de `ConnectionBorrowTimeout` configurado.	Las posibles soluciones son las siguientes: • Evite fijar las conexiones de proxy. Consulte Cómo evitar la fijación de RDS Proxy. • Revise las configuraciones `ConnectionBorrowTimeout` y `MaxConnectionsPercent` . Consulte Observaciones sobre la conexión de RDS Proxy. • Revise la disponibilidad del destino. Consulte `AvailabilityPercentage` en Supervisión de las métricas de RDS Proxy con Amazon CloudWatch.

Error	Causa	Solución
`ERROR XX000: Request returned an error: database_error .`	La conexión de base de datos establecida desde el proxy devolvió un error.	La solución depende del error específico de la base de datos. Un ejemplo es: `Request returned an error: database "your-database-name" does not exist`. Esto significa que el nombre de base de datos especificado no existe en el servidor de bases de datos. O significa que el nombre de usuario utilizado como nombre de base de datos (si no se especifica un nombre de base de datos) no existe en el servidor.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	El número de solicitudes simultáneas con autentica ción de IAM desde el cliente al proxy ha superado el límite.	Reduzca la velocidad en la que se establecen las conexiones que utilizan la autenticación de IAM desde un cliente PostgreSQL.

Error	Causa	Solución
`ERROR 28000: Enable IAM authentication for the client connection to the proxy and try again.`	RDS Proxy no se puede conectar a la base de datos porque la autenticación de IAM no está habilitada para la conexión del cliente al proxy. Esto ocurre cuando el parámetro `DefaultAuthScheme` del proxy está configurado como `IAM_AUTH` con un usuario registrado, pero el cliente utiliza la autenticación de contraseñas en lugar de la autenticación de IAM.	Habilite la autenticación de IAM para la conexión del cliente al proxy e inténtelo de nuevo.
`ERROR 28000: Configure IAM authentication as the DefaultAuthScheme in your proxy and try again.`	RDS Proxy no se puede conectar a la base de datos porque `DefaultAuthScheme` no está configurado en `IAM_AUTH`. El parámetro `DefaultAuthScheme` del proxy está establecido en `NONE`, pero el cliente está intentando utilizar la autenticación de IAM.	Establezca `DefaultAuthScheme` en `IAM_AUTH` para el proxy e inténtelo de nuevo.

Se ha eliminado la solución de problemas de una base de datos de **`postgres`**

Si elimina la base de datos de `postgres` de la instancia por error, tendrá que restaurarla para restablecer la conectividad con la instancia. Ejecute los siguientes comandos dentro de la instancia de base de datos:

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Uso del proxy de RDS con AWS CloudFormation

Puede usar el proxy de RDS con AWS CloudFormation. Esto le ayuda a crear grupos de recursos relacionados. Dicho grupo puede incluir un proxy que se puede conectar a una clúster de bases de datos de Aurora. La compatabilidad del proxy de RDS en CloudFormation implica dos nuevos tipos de registro: `DBProxy` y `DBProxyTargetGroup`.

En la siguiente descripción, se muestra una plantilla de CloudFormation de ejemplo para el proxy de RDS.

```
Resources:
 DBProxy:
   Type: AWS::RDS::DBProxy
   Properties:
     DBProxyName: CanaryProxy
     EngineFamily: MYSQL
     RoleArn:
      Fn::ImportValue: SecretReaderRoleArn
     Auth:
       - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
     VpcSubnetIds:
       Fn::Split: [",", "Fn::ImportValue": SubnetIds]

 ProxyTargetGroup:
   Type: AWS::RDS::DBProxyTargetGroup
   Properties:
     DBProxyName: CanaryProxy
     TargetGroupName: default
     DBInstanceIdentifiers:
       - Fn::ImportValue: DBInstanceName
   DependsOn: DBProxy
```

Para obtener más información sobre los recursos de este ejemplo, consulte `DBProxy` y `DBProxyTargetGroup`.

Para obtener más información acerca de los recursos que puede crear mediante CloudFormation, consulte la Referencia del tipo de recurso de RDS.

Uso de RDS Proxy con bases de datos globales de Aurora

Una base de datos global de Aurora es una sola base de datos que abarca varias Regiones de AWS, lo que permite lecturas globales de baja latencia y la recuperación de desastres provocados por las interrupciones que afectan a regiones enteras. Proporciona tolerancia a fallos integrada para la implementación porque la instancia de base de datos no depende de una sola Región de AWS, sino de varias regiones y zonas de disponibilidad diferentes. Para obtener más información, consulte Uso de una base de datos global de Amazon Aurora.

Se puede utilizar RDS Proxy con cualquier clúster de base de datos global de Aurora. Antes de empezar a utilizar estas funciones en conjunto, debe familiarizarse con la siguiente información.

> **Important**
>
> Si el clúster de base de datos forma parte de una base de datos global con el reenvío de escritura activado, reduzca el valor `MaxConnectionsPercent` del proxy según la cuota asignada para el reenvío de escritura. La cuota de reenvío de escritura se establece en el parámetro del clúster de base de datos `aurora_fwd_writer_max_connections_pct`. Para obtener información sobre el reenvío de escritura, consulte Uso del reenvío de escritura en una base de datos Amazon Aurora global.

Limitaciones de RDS Proxy con bases de datos globales

Cuando el clúster de base de datos de Aurora tiene activado el reenvío de escritura, RDS Proxy no admite el valor `SESSION` de la variable `aurora_replica_read_consistency`. Esto puede provocar un comportamiento inesperado.

Cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales

Cuando comprenda cómo funcionan los puntos de conexión de RDS Proxy con las bases de datos globales, podrá administrar mejor las aplicaciones que utilizan las bases de datos de Aurora con ambas funciones.

En el caso de un proxy con el clúster principal de una base de datos global como destino registrado, los puntos de conexión del proxy funcionan de la misma manera que con cualquier clúster de base de datos de Aurora. Los puntos de conexión de escritura o lectura del proxy envían todas las

solicitudes a la instancia del escritor del clúster. Los puntos de conexión de solo lectura del proxy envían todas las solicitudes a las instancias del lector. Si un lector deja de estar disponible mientras la conexión está abierta, RDS Proxy redirige las consultas posteriores de la conexión a otra instancia del lector. En el caso de un proxy con un clúster secundario como destino registrado, las solicitudes que se envían a los puntos de conexión de solo lectura del proxy también se envían a las instancias del lector. Como el clúster no tiene instancias de escritura, las solicitudes que se envían a los puntos de conexión de lectura o escritura fallan y muestran el error «`The target group doesn't have any associated read/write instances`».

Las operaciones de transición y conmutación por error de base de datos global implican un cambio de rol entre el clúster de base de datos principal y uno de los secundarios. Cuando el clúster secundario seleccionado se convierte en el nuevo principal, una de sus instancias de lector pasa a ser de escritor. Esta instancia de base de datos ahora será la nueva instancia de escritor del clúster global. Asegúrese de redirigir las operaciones de escritura de la aplicación al punto de conexión de lectura o escritura correspondiente del proxy asociado al nuevo clúster principal. Este punto de conexión proxy puede ser el punto de conexión predeterminado o un punto de conexión de lectura o escritura personalizado.

RDS Proxy pone en cola todas las solicitudes a través de puntos de conexión de lectura o escritura y las envía a la instancia de escritor del nuevo clúster principal en cuanto esté disponible. Lo hace independientemente de si se ha completado la operación de transición o conmutación por error. Durante la transición o conmutación por error, el punto de conexión predeterminado del proxy del clúster principal anterior sigue aceptando operaciones de escritura. Sin embargo, en cuanto ese clúster se convierte en un clúster secundario, se produce un error en todas las operaciones de escritura. Para obtener información acerca de cómo y cuándo realizar tareas de transición o conmutación por error globales específicas, consulte los siguientes temas:

- Transición de base de datos global: Ejecución de transiciones para bases de datos globales de Amazon Aurora
- Conmutación por error de base de datos global: Recuperación de una base de datos global Amazon Aurora de una interrupción no planificada

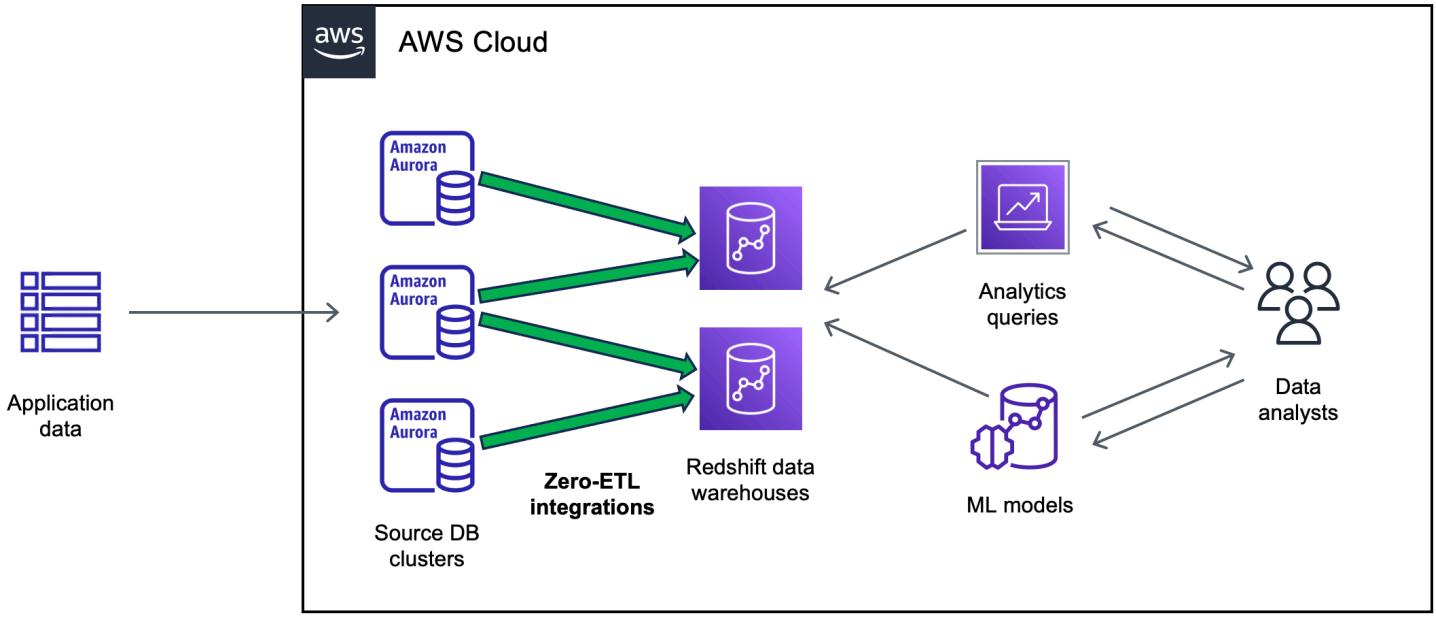
Integraciones sin ETL de Aurora

Una integración sin ETL de Aurora con Amazon Redshift y Amazon SageMaker AI permite realizar análisis y machine learning (ML) casi en tiempo real mediante datos de Aurora. Es una solución totalmente administrada que permite que los datos transaccionales estén disponibles en el destino de análisis después de escribirlos en un clúster de base de datos de Aurora. La extracción, transformación y carga (ETL) es un proceso en el que se combinan datos de numerosos orígenes en un gran almacenamiento de datos central.

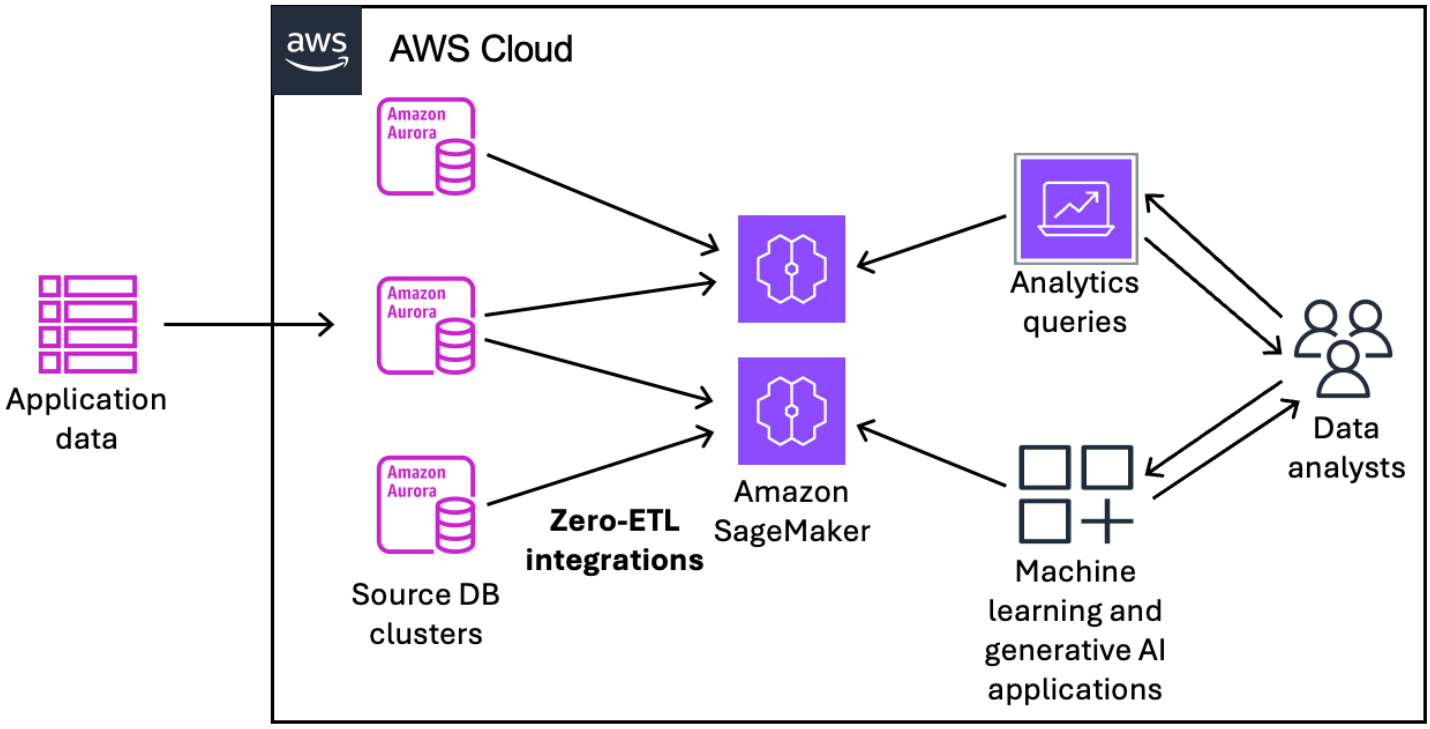
La integración sin ETL hace que los datos de clúster de base de datos de Aurora estén disponibles en Amazon Redshift o un Amazon SageMaker AI Lakehouse prácticamente en tiempo real. Una vez que los datos están en el almacén de datos de destino o lago de datos, puede alimentar cargas de trabajo de análisis, ML e IA con las capacidades integradas, como el machine learning, las vistas materializadas, el uso compartido de datos, el acceso federado a varios almacenamientos de datos y lagos de datos, y las integraciones con Amazon SageMaker AI, Quick Suite y otros Servicios de AWS.

Para crear una integración sin ETL, especifique un clúster de base de datos de Aurora como origen y un almacén de datos o almacén de lago compatible como destino. La integración replica los datos de la base de datos de origen en el almacén de datos de destino o almacén de lago.

El siguiente diagrama ilustra esta funcionalidad para una integración sin ETL con Amazon Redshift:



En el siguiente diagrama se ilustra esta funcionalidad para la integración sin ETL con un Amazon SageMaker AI Lakehouse:



La integración supervisa el estado de la canalización de datos y se recupera de los problemas cuando es posible. Puede crear integraciones a partir de varios clústeres de base de datos de Aurora en un único almacén de datos o almacén de lago de destino, lo que le permite obtener información en varias aplicaciones.

Para obtener información sobre los precios de las integraciones sin ETL, consulte Precios de Amazon Aurora y Precios de Amazon Redshift.

Temas

- Ventajas
- Conceptos clave
- Limitaciones
- Cuotas
- Regiones admitidas
- Introducción a las integraciones sin ETL de Aurora
- Creación de integraciones sin ETL de Aurora con Amazon Redshift
- Creación de integraciones sin ETL de Aurora con un Amazon SageMaker Lakehouse

- Filtrado de datos para integraciones sin ETL de Aurora
- Cómo agregar datos en un clúster de base de datos de Aurora de origen y dirigirle consultas en Amazon Redshift
- Visualización y supervisión de las integraciones sin ETL de Aurora
- Modificación de las integraciones sin ETL de Aurora
- Eliminación de las integraciones sin ETL de Aurora
- Solución de problemas de integraciones sin ETL de Aurora con Amazon Redshift

Ventajas

Las integraciones sin ETL de Aurora tienen los siguientes beneficios:

- Le ayudan a obtener información holística a partir de numerosos orígenes de datos.
- Eliminan la necesidad de crear y mantener canalizaciones de datos complejas que realicen operaciones de extracción, transformación y carga (ETL). Las integraciones sin ETL eliminan los inconvenientes derivados de la creación y administración de canalizaciones, ya que las aprovisionan y administran por usted.
- Reducen la carga operativa y los costos para que pueda centrarse en mejorar sus aplicaciones.
- Le permite aprovechar las capacidades de análisis y ML de destino para obtener información a partir de datos transaccionales y de otro tipo, a fin de responder de manera eficaz a eventos críticos y urgentes.

Conceptos clave

Cuando empiece a utilizar las integraciones sin ETL, tenga en cuenta los siguientes conceptos:

Integración

: Una canalización de datos totalmente administrada que replica automáticamente los datos y esquemas transaccionales de un clúster de base de datos de Aurora a un almacén de datos o catálogo.

Clúster de base de datos de origen

: El clúster de base de datos de Aurora desde donde se replican los datos. Puede especificar un clúster de base de datos que utilice instancias de base de datos aprovisionadas o instancias de base de datos de Aurora Serverless v2 como origen.

Destino

El almacén de datos o almacén de lago en el que se replican los datos. Hay dos tipos de almacenamientos de datos: un almacenamiento de datos de clústeres aprovisionados y un almacenamiento de datos sin servidor. Un almacenamiento de datos de clústeres aprovisionados es una colección de recursos de computación denominados nodos que están organizados en un grupo llamado clúster. Un almacenamiento de datos sin servidor se compone de un grupo de trabajo que almacena los recursos de computación y un espacio de nombres que aloja los objetos y usuarios de la base de datos. Ambos almacenes de datos ejecutan un motor de análisis y contienen una o más bases de datos.

Un almacén de lago de destino consta de catálogos, bases de datos, tablas y vistas. Para obtener más información sobre la arquitectura del almacén de lago, consulte SageMaker Lakehouse components en la Guía del usuario de Amazon SageMaker AI Unified Studio.

Múltiples clústeres de base de datos de origen pueden escribir en el mismo destino.

Para obtener más información, consulte Arquitectura del sistema de almacenamiento de datos en la Guía del desarrollador de Amazon Redshift.

Limitaciones

Las siguientes limitaciones se aplican a las integración sin ETL de Aurora.

Temas

- Limitaciones generales
- Limitaciones de Aurora MySQL
- Limitaciones de Aurora PostgreSQL
- Limitaciones de Amazon Redshift
- Amazon SageMaker AILimitaciones de Lakehouse

Limitaciones generales

- El clúster de base de datos de origen debe estar en la misma región que el destino.
- No puede cambiar el nombre de un clúster de base de datos ni ninguna de sus instancias si ya tiene integraciones.

- No se pueden crear varias integraciones entre las mismas bases de datos de origen y de destino.
- No puede eliminar un clúster de base de datos que ya tenga integraciones. Primero debes eliminar todas las integraciones asociadas.
- Si detiene el clúster de base de datos de origen, es posible que las últimas transacciones no se repliquen en el destino hasta que reanude el clúster.
- Si el clúster es el origen de una implementación azul/verde, los entornos azul y verde no pueden tener integraciones sin ETL existentes durante la transición. Primero debe eliminar la integración, realizar la transición y, a continuación, volver a crear la integración.
- Un clúster de base de datos debe contener al menos una instancia de base de datos para ser el origen de una integración.
- No puede crear una integración para un clúster de base de datos de origen que sea un clon entre cuentas, como los que se comparten mediante AWS Resource Access Manager (AWS RAM).
- Si el clúster de origen es el clúster de base de datos primario de una base de datos global de Aurora y se cambia por error a uno de sus clústeres secundarios, la integración queda inactiva. Debe eliminar y volver a crear la integración.
- No puede crear una integración para una base de datos de origen en la que se esté creando otra integración de forma activa.
- Cuando se crea una integración por primera vez, o cuando se vuelve a sincronizar una tabla, la transferencia de datos del origen al destino puede tardar entre 20 y 25 minutos o más, en función del tamaño de la base de datos de origen. Este retardo puede provocar un aumento del retardo en la réplica.
- Algunos tipos de datos no son compatibles. Para obtener más información, consulte the section called “Diferencias de tipos de datos”.
- Las tablas del sistema, las tablas temporales y las vistas no se replican en almacenes de destino.
- Al ejecutar comandos DDL (por ejemplo, `ALTER TABLE`) en una tabla de origen, se puede desencadenar una resincronización de la tabla, lo que hace que la tabla no esté disponible para realizar consultas mientras se resincroniza. Para obtener más información, consulte the section called “Una o más de mis tablas de Amazon Redshift requieren una resincronización”.

Limitaciones de Aurora MySQL

- El clúster de base de datos de origen debe ejecutar una versión compatible de Aurora MySQL. Para obtener una lista de las versiones compatibles, consulte the section called “Integraciones sin ETL ”.

- Las integraciones sin ETL se basan en el registro binario de MySQL (binlog) para capturar los cambios en los datos en curso. No utilice el filtrado de datos basado en binlog, ya que puede provocar incoherencias entre los datos de las bases de datos de origen y de destino.
- Las integraciones sin ETL solo son compatibles con bases de datos configuradas para usar el motor de almacenamiento de InnoDB.
- No se admiten referencias de clave externas con actualizaciones de tablas predefinidas. En concreto, las reglas `ON DELETE` y `ON UPDATE` no son compatibles con las acciones `CASCADE`, `SET NULL` y `SET DEFAULT`. Si se intenta crear o actualizar una tabla con este tipo de referencias a otra tabla, se producirá un error en la tabla.
- Las transacciones XA realizadas en el clúster de base de datos de origen hacen que la integración entre en un estado de `Syncing`.

Limitaciones de Aurora PostgreSQL

- El clúster de base de datos de origen debe ejecutar una versión compatible de Aurora PostgreSQL. Para obtener una lista de las versiones compatibles, consulte the section called “Integraciones sin ETL ”.
- Si selecciona un clúster de base de datos de origen de Aurora PostgreSQL, debe especificar al menos un patrón de filtro de datos. Como mínimo, el patrón debe incluir una única base de datos (*`database-name`*`.*.*`) para la replicación en el almacén de destino. Para obtener más información, consulte the section called “Filtrado de datos para integraciones sin ETL”.
- Todas las bases de datos creadas en el clúster de base de datos de Aurora PostgreSQL de origen deben utilizar la codificación UTF-8.
- Al utilizar la partición declarativa, las particiones de la tabla se replicarán en Amazon Redshift. Sin embargo, la tabla particionada en sí no se replica en Amazon Redshift.
- No se admiten las transacciones bifásicas.
- Si elimina todas las instancias de base de datos de un clúster de base de datos que es el origen de una integración y, a continuación, vuelve a agregar una instancia de base de datos, la replicación se interrumpe entre los clústeres de origen y de destino.
- El clúster de base de datos de origen no puede utilizar la base de datos ilimitada de Aurora.
- Las claves principales son obligatorias en todas las tablas presentes en el filtro de datos. Todas las tablas que no tengan una clave principal pasarán al estado erróneo.

Limitaciones de Amazon Redshift

Para obtener una lista de limitaciones de Amazon Redshift relacionadas con las integraciones sin ETL, consulte Consideraciones al utilizar las integraciones sin ETL con Amazon Redshift de la Guía de administración de Amazon Redshift.

Amazon SageMaker AILimitaciones de Lakehouse

La siguiente es una limitación para las integraciones sin ETL de Amazon SageMaker AI Lakehouse.

- Los nombres de catálogo están limitados a 19 caracteres de longitud.

Cuotas

La cuenta tiene las siguientes cuotas relacionadas con las integraciones sin ETL de Aurora. Cada una de las cuotas se aplica a una sola región, a no ser que se especifique otra cosa.

Nombre	Predeterminado/a	Descripción
Integraciones	100	El número total de integraciones dentro de una Cuenta de AWS.
Integraciones por destino	50	El número de integraciones que envían datos a un único almacén de datos o almacén de lago de destino.
Integraciones por clúster de origen	5	La cantidad de integraciones que envían datos desde un solo clúster de base de datos de origen.

Además, el almacén de destino establece algunos límites en la cantidad de tablas permitidas en cada instancia de base de datos o nodo de clúster. Para obtener más información sobre cuotas y límites de Amazon Redshift, consulte Cuotas y límites de Amazon Redshift en la Guía de administración de Amazon Redshift.

Regiones admitidas

Las integraciones sin ETL de Aurora están disponibles en un subconjunto de Regiones de AWS. Para obtener una lista de las regiones admitidas, consulte the section called “Integraciones sin ETL ”.

Introducción a las integraciones sin ETL de Aurora

Antes de crear una integración sin ETL, configure el clúster de base de datos de Aurora y el almacén de datos con los parámetros y permisos necesarios. Durante la configuración, realizará los siguientes pasos:

1. Creación de un grupo de parámetros de clúster de base de datos personalizado.
2. Cree un clúster de base de datos de origen.
3. Cree un almacén de datos de destino para Amazon Redshift o cree un lakehouse de Amazon SageMaker AI de destino.

Una vez que haya completado estas tareas, continúe con the section called “Creación de integraciones sin ETL con Amazon Redshift” o the section called “Creación de integraciones sin ETL con un Amazon SageMaker Lakehouse”.

Puede utilizar los SDK de AWS para automatizar el proceso de configuración. Para obtener más información, consulte the section called “Configuración de una integración mediante los AWS SDK”.

> **Tip**
>
> Puede dejar que RDS complete estos pasos de configuración automáticamente mientras crea la integración, en lugar de hacerlos de forma manual. Para empezar inmediatamente a crear una integración, consulte the section called “Creación de integraciones sin ETL con Amazon Redshift”.

En el paso 3, puede optar por crear un almacén de datos de destino (paso 3a) o un Lakehouse de destino (paso 3b), según sus necesidades:

- Elija un almacén de datos si necesita capacidades de almacenamiento de datos tradicionales con análisis basados en SQL.

- Elija un lakehouse de Amazon SageMaker AI si necesita capacidades de machine learning y desea utilizar las características de lakehouse para la ciencia de datos y los flujos de trabajo de ML.

Crear un grupo de parámetros de clúster de base de datos personalizado

Las integraciones sin ETL de Aurora requieren valores específicos para los parámetros del clúster de base de datos que controlan la replicación. En concreto, Aurora MySQL requiere un binlog mejorado (`aurora_enhanced_binlog`) y Aurora PostgreSQL requiere replicación lógica mejorada (`aurora.enhanced_logical_replication`).

Para configurar el registro binario o la replicación lógica, primero debe crear un grupo de parámetros personalizado del clúster de base de datos y, a continuación, asociarlo al clúster de base de datos de origen.

Aurora MySQL (familia aurora-mysql):

- `aurora_enhanced_binlog=1`
- `binlog_backup=0`
- `binlog_format=ROW`
- `binlog_replication_globaldb=0`
- `binlog_row_image=full`
- `binlog_row_metadata=full`

Además, compruebe que el parámetro `binlog_transaction_compression` no esté establecido en `ON` y que el parámetro `binlog_row_value_options` no esté establecido en `PARTIAL_JSON`.

Para obtener más información sobre el binlog mejorado de Aurora MySQL, consulte the section called “Configuración del binlog mejorado”.

Aurora PostgreSQL (familia aurora-postgresql):

- `rds.logical_replication=1`
- `aurora.enhanced_logical_replication=1`
- `aurora.logical_replication_backup=0`
- `aurora.logical_replication_globaldb=0`

Al habilitar la replicación lógica mejorada (`aurora.enhanced_logical_replication`), siempre se escribirán todos los valores de las columnas en el registro de escritura anticipada (WAL), incluso si `REPLICA IDENTITY FULL` no está habilitada. Esto podría aumentar las IOPS del clúster de base de datos de origen.

> **Important**
>
> Si habilita o desactiva el parámetro del clúster de base de datos `aurora.enhanced_logical_replication`, la instancia de base de datos principal invalida todas las ranuras de replicación lógica. Esto detiene la replicación desde el origen hasta el destino y debe volver a crear los intervalos de replicación en la instancia de base de datos principal. Para evitar interrupciones, mantenga el estado de parámetro coherente durante la replicación.

Paso 2: seleccionar o crear un clúster de base de datos de origen

Tras crear un grupo de parámetros personalizado del clúster de base de datos, elija o cree un un clúster de base de datos de Aurora. Este clúster será el origen de la replicación de datos en el almacén de datos de destino. Puede especificar un clúster de base de datos que utilice instancias de base de datos o instancias de base de datos de Aurora Serverless v2 aprovisionadas. Para obtener instrucciones sobre cómo crear un clúster de base de datos, consulte the section called “Creación de un clúster de base de datos” o the section called “Creación de un clúster de bases de datos de Aurora Serverless v2”.

La base de datos debe ejecutar una versión de motor de base de datos compatible. Para obtener una lista de las versiones compatibles, consulte the section called “Integraciones sin ETL ”.

Al crear la base de datos, en Configuración adicional, cambie el grupo de parámetros de clúster de base de datos predeterminado por el grupo de parámetros personalizado que ha creado en el paso anterior.

> **Note**
>
> Si asocia el grupo de parámetros al clúster de base de datos después de haber creado el clúster, debe reiniciar la instancia de base de datos principal del clúster para aplicar los cambios y poder crear una integración sin ETL. Para obtener instrucciones, consulte the section called “Reinicio de un clúster o de una instancia de base de datos de Aurora”.

Paso 3a: crear un almacén de datos de destino

Tras crear el clúster de base de datos de origen, debe crear y configurar un almacén de datos de destino. El almacenamiento de datos debe cumplir los siguientes requisitos:

- Uso de un tipo de nodo RA3 con al menos dos nodos o Redshift sin servidor.
- Cifrado (si se utiliza un clúster aprovisionado). Para obtener más información, consulte Cifrado de base de datos de Amazon Redshift.

Para obtener instrucciones sobre cómo crear un almacenamiento de datos, consulte la sección Creación de un clúster para clústeres aprovisionados o Creación de un grupo de trabajo con un espacio de nombres para Redshift Serverless.

Activar la distinción entre mayúsculas y minúsculas en el almacén de datos

Para que la integración funcione, el parámetro de distinción entre mayúsculas y minúsculas (`enable_case_sensitive_identifier`) debe estar habilitado en el almacenamiento de datos. De forma predeterminada, la distinción entre mayúsculas y minúsculas está desactivada en todos los clústeres y grupos de trabajo sin servidor de Redshift suministrados.

Para activar la distinción entre mayúsculas y minúsculas, realice los siguientes pasos en función del tipo de almacén de datos:

- Clúster aprovisionado: para habilitar la distinción entre mayúsculas y minúsculas en un clúster aprovisionado, cree un grupo de parámetros personalizado con el parámetro `enable_case_sensitive_identifier` habilitado. A continuación, asocie el grupo de parámetros al clúster. Para obtener instrucciones, consulte la sección Administración de grupos de parámetros mediante la consola o Configuración de los valores de parámetros mediante la AWS CLI.

  > **Note**
  >
  > Recuerde reiniciar el clúster después de asociarlo el grupo de parámetros personalizado.

- Grupo de trabajo sin servidor: para habilitar la distinción entre mayúsculas y minúsculas en un grupo de trabajo sin servidor de Redshift, debe usar AWS CLI. Actualmente, la consola de Amazon Redshift no permite modificar los valores de los parámetros de Redshift sin servidor. Envíe la siguiente solicitud de update-workgroup:

```
aws redshift-serverless update-workgroup \
  --workgroup-name target-workgroup \
  --config-parameters
 parameterKey=enable_case_sensitive_identifier,parameterValue=true
```

No es necesario reiniciar un grupo de trabajo después de modificar los valores de los parámetros.

Configure la autorización para el almacenamiento de datos

Tras crear un almacenamiento de datos, debe configurar el clúster de base de datos de Aurora de origen como origen de integración autorizado. Para obtener instrucciones, consulte Configuración de la autorización para el almacenamiento de datos de Amazon Redshift.

Configuración de una integración mediante los AWS SDK

En lugar de configurar cada recurso manualmente, puede ejecutar el siguiente script de Python para configurar automáticamente los recursos necesarios. El ejemplo de código utiliza AWS SDK para Python (Boto3) para crear un clúster de base de datos de Amazon Aurora de origen y un almacén de datos de destino, cada uno de ellos con los valores de parámetros necesarios. A continuación, espera a que las bases de datos estén disponibles antes de crear una integración sin ETL entre ellas. Puede comentar diferentes funciones dependiendo de los recursos que necesite configurar.

Ejecute los siguientes comandos para asegurarse de que dispone de todas las dependencias necesarias:

```
pip install boto3
pip install time
```

En el script, si lo desea, modifique los nombres de los grupos de origen, destino y parámetros. La función final crea una integración denominada `my-integration` después de configurar los recursos.

Ejemplo de código Python

Aurora MySQL

```
import boto3
import time
```

```
# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora MySQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-mysql8.0',
        Description='For Aurora MySQL binary logging'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'aurora_enhanced_binlog',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_format',
                'ParameterValue': 'ROW',
                'ApplyMethod': 'pending-reboot'
            },
```

```
            {
                'ParameterName': 'binlog_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_image',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_metadata',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-mysql',
        EngineVersion='8.0.mysql_aurora.3.05.2',
        DatabaseName='myauroradb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-mysql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""
```

```
    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora MySQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
```

```
        Policy='''
        {
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
```

```
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Aurora PostgreSQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')
```

```
source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora PostgreSQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-postgresql16',
        Description='For Aurora PostgreSQL logical replication'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'rds.logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.enhanced_logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
```

```
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-postgresql',
        EngineVersion='16.4.aurora-postgresql',
        DatabaseName='mypostgresdb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-postgresql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora PostgreSQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
```

```
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
```

```
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
```

```
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Paso 3b: creación de un catálogo de AWS Glue para una integración sin ETL de Amazon SageMaker AI

Al crear una integración sin ETL con un Amazon SageMaker AI de Lakehouse, debe crear un catálogo administrado por AWS Glue en AWS Lake Formation. El catálogo de destino debe ser un catálogo administrado por Amazon Redshift. Para crear un catálogo administrado por Amazon Redshift, cree primero el rol vinculado al servicio de `AWSServiceRoleForRedshift`. En la consola de Lake Formation, agregue `AWSServiceRoleForRedshift` como administrador de solo lectura.

Para obtener más información sobre las tareas anteriores, consulte los siguientes temas.

- Para obtener información sobre la creación de un catálogo administrado por Amazon Redshift, consulte Creación de un catálogo administrado por Amazon Redshift en AWS Glue Data Catalog en la Guía para desarrolladores de AWS Lake Formation.
- Para obtener información sobre el role vinculado al servicio para Amazon Redshift, consulte Uso de roles vinculados a servicios para Amazon Redshift en la Guía de administración de Amazon Redshift.
- Para obtener información sobre los permisos de administrador de solo lectura para Lake Formation, consulte la referencia sobre las personas de Lake Formation y los permisos de IAM en la Guía para desarrolladores de AWS Lake Formation.

Configuración de los permisos para el catálogo de AWS Glue de destino

Antes de crear un catálogo de destino para la integración sin ETL, debe crear el rol de creación de destino de Lake Formation y el rol de transferencia de datos de AWS Glue. Utilice el rol de creación de destino de Lake Formation para crear el catálogo de destino. Al crear el catálogo de destino, ingrese el rol de transferencia de datos de Glue en el campo Rol de IAM de la sección Acceso desde motores.

Rol de creación de destino de Lake Formation

El rol de creación de destino debe ser un administrador de Lake Formation y requiere los siguientes permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lakeformation:RegisterResource",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutEncryptionConfiguration",
                "iam:PassRole",
                "glue:CreateCatalog",
                "glue:GetCatalog",
                "s3:PutBucketTagging",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "redshift-serverless:CreateNamespace",
                "s3:DeleteBucket",
                "s3:PutBucketVersioning",
                "redshift-serverless:CreateWorkgroup"
            ],
            "Resource": [
```

```
                "arn:aws:glue:*:111122223333:catalog",
                "arn:aws:glue:*:111122223333:catalog/*",
                "arn:aws:s3:::*",
                "arn:aws:redshift-serverless:*:111122223333:workgroup/*",
                "arn:aws:redshift-serverless:*:111122223333:namespace/*",
                "arn:aws:iam::111122223333:role/GlueDataCatalogDataTransferRole"
            ]
        }
    ]
}
```

El rol de creación de destino debe tener la siguiente relación de confianza.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::111122223333:user/Username"
          },
          "Action": "sts:AssumeRole"
        }
    ]
}
```

Rol de transferencia de datos de Glue

El rol de transferencia de datos de Glue es necesario para las operaciones del catálogo de MySQL y debe tener los siguientes permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataTransferRolePolicy",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "glue:GetCatalog",
                "glue:GetDatabase"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

El rol de transferencia de datos de Glue debe tener la siguiente relación de confianza.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "glue.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Siguientes pasos

Con un clúster de base de datos de Aurora de origen y un almacén de datos de destino de Amazon Redshift o Lakehouse de Amazon SageMaker AI, puede crear una integración sin ETL y datos de replicación. Para obtener instrucciones, consulte the section called “Creación de integraciones sin ETL con Amazon Redshift”.

Creación de integraciones sin ETL de Aurora con Amazon Redshift

Al crear una integración sin ETL de Aurora, debe especificar un clúster de base de datos de Aurora de origen y un almacenamiento de datos de Amazon Redshift de destino. También puede personalizar la configuración de cifrado y añadir etiquetas. Aurora crea una integración entre el clúster de base de datos de origen y su destino. Una vez que la integración esté activa, todos los datos que inserte en el clúster de base de datos de origen se replicarán en el destino configurado de Amazon Redshift.

Requisitos previos

Antes de crear una integración sin ETL, debe crear un clúster de base de datos de origen y un almacenamiento de datos de Amazon Redshift de destino. También debe permitir la réplica en el almacenamiento de datos añadiendo el clúster de base de datos como origen de integración autorizado.

Para obtener instrucciones para completar cada uno de estos pasos, consulte the section called “Introducción a las integraciones sin ETL”.

Permisos necesarios

Para crear una integración sin ETL se necesitan determinados permisos de IAM. Como mínimo, necesita permisos para realizar las siguientes acciones:

- Crear integraciones sin ETL para el clúster de base de datos de Aurora de origen.
- Ver y eliminar todas las integraciones sin ETL.
- Crear integraciones entrantes en el almacenamiento de datos de destino.

Las políticas de ejemplo siguiente muestran los permisos con privilegios mínimos necesarios para crear y administrar integraciones. Es posible que no necesite estos permisos exactos si su usuario o rol tiene permisos más amplios, como una política administrada `AdministratorAccess`.

> **Note**
>
> Los nombres de recurso de Amazon (ARN) de Redshift tienen el siguiente formato. Tenga en cuenta el uso de la barra diagonal (`/`) en lugar de dos puntos (`:`) antes del UUID del espacio de nombres sin servidor.
>
> - Clúster aprovisionado: `arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid`
> - Sin servidor: `arn:aws:redshift-serverless:{region}:{account-id}:namespace/namespace-uuid`

Política de ejemplo para el destino de Redshift

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CreateIntegration",
      "Effect": "Allow",
      "Action": [
        "rds:CreateIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:db:source-db",
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "DescribeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DescribeIntegrations"
      ],
      "Resource": [
      "arn:aws:rds:us-east-1:123456789012:integration:*"
  ]
    },
```

```
    {
      "Sid": "ChangeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DeleteIntegration",
        "rds:ModifyIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "AllowRedShiftIntegration",
      "Effect": "Allow",
      "Action": [
        "redshift:CreateInboundIntegration"
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:123456789012:namespace:namespace-uuid"
      ]
    }
  ]
}
```

Elegir un almacenamiento de datos de destino en una cuenta diferente

Si tiene previsto especificar un almacenamiento de datos de Amazon Redshift de destino distinto de una Cuenta de AWS, debe crear un rol que permita a los usuarios de la cuenta actual acceder a los recursos de la cuenta de destino. Para obtener más información, consulte Proporcionar acceso a un usuario de IAM a otra Cuenta de AWS propia.

El rol debe tener los siguientes permisos, que permiten al usuario ver los clústeres aprovisionados de Amazon Redshift y los espacios de nombres de Redshift sin servidor disponibles en la cuenta de destino.

Permisos necesarios y política de confianza

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

El rol debe tener la siguiente política de confianza, que especifica el ID de la cuenta de destino.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Para obtener instrucciones sobre cómo crear los roles, consulte Creación de un rol mediante políticas de confianza personalizadas.

Creación de integraciones sin ETL

Puede crear la integración sin ETL mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

> **Important**
>
> Las integraciones sin ETL no admiten operaciones de actualización o resincronización. Si encuentra problemas con una integración después de la creación, debe eliminarla y crear una nueva.

Consola de RDS

Para eliminar una integración sin ETL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación izquierdo, elija Integraciones sin ETL.
3. Elija Crear integración sin ETL.
4. En Identificador de la integración, introduzca un nombre para la integración. El nombre puede tener hasta 63 caracteres alfanuméricos y puede incluir guiones.

   > **Important**
   >
   > Los nombres de catálogo están limitados a 19 caracteres de longitud. Asegúrese de que el identificador de integración cumpla este requisito si se va a utilizar como nombre de catálogo.

5. Elija Siguiente.
6. En Origen, seleccione el clúster de base de datos de Aurora donde se originarán los datos.

   > **Note**
   >
   > RDS le avisa si los parámetros del clúster de base de datos no están configurados correctamente. Si aparece este mensaje, puede elegir la opción Corregir automáticamente o configurarlos manualmente. Para obtener instrucciones sobre cómo corregirlos manualmente, consulte the section called “Crear un grupo de parámetros de clúster de base de datos personalizado”.
   > Para modificar los parámetros del clúster de base de datos es necesario reiniciar. Para crear la integración, es necesario completar el reinicio y aplicar correctamente los nuevos valores de parámetros en el clúster.

7. (Opcional) Seleccione Personalizar las opciones de filtrado de datos y agregue filtros de datos a la integración. Puede usar filtros de datos para definir el alcance de la replicación en el almacenamiento de datos de destino. Para obtener más información, consulte the section called "Filtrado de datos para integraciones sin ETL".
8. Cuando el clúster de base de datos de origen esté configurado correctamente, seleccione Siguiente.
9. En Destino, haga lo siguiente:

    1. (Opcional) Para utilizar una cuenta diferente a la Cuenta de AWS para el destino de Amazon Redshift, elija Especificar una cuenta diferente. A continuación, introduzca el ARN del rol de IAM con permisos para mostrar sus almacenamientos de datos. Para obtener instrucciones para crear el rol de IAM, consulte the section called "Elegir un almacenamiento de datos de destino en una cuenta diferente".
    2. Para el almacenamiento de datos de Amazon Redshift, seleccione el destino para los datos replicados del clúster de base de datos de origen. Puede elegir un clúster de Amazon Redshift aprovisionado o un espacio de nombres Redshift sin servidor como destino.

    > **Note**
    >
    > RDS le avisa si la política de recursos o la configuración de distinción entre mayúsculas y minúsculas del almacenamiento de datos especificado no están configuradas correctamente. Si aparece este mensaje, puede elegir la opción Corregir automáticamente o configurarlas manualmente. Para obtener instrucciones sobre cómo corregirlas manualmente, consulte Activación de la distinción entre mayúsculas y minúsculas en el almacenamiento de datos y Configuración de la autorización para el almacenamiento de datos en la Guía de administración de Amazon Redshift.
    > Para modificar la distinción entre mayúsculas y minúsculas en un clúster de Redshift aprovisionado es necesario reiniciar. Para crear la integración, es necesario completar el reinicio y aplicar correctamente el nuevo valor del parámetro al clúster.
    > Si el origen y el destino seleccionados están en Cuentas de AWS diferentes, Amazon RDS no podrá corregir esta configuración automáticamente. Debe acceder a la otra cuenta y corregirla manualmente en Amazon Redshift.

10. Una vez que el almacenamiento de datos de destino esté configurado correctamente, seleccione Siguiente.

11. (Opcional) En Etiquetas, añada una o más etiquetas al trabajo de integración. Para obtener más información, consulte the section called “Etiquetado de los recursos de Aurora y RDS”.
12. En el caso del cifrado, especifique cómo desea que se cifra la integración. De forma predeterminada, RDS cifra todas las integraciones con una Clave propiedad de AWS. Para elegir una clave administrada por el cliente en su lugar, active Personalizar la configuración de cifrado y elija una clave de KMS para usarla en el cifrado. Para obtener más información, consulte the section called “Cifrado de recursos de Amazon Aurora”.

    Si lo desea, añada un contexto de cifrado. Para obtener más información, consulte Contexto de cifrado en la Guía para desarrolladores de AWS Key Management Service.

    > **Note**
    >
    > Amazon RDS añade los siguientes pares de contexto de cifrado (además de los que usted incluya):
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`
    >
    > Esto reduce el número total de pares que se pueden añadir (de 8 a 6) y contribuye al límite total de caracteres en la restricción de concesiones. Para obtener más información, consulte Uso de restricciones de concesiones en la Guía para desarrolladores de AWS Key Management Service.
13. Elija Siguiente.
14. Revise la configuración de integración y elija Crear integración sin ETL.

    Si se produce un error en la creación, consulte the section called “No puedo crear una integración sin ETL” para ver los pasos de solución de problemas.

La integración tiene un estado de `Creating` mientras se crea y el almacenamiento de datos de Amazon Redshift de destino tiene un estado de `Modifying`. Durante este tiempo, no puede consultar el almacenamiento de datos ni realizar ningún cambio de configuración en él.

Cuando la integración se crea correctamente, tanto el estado de la integración como el almacenamiento de datos de Amazon Redshift de destino cambian a `Active`.

AWS CLI

Para crear una integración sin ETL mediante la AWS CLI, utilice el comando create-integration con las siguientes opciones:

> **Note**
>
> Recuerde que los nombres de catálogo están limitados a 19 caracteres. Elija el nombre de la integración en consecuencia si se va a utilizar como nombre de catálogo.

- `--integration-name`: especifique un nombre para la integración.
- `--source-arn`: especifique el ARN de la del clúster de base de datos de Aurora que será el origen de la integración.
- `--target-arn`: especifique el ARN del almacenamiento de datos de Amazon Redshift que será el destino de la integración.

Example

Para Linux, macOS o Unix:

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

Para Windows:

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

API de RDS

Para crear una integración sin ETL mediante la API de Amazon RDS, utilice la operación `CreateIntegration` con los siguientes parámetros:

> **Note**
>
> Los nombres de catálogo están limitados a 19 caracteres. Asegúrese de que el parámetro IntegrationName cumpla este requisito si se va a utilizar como nombre de catálogo.

- `IntegrationName`: especifique un nombre para la integración.
- `SourceArn`: especifique el ARN de la del clúster de base de datos de Aurora que será el origen de la integración.
- `TargetArn`: especifique el ARN del almacenamiento de datos de Amazon Redshift que será el destino de la integración.

Cifrado de integraciones con una clave administrada por el cliente

Si especifica una clave de KMS personalizada en lugar de una Clave propiedad de AWS al crear una integración, la política de claves debe darle a la entidad principal del servicio Amazon Redshift acceso principal a la acción `CreateGrant`. Además, debe permitir al usuario actual realizar las acciones `DescribeKey` y `CreateGrant`.

En la siguiente política de muestra se demuestra cómo proporcionar los permisos necesarios en su política de claves. Incluye claves de contexto que sirven para reducir aún más el alcance de los permisos.

Política de claves de muestra

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
```

```
        },
        {
            "Sid": "Allows the Redshift service principal to add a grant to a KMS
key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
```

```
        },
        {
            "Sid": "Allows the current uer or role to retrieve information about
 a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Para obtener más información, consulte Creating a key policy en la Guía del desarrollador de AWS Key Management Service.

Siguientes pasos

Tras crear correctamente una integración sin ETL, debe crear una base de datos de destino dentro del clúster o grupo de trabajo de Amazon Redshift de destino. A continuación, puede empezar a agregar datos a al clúster de base de datos de Aurora de origen y a dirigirle consultas en Amazon Redshift. Para obtener instrucciones, consulte Creating destination databases in Amazon Redshift.

Creación de integraciones sin ETL de Aurora con un Amazon SageMaker Lakehouse

Cuando crea una integración sin ETL de Aurora con un Amazon SageMaker Lakehouse, debe especificar un clúster de base de datos de Aurora de origen y el catálogo administrado por AWS Glue de destino. También puede personalizar la configuración de cifrado y añadir etiquetas. Aurora crea una integración entre el clúster de base de datos de origen y su destino. Una vez que la integración esté activa, todos los datos que inserte en el clúster de base de datos de origen se replicarán en el destino configurado.

Requisitos previos

Antes de crear una integración sin ETL con un Amazon SageMaker Lakehouse, debe crear un clúster de base de datos de origen y un catálogo administrado por AWS Glue de destino. También

debe permitir la replicación en el catálogo agregando el clúster de base de datos como origen de integración autorizado.

Para obtener instrucciones para completar cada uno de estos pasos, consulte the section called "Introducción a las integraciones sin ETL".

Permisos necesarios

Se necesitan determinados permisos de IAM para crear una integración sin ETL con un Amazon SageMaker Lakehouse. Como mínimo, necesita permisos para realizar las siguientes acciones:

- Crear integraciones sin ETL para el clúster de base de datos de Aurora de origen.
- Ver y eliminar todas las integraciones sin ETL.
- Cree integraciones entrantes en el catálogo administrado por AWS Glue de destino.
- Acceda a los buckets de Amazon S3 que utiliza el catálogo administrado por AWS Glue.
- Utilice claves de AWS KMS para el cifrado si está configurado un cifrado personalizado.
- Registre recursos con Lake Formation.
- Incluya la política de recursos en el catálogo administrado por AWS Glue para autorizar las integraciones entrantes.

La política de ejemplo siguiente demuestra los permisos con privilegios mínimos necesarios para crear y administrar integraciones con un Amazon SageMaker Lakehouse. Es posible que no necesite estos permisos exactos si su usuario o rol tiene permisos más amplios, como una política administrada `AdministratorAccess`.

Además, debe configurar una política de recursos en el catálogo administrado por AWS Glue de destino para autorizar las integraciones entrantes. Use el comando de la AWS CLI siguiente para aplicar la política de recursos.

Comando de la AWS CLI de ejemplo para autorizar las integraciones entrantes en el catálogo de destino

```
aws glue put-resource-policy \
      --policy-in-json  '{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
```

```
        },
        "Action": [
            "glue:AuthorizeInboundIntegration"
        ],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"],
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:rds:region:account_id:db:source_name"
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "account_id"
        },
        "Action": ["glue:CreateInboundIntegration"],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"]
    }
    ]
}' \
      --region region
```

> **Note**
>
> Los nombres de recurso de Amazon (ARN) del catálogo de Glue tienen el siguiente formato:
>
> - Catálogo de Glue: `arn:aws:glue:{region}:{account-id}:catalog/catalog-name`

Elección de un catálogo administrado por AWS Glue de destino en una cuenta diferente

Si piensa especificar un catálogo administrado por AWS Glue de destino que está en otra Cuenta de AWS, debe crear un rol que permita a los usuarios de la cuenta actual acceder a los recursos de la cuenta de destino. Para obtener más información, consulte Proporcionar acceso a un usuario de IAM a otra Cuenta de AWS propia.

El rol debe tener los siguientes permisos, que permiten al usuario ver los catálogos de AWS Glue disponibles en la cuenta de destino.

Permisos necesarios y política de confianza

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:GetCatalog"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

El rol debe tener la siguiente política de confianza, que especifica el ID de la cuenta de destino.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::111122223333:root"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Para obtener instrucciones sobre cómo crear los roles, consulte Creación de un rol mediante políticas de confianza personalizadas.

Creación de integraciones sin ETL con un Amazon SageMaker Lakehouse

Puede crear una integración sin ETL con un Amazon SageMaker Lakehouse mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

> **Important**
>
> Las integraciones sin ETL con un Amazon SageMaker Lakehouse no admiten operaciones de actualización o resincronización. Si encuentra problemas con una integración después de la creación, debe eliminarla y crear una nueva.

Consola de RDS

Creación de una integración sin ETL con un Amazon SageMaker Lakehouse

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación izquierdo, elija Integraciones sin ETL.
3. Elija Crear integración sin ETL.
4. En Identificador de la integración, introduzca un nombre para la integración. El nombre puede tener hasta 63 caracteres alfanuméricos y puede incluir guiones.
5. Elija Siguiente.
6. En Origen, seleccione el clúster de base de datos de Aurora donde se originarán los datos.

> **Note**
>
> RDS le avisa si los parámetros del clúster de base de datos no están configurados correctamente. Si aparece este mensaje, puede elegir la opción Corregir automáticamente o configurarlos manualmente. Para obtener instrucciones sobre cómo corregirlos manualmente, consulte the section called “Crear un grupo de parámetros de clúster de base de datos personalizado”.
> Para modificar los parámetros del clúster de base de datos es necesario reiniciar. Para crear la integración, es necesario completar el reinicio y aplicar correctamente los nuevos valores de parámetros en el clúster.

7. (Opcional) Seleccione Personalizar las opciones de filtrado de datos y agregue filtros de datos a la integración. Puede usar filtros de datos para definir el alcance de la replicación en el Amazon SageMaker Lakehouse de destino. Para obtener más información, consulte the section called “Filtrado de datos para integraciones sin ETL”.
8. Cuando el clúster de base de datos de origen esté configurado correctamente, seleccione Siguiente.
9. En Destino, haga lo siguiente:

    1. (Opcional) Para utilizar una Cuenta de AWS diferente para el Amazon SageMaker Lakehouse de destino, elija Especificar una cuenta diferente. A continuación, ingrese el ARN de un rol de IAM con permisos para mostrar los catálogos de AWS Glue. Para obtener instrucciones para crear el rol de IAM, consulte the section called “Elección de un catálogo administrado por AWS Glue de destino en una cuenta diferente”.
    2. Para el catálogo de AWS Glue, seleccione el destino de los datos replicados del clúster de la base de datos de origen. Puede elegir un catálogo administrado por AWS Glue existente como destino.
    3. El rol de IAM de destino necesita permisos de descripción en el catálogo de destino y debe tener los siguientes permisos:

        JSON

        ```
        {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "VisualEditor0",
                    "Effect": "Allow",
                    "Action": "glue:GetCatalog",
                    "Resource": [
                        "arn:aws:glue:us-east-1:111122223333:catalog/*",
                        "arn:aws:glue:us-east-1:111122223333:catalog"
                    ]
                }
            ]
        }
        ```

        El rol de IAM de destino debe tener la siguiente relación de confianza:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. Debe conceder al rol de IAM de destino permisos de descripción para el catálogo administrado por AWS Glue de destino con el rol de administrador de Lake Formation creado en Paso 3b: creación de un catálogo de AWS Glue para una integración sin ETL de Amazon SageMaker AI.

> **Note**
>
> RDS le notificará si la política de recursos o la configuración del catálogo administrado por AWS Glue especificado no están configurados correctamente. Si aparece este mensaje, puede elegir la opción Corregir automáticamente o configurarlas manualmente. Si el origen y el destino seleccionados están en Cuentas de AWS diferentes, Amazon RDS no podrá corregir esta configuración automáticamente. Debe acceder a la otra cuenta y corregirla manualmente en SageMaker Unified Studio.

10. Una vez que el catálogo administrado por AWS Glue de destino esté configurado correctamente, elija Siguiente.
11. (Opcional) En Etiquetas, añada una o más etiquetas al trabajo de integración. Para obtener más información, consulte the section called “Etiquetado de los recursos de Aurora y RDS”.
12. En el caso del cifrado, especifique cómo desea que se cifra la integración. De forma predeterminada, RDS cifra todas las integraciones con una Clave propiedad de AWS. Para elegir una clave administrada por el cliente en su lugar, active Personalizar la configuración de cifrado y elija una clave de KMS para usarla en el cifrado. Para obtener más información, consulte the section called “Cifrado de recursos de Amazon Aurora”.

Si lo desea, añada un contexto de cifrado. Para obtener más información, consulte Contexto de cifrado en la Guía para desarrolladores de AWS Key Management Service.

> **Note**
>
> Amazon RDS añade los siguientes pares de contexto de cifrado (además de los que usted incluya):
>
> - `aws:glue:integration:arn` - `IntegrationArn`
> - `aws:servicename:id` - `glue`
>
> Esto reduce el número total de pares que se pueden añadir (de 8 a 6) y contribuye al límite total de caracteres en la restricción de concesiones. Para obtener más información, consulte Uso de restricciones de concesiones en la Guía para desarrolladores de AWS Key Management Service.

13. Elija Siguiente.
14. Revise la configuración de integración y elija Crear integración sin ETL.

    Si se produce un error en la creación, consulte the section called “Solución de problemas de integración sin ETL” para ver los pasos de solución de problemas.

La integración tiene un estado de `Creating` mientras se crea y el Amazon SageMaker Lakehouse de destino tiene un estado de `Modifying`. Durante este tiempo, no puede consultar el catálogo ni realizar ningún cambio de configuración en él.

Cuando la integración se crea correctamente, el estado de la integración y el Amazon SageMaker Lakehouse de destino cambian a `Active`.

AWS CLI

Para preparar un catálogo administrado por AWS Glue de destino para una integración sin ETL utilizando la AWS CLI, primero debe utilizar el comando create-integration-resource-property con las siguientes opciones:

- `--resource-arn`: especifique el ARN del catálogo administrado por AWS Glue que será el destino de la integración.

- `--target-processing-properties`: especifique el ARN del rol de IAM para acceder al catálogo administrado por AWS Glue de destino

```
aws glue create-integration-resource-property --region us-east-1
 --resource-arn arn:aws:glue:region:account_id:catalog/catalog_name \
 --target-processing-properties '{"RoleArn" : "arn:aws:iam::account_id:role/
TargetIamRole"}'
```

Para crear una integración sin ETL mediante un Amazon SageMaker Lakehouse, utilice la AWS CLI, use el comando create-integration con las siguientes opciones:

- `--integration-name`: especifique un nombre para la integración.
- `--source-arn`: especifique el ARN de la del clúster de base de datos de Aurora que será el origen de la integración.
- `--target-arn`: especifique el ARN del catálogo administrado por AWS Glue que será el destino de la integración.

Example

Para Linux, macOS o Unix:

```
aws rds create-integration \
    --integration-name my-sagemaker-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db \
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

Para Windows:

```
aws rds create-integration ^
    --integration-name my-sagemaker-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db ^
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

API de RDS

Para crear una integración sin ETL con Amazon SageMaker mediante la API de Amazon RDS, utilice la operación `CreateIntegration` con los siguientes parámetros:

> **Note**
>
> Los nombres de catálogo están limitados a 19 caracteres. Asegúrese de que el parámetro IntegrationName cumpla este requisito si se va a utilizar como nombre de catálogo.

- `IntegrationName`: especifique un nombre para la integración.
- `SourceArn`: especifique el ARN de la del clúster de base de datos de Aurora que será el origen de la integración.
- `TargetArn`: especifique el ARN del catálogo administrado por AWS Glue que será el destino de la integración.

Cifrado de integraciones con una clave administrada por el cliente

Si especifica una clave de KMS personalizada en lugar de una Clave propiedad de AWS al crear una integración con Amazon SageMaker, la política de claves debe proporcionar a la entidad principal de servicio SageMaker Unified Studio acceso a la acción `CreateGrant`. Además, debe permitir al usuario actual realizar las acciones `DescribeKey` y `CreateGrant`.

En la siguiente política de muestra se demuestra cómo proporcionar los permisos necesarios en su política de claves. Incluye claves de contexto que sirven para reducir aún más el alcance de los permisos.

Política de claves de muestra

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "EnablesIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
```

```
        },
        {
            "Sid": "GlueServicePrincipalAddGrant",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleAddGrantKMSKey",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
```

```
        {
            "Sid": "AllowsCurrentUserRoleRetrieveKMSKeyInformation",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Para obtener más información, consulte Creating a key policy en la Guía del desarrollador de AWS Key Management Service.

Siguientes pasos

Después de crear correctamente una integración sin ETL con Amazon SageMaker, puede comenzar a agregar datos a clúster de base de datos de Aurora de destino, y consultarlos en el Amazon SageMaker Lakehouse. Los datos se replicarán automáticamente y estarán disponibles para las cargas de trabajo de análisis y machine learning.

Filtrado de datos para integraciones sin ETL de Aurora

Las integraciones sin ETL de Aurora admiten el filtrado de datos, lo que le permite controlar qué datos se replican desde el clúster de base de datos de Aurora de origen al almacén de datos de destino. En lugar de replicar toda la base de datos, puede aplicar uno o más filtros para incluir o excluir selectivamente tablas específicas. Esto lo ayuda a optimizar el almacenamiento y el rendimiento de las consultas al garantizar que solo se transfieran los datos relevantes. Actualmente, el filtrado está limitado a los niveles de base de datos y tabla. No se admite el filtrado a nivel de columna y fila.

El filtrado de datos puede resultar útil cuando desee:

- Una determinadas tablas de dos o más clústeres de origen diferentes y no necesita datos completos del clúster.
- Ahorrar costos realizando análisis utilizando únicamente un subconjunto de tablas en lugar de una flota completa de bases de datos.

- Filtrar la información confidencial (como números de teléfono, direcciones o datos de tarjetas de crédito) de determinadas tablas.

Puede agregar filtros de datos a una integración sin ETL mediante la Consola de administración de AWS, la AWS Command Line Interface (AWS CLI) o la API de Amazon RDS.

Si la integración tiene un clúster aprovisionado como destino, el clúster debe tener la revisión 180 o uno posterior para usar el filtrado de datos.

Temas

- Formato de un filtro de datos
- Lógica de filtros
- Prioridad del filtro
- Ejemplos de Aurora MySQL
- Ejemplos de Aurora PostgreSQL
- Adición de filtros de datos a una integración
- Eliminación de filtros de datos de una integración

Formato de un filtro de datos

Puede definir varios filtros para una sola integración. Cada filtro incluye o excluye cualquier tabla de base de datos existente y futura que coincida con uno de los patrones de la expresión del filtro. Las integraciones sin ETL de Aurora utilizan la sintaxis de filtro Maxwell para el filtrado de datos.

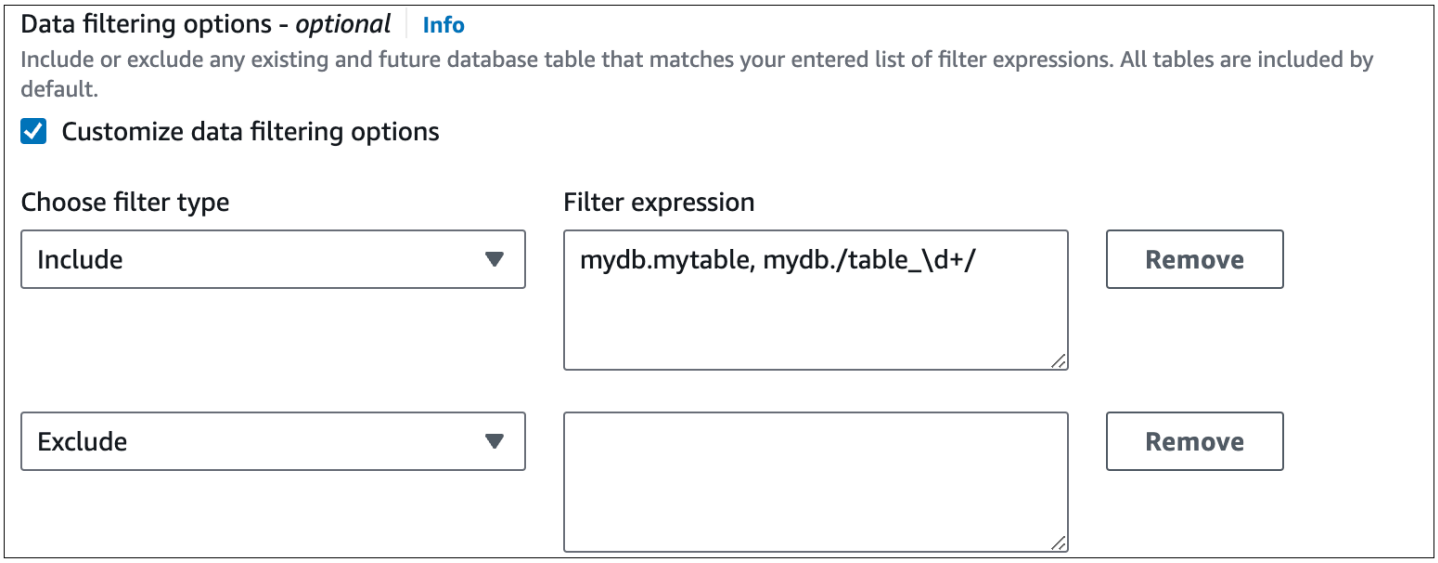
Cada filtro tiene los siguientes elementos:

Elemento	Descripción
Tipo de filtro	Un tipo de filtro `Include` incluye todas las tablas que coinciden con uno de los patrones de la expresión de filtro. Un tipo de filtro `Exclude` excluye todas las tablas que coinciden con uno de los patrones.

Elemento	Descripción
Expresión de filtro	Una lista separada por comas de patrones. Las expresiones deben usar la sintaxis de filtro Maxwell.
Pattern	Un patrón de filtro en el formato *database*.*table* para Aurora MySQL o *database*.*schema*.*table* para Aurora PostgreSQL. Puede especificar nombres literales o definir expresiones regulares. **Note** Para Aurora MySQL, se admiten expresiones regulares tanto en el nombre de la base de datos como de la tabla. Para Aurora PostgreSQL, las expresiones regulares solo se admiten en el nombre del esquema y la tabla, no en el nombre de la base de datos. No pueden incluir filtros en columnas ni listas de denegación. Una sola integración puede tener un máximo de 99 patrones en total. En la consola, puede introducir patrones dentro de una sola expresión de filtro o distribuirlos entre varias expresiones. Un único patrón no puede superar los 256 caracteres de longitud.

> **Important**
>
> Si selecciona un clúster de base de datos de origen de Aurora PostgreSQL, debe especificar al menos un patrón de filtro de datos. Como mínimo, el patrón debe incluir una única base de datos (*`database-name`*`.*.*`) para la replicación en el almacén de datos de destino.

En la imagen siguiente, se muestra la estructura de los filtros de datos de Aurora MySQL en la consola:



> **Important**
>
> No incluya información de identificación personal, confidencial o sensible en sus patrones de filtros.

Filtros de datos en la AWS CLI

Cuando se utiliza la AWS CLI para agregar un filtro de datos, la sintaxis es ligeramente diferente a la de la consola. Debe asignar un tipo de filtro (`Include` o `Exclude`) a cada patrón individualmente, por lo que no puede agrupar varios patrones en un mismo tipo de filtro.

Por ejemplo, en la consola puede agrupar los siguientes patrones separados por comas en una sola instrucción `Include`:

Aurora MySQL

```
mydb.mytable, mydb./table_\d+/
```

Aurora PostgreSQL

```
mydb.myschema.mytable, mydb.myschema./table_\d+/
```

Sin embargo, al utilizar la AWS CLI, el mismo filtro de datos debe tener el siguiente formato:

Aurora MySQL

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

Aurora PostgreSQL

```
'include: mydb.myschema.mytable, include: mydb.myschema./table_\d+/'
```

Lógica de filtros

Si no especifica ningún filtro de datos en la integración, Aurora asume un filtro predeterminado de `include:*.*`, que replica todas las tablas en el almacén de datos de destino. No obstante, si agrega al menos un filtro, la lógica predeterminada cambia a `exclude:*.*`, que excluye todas las tablas de forma predeterminada. Esto le permite definir explícitamente qué bases de datos y tablas se incluirán en la replicación.

Por ejemplo, si hace lo siguiente:

```
'include: db.table1, include: db.table2'
```

Aurora evalúa el filtro de la siguiente manera:

```
'exclude:*.*, include: db.table1, include: db.table2'
```

Por lo tanto, Aurora solo replica `table1` y `table2` de la base de datos llamada db en el almacén de datos de destino.

Prioridad del filtro

Aurora evalúa los filtros de datos en el orden que especifique. En la Consola de administración de AWS, procesa las expresiones de filtro de izquierda a derecha y de arriba abajo. Un segundo filtro o un patrón individual que siga al primero puede anularlo.

Por ejemplo, si el primer filtro es `Include books.stephenking`, solo incluye la tabla `stephenking` de la base de datos `books`. Sin embargo, si agrega un segundo filtro, `Exclude books.*`, este anulará el primer filtro. Esto evita que las tablas del índice de `books` se repliquen en el almacén de datos de destino.

Cuando especifica al menos un filtro, la lógica comienza con la suposición de `exclude:*.*` de forma predeterminada, lo que excluye automáticamente todas las tablas de la replicación. Como práctica recomendada, defina los filtros de más amplio a más específico. Comience con una o más instrucciones `Include` para especificar los datos que desea replicar; a continuación, agregue filtros `Exclude` para eliminar selectivamente ciertas tablas.

El mismo principio se aplica a los filtros que se definen mediante la AWS CLI. Aurora evalúa estos patrones de filtro en el orden en que los especifique, por lo que un patrón podría anular a otro especificado antes que él.

Ejemplos de Aurora MySQL

En los siguientes ejemplos, se muestra cómo funciona el filtrado de datos para las integraciones sin ETL de ejemplos de Aurora MySQL:

- Incluir todas las bases de datos y todas las tablas:

```
'include: *.*'
```

- Incluir todas las tablas en la base de datos `books`:

```
'include: books.*'
```

- Excluya cualquier tabla con el nombre `mystery`:

```
'include: *.*, exclude: *.mystery'
```

- Incluir dos tablas específicas en la base de datos `books`:

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- Incluya todas las tablas de la base de datos `books`, excepto las que contengan la subcadena `mystery`:

```
'include: books.*, exclude: books./.*mystery.*/'
```

- Incluya todas las tablas de la base de datos `books`, excepto las que comiencen por `mystery`:

```
'include: books.*, exclude: books./mystery.*/'
```

- Incluya todas las tablas de la base de datos `books`, excepto las que finalicen por `mystery`:

```
'include: books.*, exclude: books./.*mystery/'
```

- Incluya todas las tablas de la base de datos `books` que comiencen por `table_`, excepto la que se llama `table_stephen_king`. Por ejemplo, `table_movies` o `table_books` se replicaría, pero no `table_stephen_king`.

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

Ejemplos de Aurora PostgreSQL

En los siguientes ejemplos, se muestra cómo funciona el filtrado de datos para las integraciones sin ETL de Aurora PostgreSQL:

- Incluir todas las tablas en la base de datos `books`:

```
'include: books.*.*'
```

- Excluya cualquier tabla nombrada `mystery` en la base de datos `books`:

```
'include: books.*.*, exclude: books.*.mystery'
```

- Incluya una tabla dentro de la base de datos `books` en el esquema `mystery` y una tabla dentro de la base de datos `employee` en el esquema `finance`:

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- Incluya todas las tablas de la base de datos `books` y el esquema `science_fiction`, excepto las que contengan la subcadena `king`:

```
'include: books.science_fiction.*, exclude: books.*./.*king.*/
```

- Incluya todas las tablas de la base de datos `books`, excepto las que tengan un nombre de esquema que comience por `sci`:

```
'include: books.*.*, exclude: books./sci.*/.*'
```

- Incluya todas las tablas de la base de datos `books`, excepto las que estén en el esquema `mystery` y acaben en `king`:

```
'include: books.*.*, exclude: books.mystery./.*king/'
```

- Incluya todas las tablas de la base de datos `books` que comiencen por `table_`, excepto la que se llama `table_stephen_king`. Por ejemplo, `table_movies` en el esquema `fiction` y `table_books` en el esquema `mystery` se replican, pero no `table_stephen_king` en ninguno de los dos esquemas:

```
'include: books.*./table_.*/, exclude: books.*.table_stephen_king'
```

Adición de filtros de datos a una integración

Puede configurar el filtrado de datos mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

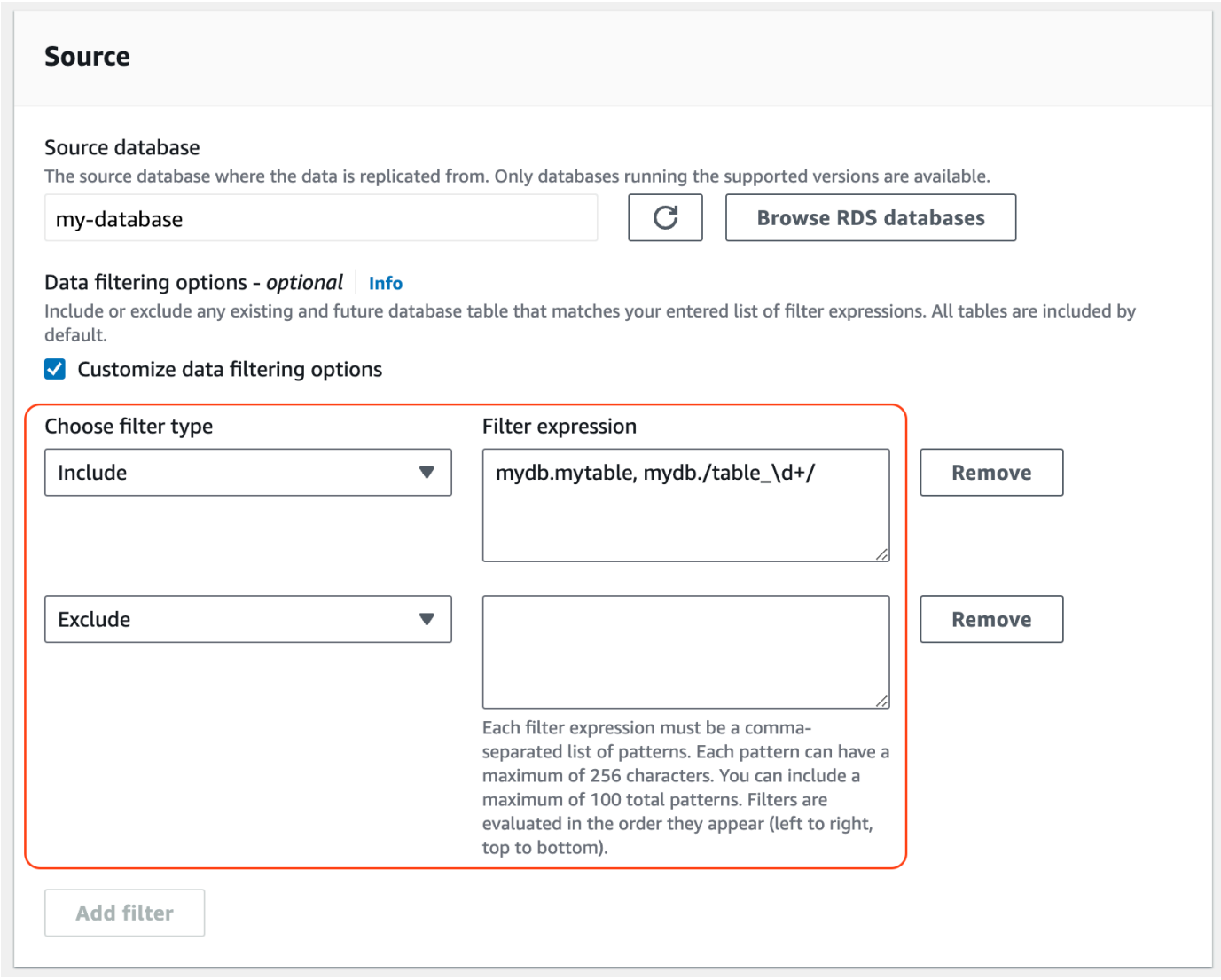
> **Important**
>
> Si agrega un filtro después de crear una integración, Aurora lo trata como si hubiera existido siempre. Elimina todos los datos del almacén de datos de destino que no coincidan con los nuevos criterios de filtrado y vuelve a sincronizar todas las tablas afectadas.

Consola de RDS

Adición de filtros de datos a una integración sin ETL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Integraciones sin ETL. Seleccione la integración a la que desea agregar filtros de datos y, a continuación, elija Modificar.
3. En Origen, agregue una o más instrucciones `Include` y `Exclude`.

   En la imagen siguiente, se muestra un ejemplo de filtros de datos para una integración de MySQL:



4. Cuando esté satisfecho con los cambios, elija Continuar y Guardar cambios.

AWS CLI

Para agregar filtros de datos a una integración sin ETL mediante la AWS CLI, llame al comando modify-integration. Además del identificador de integración, especifique el parámetro `--data-filter` con una lista separada por comas de filtros `Include` y `Exclude` Maxwell.

Example

En el siguiente ejemplo, se agregan patrones de filtro a `my-integration`.

Para Linux, macOS o Unix:

```
aws rds modify-integration \
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

Para Windows:

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

API de RDS

Para modificar una integración sin ETL mediante la API de RDS, llame a la operación ModifyIntegration. Especifique el identificador de integración y proporcione una lista separada por comas de patrones de filtro.

Eliminación de filtros de datos de una integración

Al eliminar un filtro de datos de una integración, Aurora vuelve a evaluar los filtros restantes como si el filtro eliminado nunca hubiera existido. A continuación, replica en el almacén de datos de destino cualquier dato excluido anteriormente que ahora cumpla los criterios. Esto desencadena una nueva sincronización de todas las tablas afectadas.

Cómo agregar datos en un clúster de base de datos de Aurora de origen y dirigirle consultas en Amazon Redshift

Para terminar de crear una integración sin ETL que replique los datos de Amazon Aurora en Amazon Redshift, debe crear una base de datos de destino en Amazon Redshift.

Primero, conéctese a su clúster o grupo de trabajo de Amazon Redshift y cree una base de datos con una referencia a su identificador de integración. A continuación, puede empezar a añadir datos al clúster de base de datos de Aurora de origen y ver la réplica en Amazon Redshift.

Temas

- Creación de bases de datos de destino en Amazon Redshift
- Añadir datos al clúster de base de datos de origen
- Consulta de los datos de en Amazon Redshift
- Diferencias de tipos de datos entre las bases de datos Aurora y Amazon Redshift

Creación de bases de datos de destino en Amazon Redshift

Antes de empezar a replicar datos en Amazon Redshift, debe crear una base de datos de destino en su almacén de datos de destino después de crear una integración. Esta base de datos de destino debe incluir una referencia al identificador de integración. También puede utilizar la consola de Amazon Redshift o el editor de consultas v2 para crear la base de datos.

Para obtener instrucciones sobre cómo crear una base de datos de destino, consulte Creación de una base de datos de destino en Amazon Redshift.

Añadir datos al clúster de base de datos de origen

Tras configurar la integración, puede añadir algunos datos al clúster de base de datos de Aurora que desee replicar en su almacenamiento de datos de Amazon Redshift.

> **Note**
>
> Existen diferencias entre los tipos de datos en , Amazon Aurora y Amazon Redshift. Para consultar una tabla de correspondencias de tipos de datos, consulte the section called “Diferencias de tipos de datos”.

Primero, conéctese al clúster de base datos de origen mediante el cliente MySQL o PostgreSQL que prefiera. Para obtener instrucciones, consulte the section called “Conexión a un clúster de base de datos”.

A continuación, cree una tabla e inserte una fila de datos de muestra.

> **Important**
>
> Asegúrese de que la tabla tenga una clave principal. De lo contrario, no se podrá replicar en el almacenamiento de datos de destino.

Las utilidades pg_dump y pg_restore de PostgreSQL crean inicialmente tablas sin una clave principal y, después, la agregan. Si utiliza una de estas utilidades, le recomendamos que cree primero un esquema y, a continuación, cargue los datos en un comando independiente.

MySQL

En el siguiente ejemplo se usa la utilidad MySQL Workbench.

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

PostgreSQL

En el siguiente ejemplo, se utiliza el terminal interactivo psql de PostgreSQL. Al conectarse al clúster, incluya la base de datos con nombre que especificó al crear la integración.

```
psql -h mycluster.cluster-123456789012.us-east-2.rds.amazonaws.com -p 5432 -U username
 -d named_db;

named_db=> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL,
 Author VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

named_db=> INSERT INTO books_table VALUES (1, "The Shining", "Stephen King", 1977,
 "Supernatural fiction");
```

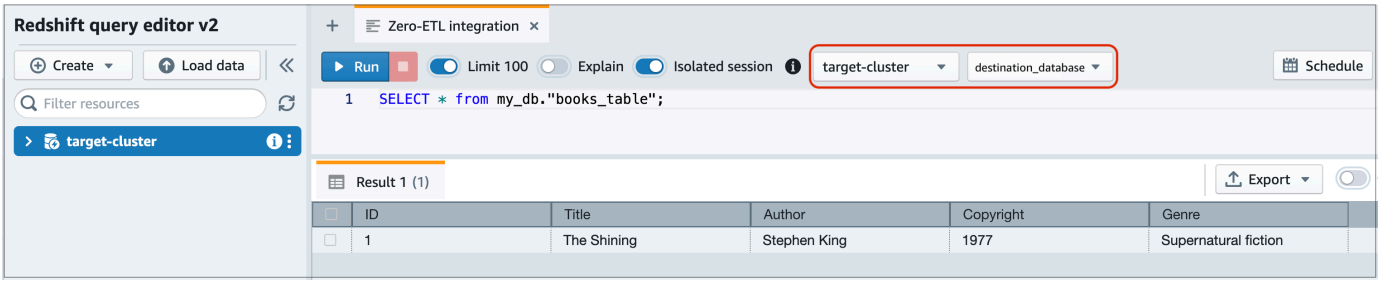
Consulta de los datos de en Amazon Redshift

Después de añadir datos en el clúster de base datos de Aurora, se replican en Amazon Redshift y ya se pueden consultar.

Consulta de datos replicados

1. Vaya a la consola de Amazon Redshift y seleccione el editor de consultas v2 en el panel de navegación izquierdo.
2. Conéctese a su clúster o grupo de trabajo y elija su base de datos de destino (la que creó a partir de la integración) en el menú desplegable (destination_database en este ejemplo). Para obtener instrucciones sobre cómo crear una base de datos de destino, consulte Creación de una base de datos de destino en Amazon Redshift.
3. Utilice una instrucción SELECT para consultar los datos. En este ejemplo, puede ejecutar el siguiente comando para seleccionar todos los datos de la tabla que creó en el clúster de base datos de Aurora de origen:

```
SELECT * from my_db."books_table";
```



- *my_db* es el nombre del esquema de la base de datos de Aurora. Esta opción solo es necesaria para las bases de datos MySQL.
- *books_table* es el nombre de la tabla Aurora.

También puede consultar los datos mediante un cliente de línea de comandos. Por ejemplo:

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                 Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+-----------------------+----------
+--------+
```

```
 1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
12192
```

> **Note**
>
> Para distinguir entre mayúsculas y minúsculas, utilice comillas dobles (" ") para los nombres de esquemas, tablas y columnas. Para obtener más información, consulte enable_case_sensitive_identifier.

Diferencias de tipos de datos entre las bases de datos Aurora y Amazon Redshift

En las siguientes tablas se muestran las asignaciones de un tipo de datos de Aurora MySQL o Aurora PostgreSQL a un tipo de datos de Amazon Redshift correspondiente. Actualmente, Amazon Aurora solo admite estos tipos de datos para integraciones sin ETL.

Si una tabla de la base de datos de origen incluye un tipo de datos no compatible, la tabla no se sincroniza y el destino de Amazon Redshift no puede utilizarla. La transmisión desde el origen al destino continúa, pero la tabla con el tipo de datos no admitido no está disponible. Para corregir la tabla y hacer que esté disponible en Amazon Redshift, debe revertir manualmente el cambio de ruptura y, a continuación, actualizar la integración ejecutando `ALTER DATABASE...INTEGRATION REFRESH`.

Temas

- Aurora MySQL
- Aurora PostgreSQL

Aurora MySQL

Tipo de datos de o Aurora MySQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
INT	INTEGER	Entero firmado de cuatro bytes	

Tipo de datos de o Aurora MySQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
SMALLINT	SMALLINT	Entero firmado de dos bytes	
TINYINT	SMALLINT	Entero firmado de dos bytes	
MEDIUMINT	INTEGER	Entero firmado de cuatro bytes	
BIGINT	BIGINT	Entero firmado de ocho bytes	
INT UNSIGNED	BIGINT	Entero firmado de ocho bytes	
TINYINT UNSIGNED	SMALLINT	Entero firmado de dos bytes	
MEDIUMINT UNSIGNED	INTEGER	Entero firmado de cuatro bytes	
BIGINT UNSIGNED	DECIMAL(20,0)	Numérico exacto de precisión seleccionable	
DECIMAL(p,s) = NUMERIC(p ,s)	DECIMAL (p,s)	Numérico exacto de precisión seleccionable	No se admiten precisiones superiores a 38 ni escalas superiores a 37
DECIMAL(p,s) UNSIGNED = NUMERIC(p,s) UNSIGNED	DECIMAL (p,s)	Numérico exacto de precisión seleccionable	No se admiten precisiones superiores a 38 ni escalas superiores a 37

Tipo de datos de o Aurora MySQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
FLOAT4/REAL	REAL	Número en coma flotante de precisión única	
FLOAT4/REAL SIN FIRMAR	REAL	Número en coma flotante de precisión única	
DOBLE/REAL/FLOAT8	DOUBLE PRECISION	Número en coma flotante de precisión doble	
DOBLE/REAL/FLOAT8 SIN FIRMAR	DOUBLE PRECISION	Número en coma flotante de precisión doble	
BIT(n)	VARBYTE (8)	Valor binario de longitud variable	
BINARY(n)	VARBYTE(n)	Valor binario de longitud variable	
VARBINARY (n)	VARBYTE(n)	Valor binario de longitud variable	
CHAR(n)	VARCHAR (n)	Valor de cadena de longitud variable	
VARCHAR (n)	VARCHAR (n)	Valor de cadena de longitud variable	

Tipo de datos de o Aurora MySQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
TEXT	VARCHAR(6535)	Valor de cadena de longitud variable de hasta 65535 bytes	
TINYTEXT	VARCHAR (255)	Valor de cadena de longitud variable de hasta 255 bytes	
MEDIUMTEXT	VARCHAR(6535)	Valor de cadena de longitud variable de hasta 65535 bytes	
LONGTEXT	VARCHAR(6535)	Valor de cadena de longitud variable de hasta 65535 bytes	
ENUM	VARCHAR(1020)	Valor de cadena de longitud variable de hasta 1020 bytes	
SET	VARCHAR(1020)	Valor de cadena de longitud variable de hasta 1020 bytes	
FECHA	FECHA	Fecha de calendario (año, mes, día)	

Tipo de datos de o Aurora MySQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
DATETIME	MARCA DE TIEMPO	Fecha y hora (sin zona horaria)	
TIMESTAMP(p)	MARCA DE TIEMPO	Fecha y hora (sin zona horaria)	
HORA	VARCHAR(18)	Valor de cadena de longitud variable de hasta 18 bytes	
YEAR	VARCHAR(4)	Valor de cadena de longitud variable de hasta 4 bytes	
JSON	SUPER	Datos o documento s semiestru cturados como valores	

Aurora PostgreSQL

Las integraciones sin ETL para Aurora PostgreSQL no admiten tipos de datos personalizados ni tipos de datos creados por extensiones.

> **Important**
>
> Las integraciones sin ETL de ETL con la característica de Amazon Redshift para Aurora PostgreSQL están en versión de vista previa. Tanto la documentación como la característica quedan sujetas a cambios. Puede utilizar esta característica solo en entornos de prueba y

no en entornos de producción. Para conocer los términos y condiciones de las versiones preliminares, consulte Betas y versiones preliminares en Términos de servicio de AWS.

Tipo de datos de Aurora PostgreSQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
bigint	BIGINT	Entero firmado de ocho bytes	
bigserial	BIGINT	Entero firmado de ocho bytes	
bit(n)	VARBYTE(n)	Valor binario de longitud variable	
bit varying(n)	VARBYTE(n)	Valor binario de longitud variable	
bit	VARBYTE(1.024.000)	Valor de cadena de longitud variable de hasta 1 024 000 bytes	
boolean	BOOLEAN	Booleano lógico (true/false)	
bytea	VARBYTE(1.024.000)	Valor de cadena de longitud variable de hasta 1 024 000 bytes	
character(n)	CHAR(n)	Cadena de caracteres de longitud fija	

Tipo de datos de Aurora PostgreSQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
character varying(n)	VARCHAR(6535)	Valor de cadena de longitud variable	
date	FECHA	Fecha de calendario (año, mes, día)	• No se admiten valores superiores a `9999-12-31` • No se admiten valores de B.C.
double precision	DOUBLE PRECISION	Números con coma flotante de precisión doble	No se admiten valores inferiores a los normales
integer	INTEGER	Entero firmado de cuatro bytes	
money	DECIMAL(203)	Importe de la divisa	
numeric(p,s)	DECIMAL (p,s)	Valor de cadena de longitud variable	• No se admiten valores `NaN` • No se admiten precisiones superiores a 38 ni escalas superiores a 37 • No se admite la escala negativa

Tipo de datos de Aurora PostgreSQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
real	REAL	Número en coma flotante de precisión única	
smallint	SMALLINT	Entero firmado de dos bytes	
smallserial	SMALLINT	Entero firmado de dos bytes	
serial	INTEGER	Entero firmado de cuatro bytes	
texto	VARCHAR(6535)	Valor de cadena de longitud variable de hasta 65 535 bytes	
time [(p)] [sin zona horaria]	VARCHAR(19)	Valor de cadena de longitud variable de hasta 4 bytes	No se admiten valores `Infinity` y `-Infinity`
time [(p)] con zona horaria	VARCHAR(22)	Valor de cadena de longitud variable de hasta 22 bytes	• No se admiten valores `Infinity` y `-Infinity`

Tipo de datos de Aurora PostgreSQL	Tipos de datos de Amazon Redshift	Descripción	Limitaciones
timestamp [(p)] [sin zona horaria]	MARCA DE TIEMPO	Fecha y hora (sin zona horaria)	• No se admiten valores `Infinity` y `-Infinity` • No se admiten valores superiores a `9999-12-31` • No se admiten valores de B.C.
timestamp [(p)] con zona horaria	TIMESTAMPTZ	Fecha y hora (con zona horaria)	• No se admiten valores `Infinity` y `-Infinity` • No se admiten valores superiores a `9999-12-31` • No se admiten valores de B.C.

Visualización y supervisión de las integraciones sin ETL de Aurora

Puede acceder a los detalles de una integración sin ETL de Amazon Aurora para ver su información de configuración y su estado actual. También puede supervisar el estado de la integración consultando vistas concretas del sistema en Amazon Redshift. Además, Amazon Redshift publica determinadas métricas relacionadas con la integración en Amazon CloudWatch, que puede ver en la consola de Amazon Redshift.

Temas

- Visualización de las integraciones
- Supervisión de las integraciones mediante tablas del sistema para Amazon Redshift
- Supervisión de las integraciones con Amazon EventBridge para Amazon Redshift

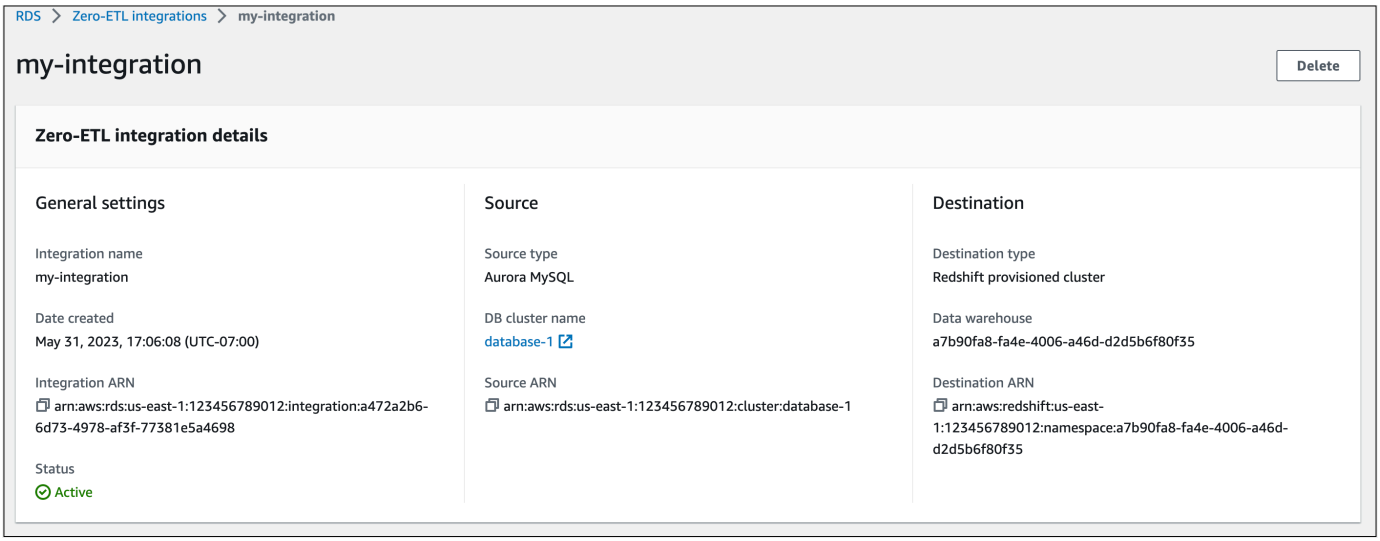
Visualización de las integraciones

Puede ver integraciones sin ETL de Aurora con la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para ver los detalles de una integración sin ETL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Integraciones sin ETL en el panel de navegación izquierdo.
3. Seleccione una integración para ver más detalles sobre ella, como el clúster de base de datos de origen y el almacenamiento de datos de destino.



Una integración puede tener los siguientes estados:

- `Creating`: la integración se está creando.
- `Active`: la integración envía datos transaccionales al almacenamiento de datos de destino.
- `Syncing`: la integración detectó un error recuperable y vuelve a almacenar los datos. Las tablas afectadas no están disponibles para consultas hasta que no terminen la resincronización.

- `Needs attention`: la integración detectó un evento o error que requiere la intervención manual para su resolución. Para solucionar el problema, siga las instrucciones del mensaje de error que aparece en la página de detalles de la integración.
- `Failed`: la integración detectó un evento o error irrecuperable que no se puede corregir. Debe eliminar y volver a crear la integración.
- `Deleting`: la integración se está eliminando.

AWS CLI

Para ver todas las integraciones sin ETL de la cuenta actual mediante la AWS CLI, utilice el comando describe-integrations y especifique la opción `--integration-identifier`.

Example

Para Linux, macOS o Unix:

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Para Windows:

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API de RDS

Para ver una integración sin ETL mediante la API de Amazon RDS, utilice la operación `DescribeIntegrations` con el parámetro `IntegrationIdentifier`.

Supervisión de las integraciones mediante tablas del sistema para Amazon Redshift

Amazon Redshift dispone de muchas tablas y vistas de sistema que contienen información acerca de cómo funciona el sistema. Puede consultar estas tablas y vistas de sistema de la misma forma que lo haría con cualquier otra tabla de bases de datos. Para obtener más información acerca de las vistas y tablas del sistema en Amazon Redshift, consulte la Referencia de las tablas y vistas de sistema en la Guía del desarrollador de bases de datos de Amazon Redshift.

Puede consultar las siguientes vistas y tablas del sistema para obtener información sobre las integraciones sin ETL de Aurora:

- SVV_INTEGRATION: proporciona detalles de configuración de sus integraciones.
- SVV_INTEGRATION_TABLE_STATE: describe el estado de cada tabla de una integración.
- SYS_INTEGRATION_TABLE_STATE_CHANGE: muestra los cambios de estado de las tablas de una integración.
- SYS_INTEGRATION_ACTIVITY: proporciona información sobre las ejecuciones de integración finalizadas.

Todas las métricas de Amazon CloudWatch provienen de Amazon Redshift. Para obtener información, consulte Métricas para integraciones sin ETL en la Guía de administración de Amazon Redshift. Actualmente, Amazon Aurora no publica ninguna métrica relacionada con la integración en CloudWatch.

Supervisión de las integraciones con Amazon EventBridge para Amazon Redshift

Amazon Redshift envía eventos relacionados con la integración a Amazon EventBridge. Para obtener una lista de los eventos y sus correspondientes ID, consulte la sección sobre notificaciones de eventos de integración sin ETL con Amazon EventBridge en la Guía de administración de Amazon Redshift.

Modificación de las integraciones sin ETL de Aurora

Solo puede modificar el nombre, la descripción y las opciones de filtrado de datos para una integración sin ETL en un almacén de datos compatible. No puede modificar la clave AWS KMS utilizada para cifrar la integración ni las bases de datos de origen o destino.

Si agrega un filtro a una integración existente, Aurora volverá a evaluar el filtro como si hubiera existido siempre. Elimina cualquier dato que se encuentre actualmente en el almacén de datos de destino y que no coincida con los nuevos criterios de filtrado. Si elimina un filtro de datos de una integración, este replica todos los datos que anteriormente no coincidían con los criterios de filtrado (pero que ahora sí) en el almacenamiento de datos de destino. Para obtener más información, consulte the section called “Filtrado de datos para integraciones sin ETL”.

Puede modificar la integración sin ETL mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS.

Consola de RDS

Modificación de una integración sin ETL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Integraciones sin ETL y, a continuación, elija la integración que desee modificar.
3. Elija Modificar y modifique cualquier configuración disponible.
4. Cuando haya realizado todos los cambios que desee, elija Modificar.

AWS CLI

Para modificar una integración sin ETL utilizando la AWS CLI, llame al comando modify-integration. Junto con `--integration-identifier`, especifique cualquiera de las siguientes opciones:

- `--integration-name`: especifique un nombre nuevo para la integración.
- `--description`: especifique una descripción nueva para la integración.
- `--data-filter`: especifique las opciones de filtrado de datos para la integración. Para obtener más información, consulte the section called “Filtrado de datos para integraciones sin ETL”.

Example

La siguiente solicitud modifica una integración existente.

Para Linux, macOS o Unix:

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

Para Windows:

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
```

```
    --integration-name my-renamed-integration
```

API de RDS

Para modificar una integración sin ETL mediante la API de RDS, llame a la operación ModifyIntegration. Especifique el identificador de integración y los parámetros que desee modificar.

Eliminación de las integraciones sin ETL de Aurora

Al eliminar una integración sin ETL, Amazon Aurora la elimina del clúster de base de datos de Aurora de origen. Los datos transaccionales no se eliminan de Amazon Aurora ni del destino de los análisis, pero Aurora no envía nuevos datos a Amazon Redshift o Amazon SageMaker.

Solo puede eliminar una integración si su estado es `Active`, `Failed`, `Syncing`, o `Needs attention`.

Puede eliminar las integraciones sin ETL mediante la Consola de administración de AWS, AWS CLI o la API de RDS.

Consola

Eliminación de una integración sin ETL

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Integraciones sin ETL en el panel de navegación.
3. Seleccione la integración sin ETL que desea eliminar.
4. Elija Acciones, Eliminar dominio y confirme la eliminación.

AWS CLI

Para eliminar una integración sin ETL, utilice el comando delete-integration y especifique la opción `--integration-identifier`.

Example

Para Linux, macOS o Unix:

```
aws rds delete-integration \
```

```
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Para Windows:

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API de RDS

Para eliminar una integración sin ETL mediante la API de Amazon RDS, utilice la operación `DeleteIntegration` con el parámetro `IntegrationIdentifier`.

Solución de problemas de integraciones sin ETL de Aurora con Amazon Redshift

Para comprobar el estado de una integración sin ETL, consulte la tabla del sistema SVV_INTEGRATION en Amazon Redshift. Si la columna `state` tiene un valor de `ErrorState`, significa que algo está mal. Para obtener más información, consulte the section called "Monitorización mediante tablas del sistema".

Utilice la siguiente información para solucionar problemas habituales relacionados con las integraciones sin ETL de Aurora con Amazon Redshift.

Temas

- No puedo crear una integración sin ETL
- Mi integración está atascada en un estado de Syncing
- Mis tablas no se replican en Amazon Redshift
- Una o más de mis tablas de Amazon Redshift requieren una resincronización

No puedo crear una integración sin ETL

Si no puede crear una integración sin ETL, asegúrese de que los siguientes elementos son correctos para su clúster de base de datos de origen:

- El clúster de base de datos de origen ejecuta Aurora MySQL versión 3.05 (compatible con MySQL 8.0.32) o posterior, o Aurora PostgreSQL (compatible con PostgreSQL 15.4 y

compatibilidad sin ETL). Para validar la versión del motor, elija la pestaña Configuración del clúster de base de datos y compruebe la versión del motor.

- Ha configurado correctamente los parámetros del clúster de base de datos. Si los parámetros requeridos están configurados incorrectamente o no están asociados al clúster, se produce un error en la creación. Consulte the section called “Crear un grupo de parámetros de clúster de base de datos personalizado”.

Además, asegúrese de que lo siguiente sea correcto para su almacenamiento de datos de destino:

- La distinción entre mayúsculas y minúsculas está activada. Consulte Turn on case sensitivity for your data warehouse.
- Ha añadido la entidad principal autorizado y el origen de integración correctos. Consulte Configuración de la autorización para el almacenamiento de datos de Amazon Redshift.
- El almacenamiento de datos está cifrado (si se trata de un clúster aprovisionado). Consulte Cifrado de la base de datos de Amazon Redshift.

Mi integración está atascada en un estado de **`Syncing`**

Es posible que su integración muestre continuamente el estado `Syncing` si cambia el valor de uno de los parámetros de base de datos necesarios.

Para solucionarlo, compruebe los valores de los parámetros del grupo de parámetros asociado al clúster de base de datos de origen y asegúrese de que coincidan con los valores requeridos. Para obtener más información, consulte the section called “Crear un grupo de parámetros de clúster de base de datos personalizado”.

Si modifica algún parámetro, asegúrese de reiniciar el clúster de base de datos para aplicar los cambios.

Mis tablas no se replican en Amazon Redshift

Si no ve reflejadas una o varias tablas en Amazon Redshift, puede ejecutar el siguiente comando para volver a sincronizarlas:

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

Para obtener más información, consulte ALTER DATABASE en la Referencia de SQL de Amazon Redshift.

Es posible que los datos no se estén replicando porque una o varias de las tablas de origen no tienen una clave principal. El panel de supervisión de Amazon Redshift muestra el estado de estas tablas como `Failed` y el estado de la integración sin ETL global cambia a `Needs attention`. Para resolver este problema, puede identificar una clave existente en la tabla que pueda convertirse en clave principal o puede añadir una clave principal sintética. Para obtener soluciones detalladas, consulte los siguientes recursos:

- Handle tables without primary keys while creating Amazon Aurora MySQL or Amazon RDS for MySQL zero-ETL integrations with Amazon Redshift
- Handle tables without primary keys while creating Amazon Aurora PostgreSQL zero-ETL integrations with Amazon Redshift

Una o más de mis tablas de Amazon Redshift requieren una resincronización

La ejecución de algunos comandos en el clúster de base de datos de origen puede requerir que las tablas se vuelvan a sincronizar. En estos casos, la vista del sistema SVV_INTEGRATION_TABLE_STATE muestra un `table_state` de `ResyncRequired`, lo que significa que la integración debe volver a cargar por completo los datos de esa tabla de MySQL a Amazon Redshift.

Cuando la tabla comienza a resincronizarse, entra en un estado de `Syncing`. No es necesario realizar ninguna acción manual para volver a sincronizar una tabla. Mientras se vuelven a sincronizar los datos de la tabla, no puede acceder a ellos en Amazon Redshift.

A continuación se muestran algunos ejemplos de operaciones que pueden poner una tabla en estado `ResyncRequired` y las posibles alternativas que se pueden considerar.

Operación	Ejemplo	Alternativa
Añadir una columna a una posición específica	`ALTER TABLE table_name ADD COLUMN column_name INTEGER NOT NULL first;`	Amazon Redshift no admite la adición de

Operación	Ejemplo	Alternativa
		columnas en posiciones específicas mediante las palabras clave `first` o `after`. Si el orden de las columnas de la tabla de destino no es crucial, añada la columna al final de la tabla con un comando más sencillo: `ALTER TABLE table_name ADD COLUMN column_name column_type ;`

Operación	Ejemplo	Alternativa

Añadir una columna de marca temporal con la opción predeterminada `CURRENT_TIMESTAMP`	`ALTER TABLE table_name`
	`  ADD COLUMN column_name  TIMESTAMP`
	`  NOT NULL DEFAULT CURRENT_TIMESTAMP;`
		Aurora MySQL calcula el valor `CURRENT_TIMESTAMP` de las filas de la tabla existentes y no se puede simular en Amazon Redshift sin una resincronización completa de los datos de la tabla.
		
		Si es posible, cambie el valor predeterminado a una constante literal, por ejemplo, `2023-01-01 00:00:15` para evitar la latencia en la disponibilidad de la tabla.

Operación	Ejemplo	Alternativa
Realizar operaciones en varias columnas con un solo comando	`ALTER TABLE table_name ADD COLUMN column_1, RENAME COLUMN column_2 TO column_3;`	Considere la posibilidad de dividir el comando en dos operaciones distintas ADD y RENAME que no requerirá resincronización.

Uso de Aurora Serverless v2

Aurora Serverless v2 es una configuración de escalado automático bajo demanda para Amazon Aurora.Aurora Serverless v2 ayuda a automatizar los procesos de supervisión de la carga de trabajo y ajustar la capacidad de las bases de datos. La capacidad se ajusta automáticamente en función de la demanda de la aplicación. Solo se le cobrará por los recursos que consuman los clústeres de base de datos. Por lo tanto, Aurora Serverless v2 puede ayudarle a mantenerse dentro del presupuesto y evitar pagar los recursos de computación que no utiliza.

Este tipo de automatización es especialmente valioso para bases de datos de multitenencia, bases de datos distribuidas, sistemas de desarrollo y pruebas y otros entornos con cargas de trabajo muy variables e impredecibles.

Temas

- Casos de uso de Aurora Serverless v2
- Ventajas de Aurora Serverless v2
- Cómo funciona Aurora Serverless v2
- Requisitos y limitaciones para Aurora Serverless v2
- Creación de un clúster de base de datos que utiliza Aurora Serverless v2
- Administración de clústeres de bases de datos de Aurora Serverless v2
- Rendimiento y escalado para Aurora Serverless v2
- Escalado a cero ACU con pausa y reanudación automáticas para Aurora Serverless v2
- Migración a Aurora Serverless v2

Casos de uso de Aurora Serverless v2

Aurora Serverless v2 admite muchos tipos de cargas de trabajo de bases de datos Estos comprenden, entre otros, entornos de desarrollo y pruebas, sitios web y aplicaciones que tienen cargas de trabajo impredecibles y las aplicaciones más exigentes y esenciales para la empresa que requieren una gran escala y disponibilidad.

Aurora Serverless v2 es especialmente útil para los siguientes casos de uso:

- Cargas de trabajo impredecibles: cuando ejecuta cargas de trabajo diarias que tienen un aumento de actividad repentino e impredecible. Un ejemplo de ello sería un sitio dedicado al tráfico,

que experimenta un aumento repentino de la actividad cuando comienza a llover. Otro caso sería un sitio de comercio electrónico en el que aumenta el tráfico cuando se ofrecen ventas o promociones especiales. Con Aurora Serverless v2, la capacidad de la base de datos se escala automáticamente para satisfacer las necesidades de los picos de carga de la aplicación y vuelve a disminuir cuando termina el aumento de actividad. Con Aurora Serverless v2, ya no tiene que aprovisionar la capacidad para los picos ni para la carga promedio. Puede especificar un límite de capacidad superior para gestionar la peor situación. Esa capacidad no se utilizaría a menos que fuera necesario.

La granularidad de la reducción horizontal de Aurora Serverless v2 le ayuda a adaptar la capacidad a las necesidades de su base de datos. En un clúster aprovisionado, para el escalado vertical hay que añadir una nueva instancia de base de datos. En un clúster de Aurora Serverless v1, para el escalado vertical hay que duplicar el número de unidades de capacidad (ACU) de Aurora para el clúster (por ejemplo, de 16 a 32 o de 32 a 64). En cambio, Aurora Serverless v2 puede añadir media ACU cuando solo se necesite un poco más de capacidad. Puede añadir 0,5, 1, 1,5, 2 o media ACU adicional en función de la capacidad adicional necesaria para gestionar un aumento de la carga de trabajo. Además, puede eliminar 0,5, 1, 1,5, 2 o media ACU adicional cuando la carga de trabajo disminuya y esa capacidad ya no se necesite.

- Aplicaciones de varios inquilinos: con Aurora Serverless v2, no es preciso administrar individualmente la capacidad de la base de datos para cada aplicación de la flota. Aurora Serverless v2 administra la capacidad de la base de datos individual por usted.

  Puede crear un clúster para cada inquilino. De esta forma, puede utilizar funciones como la clonación, la restauración de instantáneas y las bases de datos globales de Aurora para mejorar la alta disponibilidad y la recuperación de desastres según corresponda para cada inquilino.

  Cada inquilino puede tener períodos específicos de actividad e inactividad según la hora del día, la época del año, los eventos promocionales, etc. Cada clúster puede tener un amplio rango de capacidad. De esta forma, los clústeres con poca actividad incurren en los cargos mínimos de instancias de base de datos. Cualquier clúster puede escalarse de forma vertical rápidamente para gestionar períodos de actividad elevada.
- Aplicaciones nuevas: implementa una nueva aplicación y no está seguro del tamaño de instancia de base de datos que necesita. Con Aurora Serverless v2, puede configurar un clúster con una o varias instancias de base de datos y hacer que esta última se escale automáticamente de acuerdo con los requisitos de capacidad de la aplicación.
- Aplicaciones de uso mixto: supongamos que tiene una aplicación de procesamiento de transacciones en línea (OLTP), pero periódicamente experimenta picos en el tráfico de consultas.

Si especifica niveles de promoción para las instancias de base de datos de Aurora Serverless v2 en un clúster, puede configurar el clúster para que las instancias de base de datos del lector se puedan escalar independientemente de la instancia de base de datos de escritor para gestionar la carga adicional. Cuando el pico de uso disminuye, las instancias de base de datos del lector se escalan de nuevo para adaptarlas a la capacidad de la instancia de base de datos del escritor.

- Planificación de capacidad: supongamos que normalmente ajusta la capacidad de la base de datos o verifica la capacidad de la base de datos óptima para la carga de trabajo modificando las clases de instancias de base de datos de todas las instancias de base de datos de un clúster. Con Aurora Serverless v2, puede evitar esta sobrecarga administrativa. Para determinar la capacidad mínima y máxima apropiadas, puede ejecutar la carga de trabajo y comprobar cuánto se escalan realmente las instancias de base de datos.

  Puede modificar las instancias de base de datos existentes de aprovisionadas a Aurora Serverless v2 o de Aurora Serverless v2 a aprovisionadas. En estos casos, no es necesario crear un clúster nuevo o una instancia de base de datos nueva.

  Con una base de datos global de Aurora, puede que no necesite tanta capacidad para los clústeres secundarios como en el clúster principal. Puede usar instancias de base de datos de Aurora Serverless v2 en los clústeres secundarios. De esta forma, la capacidad del clúster se puede escalar verticalmente si se promueve una región secundaria que se hace cargo de la carga de trabajo de la aplicación.
- Desarrollo y pruebas: además de ejecutar las aplicaciones más exigentes, también puede utilizarlas en entornosAurora Serverless v2 de desarrollo y pruebas. Con Aurora Serverless v2, puede crear instancias de base de datos de con una capacidad mínima en lugar de utilizar clases de instancias ampliables de base de datos db.t*. Puede establecer la capacidad máxima lo suficientemente alta como para que esas instancias de base de datos puedan seguir ejecutando cargas de trabajo sustanciales sin quedarse sin memoria. Cuando la base de datos no está en uso, todas las instancias de base de datos se escalan verticalmente para evitar cargos innecesarios.

> **Tip**
>
> Para que sea cómodo usar Aurora Serverless v2 en entornos de desarrollo y pruebas, Consola de administración de AWS proporciona el acceso directo Easy create (Creación sencilla) para crear un nuevo clúster. Si elige la opción Dev/Test (Desarrollo/pruebas), Aurora crea un clúster con una instancia de base de datos de Aurora Serverless v2 y un rango de capacidad típico de un sistema de desarrollo y prueba.

Uso de Aurora Serverless v2 para cargas de trabajo aprovisionadas existentes

Supongamos que ya tiene una aplicación Aurora en ejecución en un clúster aprovisionado. Puede comprobar cómo funcionaría la aplicación con Aurora Serverless v2 añadiendo una o varias instancias de base de datos de Aurora Serverless v2 al clúster existente como instancias de base de datos del lector. Puede comprobar con qué frecuencia se escalan vertical y horizontalmente las instancias de base de datos del lector. Puede utilizar el mecanismo de conmutación por error de Aurora para promover una instancia de base de datos de Aurora Serverless v2 para que sea el escritor y comprobar cómo gestiona la carga de trabajo de lectura y escritura. De esta forma, puede realizar el cambio con un tiempo de inactividad mínimo y sin cambiar el punto de conexión que utilizan las aplicaciones cliente. Para obtener más información sobre el procedimiento para convertir los clústeres existentes a Aurora Serverless v2, consulte Migración a Aurora Serverless v2.

Ventajas de Aurora Serverless v2

Aurora Serverless v2 está destinado a cargas de trabajo variables o “con picos”. Con cargas de trabajo tan impredecibles, podría tener dificultades para planificar cuándo cambiar la capacidad de la base de datos. También podría tener problemas para realizar cambios de capacidad con la suficiente rapidez mediante los mecanismos conocidos, como añadir instancias de base de datos o cambiar clases de instancias de base de datos. Aurora Serverless v2 proporciona las siguientes ventajas para ayudar con estos casos de uso:

- Administración de la capacidad más sencilla que con la aprovisionada: Aurora Serverless v2 reduce el esfuerzo de planificar los tamaños de las instancias de base de datos y cambiar el tamaño de las instancias de base de datos a medida que cambia la carga de trabajo. También reduce el esfuerzo para mantener una capacidad uniforme para todas las instancias de base de datos en un clúster.
- Escalado más rápido y sencillo durante períodos de alta actividad: Aurora Serverless v2 escala la capacidad de memoria y de computación en función de las necesidades, sin interrumpir las transacciones del cliente ni de la carga de trabajo general. La capacidad de utilizar instancias de base de datos de lectores con Aurora Serverless v2 le ayuda a aprovechar el escalado horizontal además del escalado vertical. La capacidad de utilizar las bases de datos globales de Aurora le permite repartir la carga de trabajo de lectura de Aurora Serverless v2 entre varias Regiones de AWS. Esta capacidad es más práctica que los mecanismos de escalado de los clústeres aprovisionados. También es un método más rápido y granular que las capacidades de escalado de Aurora Serverless v1.

- Rentable durante períodos de baja actividad: Aurora Serverless v2 le ayuda a evitar el sobreaprovisionamiento de las instancias de base de datos.Aurora Serverless v2 añade recursos en incrementos granulares cuando se escalan verticalmente las instancias de base de datos. Solo paga por los recursos de bases de datos que consume. El uso de los recursos de Aurora Serverless v2 se mide por segundo. De esta forma, cuando una instancia de base de datos se reduce, la reducción del uso de recursos se registra de inmediato.
- Mayor paridad de características con aprovisionamiento: puede utilizar muchas características de Aurora con Aurora Serverless v2 que no están disponibles para Aurora Serverless v1. Por ejemplo, con Aurora Serverless v2 puede utilizar instancias de base de datos de lectores, bases de datos globales, autenticación de base de datos de AWS Identity and Access Management (IAM) y Performance Insights. También puede utilizar muchos más parámetros de configuración que con Aurora Serverless v1.

  En particular, con Aurora Serverless v2 puede aprovechar las siguientes características de los clústeres aprovisionados:

  - Instancias de base de datos:Aurora Serverless v2 puede aprovechar las instancias de base de datos de lector para escalar horizontalmente. Cuando un clúster contiene una o más instancias de base de datos de lectores, el clúster puede conmutar por error inmediatamente en caso de que haya problemas con la instancia de base de datos del escritor. Esta es una capacidad que no está disponible con Aurora Serverless v1.
  - Clústeres multi-AZ: puede distribuir las instancias de base de datos de Aurora Serverless v2 de un clúster entre varias zonas de disponibilidad (AZ). La configuración de un clúster Multi-AZ ayuda a garantizar la continuidad del negocio incluso en los raros casos en los que hay problemas que afectan a una AZ completa. Esta es una capacidad que no está disponible con Aurora Serverless v1.
  - Bases de datos globales: puede usar Aurora Serverless v2 en combinación con las bases de datos globales de Aurora para crear copias de solo lectura adicionales del clúster en otras Regiones de AWS para fines de recuperación de desastres.
  - Proxy RDS: puede usar el proxy de Amazon RDS para permitir a las aplicaciones agrupar y compartir conexiones de base de datos para mejorar su capacidad de escala.
- Escalado más rápido, más granular y menos disruptivo que con Aurora Serverless v1: Aurora Serverless v2 puede escalarse más rápido. El escalado puede cambiar la capacidad en tan solo 0,5 ACU, en lugar de duplicar o reducir a la mitad el número de ACU. El escalado suele producirse sin interrumpir el procesamiento. El escalado no implica un evento que deba tener en cuenta, como ocurre con Aurora Serverless v1. El escalado puede realizarse mientras se ejecutan las

instrucciones SQL y las transacciones están abiertas, sin necesidad de esperar a que haya un punto de silencio.

Cómo funciona Aurora Serverless v2

La siguiente información general describe cómo funciona Aurora Serverless v2.

Temas

- Información general de Aurora Serverless v2
- Configuraciones para clústeres de base de datos de Aurora
- Capacidad de Aurora Serverless v2
- Escalado en Aurora Serverless v2
- Aurora Serverless v2 y alta disponibilidad
- Aurora Serverless v2 y almacenamiento
- Parámetros de configuración de los clústeres de Aurora

Información general de Aurora Serverless v2

Amazon Aurora Serverless v2 va bien para las cargas de trabajo más exigentes y muy variables. Por ejemplo, el uso de la base de datos puede ser intensivo durante un corto periodo de tiempo, seguido de largos periodos de poca actividad o de ninguna actividad en absoluto. Ejemplos de ello son sitios web de venta minorista, juegos o deportes con eventos promocionales periódicos y bases de datos que generan informes cuando se necesitan. Otros son entornos de desarrollo y pruebas y nuevas aplicaciones en las que el uso podría aumentar rápidamente. Para casos como estos y muchos otros, la configuración correcta de la capacidad por anticipado no siempre es posible con el modelo aprovisionado. También puede resultar en costos más elevados si se aprovisiona en exceso y tiene capacidad que no utiliza.

En cambio, los clústeres aprovisionados de Aurora van bien para cargas de trabajo estables. Los clústeres aprovisionados le permiten elegir una clase de instancia de base de datos que tenga una cantidad predefinida de memoria, capacidad de CPU, ancho de banda de E/S, etc. Si la carga de trabajo cambia, modificará manualmente la clase del escritor y los lectores. El modelo aprovisionado funciona bien cuando se puede ajustar con anticipación la capacidad de los patrones de consumo esperados y se aceptan interrupciones breves mientras cambia la clase del escritor y los lectores del clúster.

Aurora Serverless v2 se ha diseñado desde cero para admitir clústeres de base de datos sin servidor escalables al instante. Aurora Serverless v2 se ha diseñado para proporcionar el mismo grado de seguridad y aislamiento que con escritores y lectores aprovisionados. Estos aspectos son cruciales en entornos de nube sin servidor multitenant. El mecanismo de escalado dinámico tiene muy poca sobrecarga para que pueda responder rápidamente a los cambios en la carga de trabajo de la base de datos. También es lo suficientemente potente como para satisfacer los incrementos drásticos en la demanda de procesamiento.

Con Aurora Serverless v2 podrá crear un clúster de bases de datos Aurora sin quedar sujeto a una capacidad de base de datos específica para escritor y lector. El usuario especifica el rango mínimo y máximo de capacidad. Aurora escala cada instancia Aurora Serverless v2 de escritura o lectura en el clúster dentro de ese rango de capacidad. Usar un clúster Multi-AZ en el que cada escritor o lector puede escalar dinámicamente permite aprovechar el escalado dinámico y de una alta disponibilidad.

Aurora Serverless v2 escala los recursos de base de datos de forma automática en función de las especificaciones de capacidad mínima y máxima. El escalado es rápido, ya que la mayoría de las operaciones de eventos de escalado mantienen al escritor o al lector en el mismo host. En los raros casos en que una instancia Aurora Serverless v2 de escritura o lectura se mueve de un host a otro, Aurora Serverless v2 administra las conexiones automáticamente. No es necesario cambiar el código de la aplicación cliente de base de datos o las cadenas de conexión de base de datos.

Con Aurora Serverless v2, al igual que ocurre con los clústeres aprovisionados, la capacidad de almacenamiento y la capacidad informática son independientes. Cuando hablamos de capacidad y escalado de Aurora Serverless v2, lo hacemos siempre de la capacidad de computación que aumenta o disminuye. Por lo tanto, el clúster puede contener muchos terabytes de datos incluso cuando la capacidad de la CPU y la memoria se reducen a niveles bajos.

En lugar de aprovisionar y administrar los servidores de bases de datos, hay que especificar la capacidad de la base de datos. Para más detalles acerca de la capacidad en Aurora Serverless v2, consulte Capacidad de Aurora Serverless v2. La capacidad real de cada instancia Aurora Serverless v2 de escritor o de lector varía con el tiempo, según la carga de trabajo. Para obtener más información sobre este mecanismo, consulte Escalado en Aurora Serverless v2.

> **Important**
>
> Con Aurora Serverless v1, el clúster tiene una única medida de capacidad informática que puede escalar entre los valores de capacidad mínima y máxima. Con Aurora Serverless v2, el clúster puede contener lectores además del escritor. Cada escritor y lector Aurora Serverless v2 pueden escalar entre los valores de capacidad mínima y máxima. Por lo

tanto, la capacidad total del clúster de Aurora Serverless v2 depende tanto del rango de capacidad que defina para el clúster de bases de datos como del número de escritores y lectores del clúster. En cualquier momento dado, solo se le cobrará por la capacidad de Aurora Serverless v2 que se estén utilizando activamente en el clúster de bases de datos de Aurora.

Configuraciones para clústeres de base de datos de Aurora

Para cada uno de sus clústeres de base de datos Aurora, puede elegir cualquier combinación de capacidad de Aurora Serverless v2, capacidad aprovisionada o ambas.

Puede configurar un clúster que contenga ambas, capacidad Aurora Serverless v2 y capacidad aprovisionada, llamado un clúster de configuración mixta. Por ejemplo, supongamos que necesita más capacidad de lectura/escritura de la disponible para un escritor Aurora Serverless v2. En este caso, puede configurar el clúster con un escritor aprovisionado muy grande. En ese caso, puede seguir utilizando Aurora Serverless v2 para los lectores. O, supongamos que la carga de trabajo de escritura del clúster varía pero la carga de trabajo de lectura es estable. En este caso, puede configurar el clúster con un escritor Aurora Serverless v2 y uno o más lectores aprovisionados.

También puede configurar un clúster de bases de datos en el que Aurora Serverless v2 administre toda la capacidad. Para ello, puede crear un nuevo clúster y utilizar Aurora Serverless v2 desde el principio. O bien, puede reemplazar toda la capacidad aprovisionada de un clúster existente por Aurora Serverless v2. Por ejemplo, algunas de las rutas de actualización de versiones anteriores del motor requieren comenzar con un escritor aprovisionado y reemplazarlo por un escritor Aurora Serverless v2. Para los procedimientos para crear un nuevo clúster de bases de datos con Aurora Serverless v2, o para cambiar un clúster de bases de datos existente a Aurora Serverless v2, consulte Creación de un clúster de bases de datos de Aurora Serverless v2 y Cambiar de un clúster aprovisionado a Aurora Serverless v2.

Si no utiliza Aurora Serverless v2 en absoluto en un clúster de bases de datos, todos los escritores y lectores del clúster de bases de datos serán aprovisionados. Este es el tipo de clúster de bases de datos más antiguo y común con el que la mayoría de los usuarios están familiarizados. De hecho, antes de Aurora Serverless, no había un nombre especial para este tipo de clúster de bases de datos de Aurora. La capacidad aprovisionada es constante. Los cargos son relativamente fáciles de pronosticar. No obstante, debe predecir de antemano cuánta capacidad necesita. En algunos casos, las predicciones pueden ser inexactas o las necesidades de capacidad pueden cambiar. En estos

casos, el clúster de bases de datos puede estar infraprovisionado (más lento de lo que se desea) o sobreaprovisionado (más caro de lo que se desea).

Capacidad de Aurora Serverless v2

La unidad de medida de Aurora Serverless v2 es la unidad de capacidad Aurora (ACU). La capacidad de Aurora Serverless v2 no va vinculada a las clases de instancias de base de datos que se utilizan para los clústeres aprovisionados.

Cada ACU es una combinación de aproximadamente 2 gigabytes (GiB) de memoria, la CPU correspondiente y las redes. El rango de capacidad de la base de datos se especifica mediante esta unidad de medida. Las métricas `ServerlessDatabaseCapacity` y `ACUUtilization` le ayudan a determinar cuánta capacidad está utilizando realmente su base de datos y dónde se encuentra dentro del rango especificado.

En cualquier momento, cada instancia de base de datos de escritura o lectura Aurora Serverless v2 datos tiene una capacidad. La capacidad se representa como un número de coma flotante que representa la ACU. La capacidad aumenta o disminuye cada vez que el escritor o el lector se escalan. Este valor se mide cada segundo. Para cada clúster de bases de datos en el que se pretenda utilizar Aurora Serverless v2, se defines un rango de capacidad: los valores de capacidad mínima y máxima entre los que puede escalar cada escritor o lector Aurora Serverless v2. El rango de capacidad es el mismo para cada escritor o lector Aurora Serverless v2 en un clúster de bases de datos. Cada escritor o lector Aurora Serverless v2 tiene su propia capacidad, la cual corresponderá a algún punto en ese rango.

En la tabla siguiente se muestran los rangos de capacidad de Aurora Serverless v2 compatibles con Aurora MySQL y Aurora PostgreSQL.

Rango de capacidad (ACU)	Versiones compatibles de Aurora MySQL	Versiones compatibles de Aurora PostgreSQL
0,5–128	3.02.0 y versiones posteriores	13.6 y versiones posteriores, 14.3 y versiones posteriores, 15.2 y versiones posteriores, 16.1 y versiones posteriores
0,5–256	3.06.0 y versiones posteriores	13.13 y versiones posteriores, 14.10 y versiones posteriores,

Rango de capacidad (ACU)	Versiones compatibles de Aurora MySQL	Versiones compatibles de Aurora PostgreSQL
		15.5 y versiones posteriores, 16.1 y versiones posteriores
0–256	3.08.0 y versiones posteriores	13.15 y versiones posteriores, 14.12 y versiones posteriores, 15.7 y versiones posteriores, 16.3 y versiones posteriores

La capacidad de Aurora Serverless v2 mínima que puede definir es de 0 ACU, para las versiones de Aurora Serverless v2 que admiten la característica de pausa automática. Puede especificar un número mayor si es menor o igual que el valor de capacidad máxima. Si se establece la capacidad mínima en un número pequeño, los clústeres de base de datos cargados ligeramente consumen recursos informáticos mínimos. Al mismo tiempo, permanecen listos para aceptar conexiones de inmediato y ampliarse cuando están ocupados.

Recomendamos establecer el mínimo en un valor que permita a cada instancia de base de datos de escritura o lectura mantener el conjunto de trabajo de la aplicación en el grupo del búfer. De esta forma, el contenido del grupo del búfer no se desecha durante los períodos de inactividad. Para saber todo lo que debe tener en cuenta a la hora de elegir el valor de capacidad mínima, consulte Elegir la configuración de capacidad de Aurora Serverless v2 mínima para un clúster. Para saber todo lo que debe tener en cuenta a la hora de elegir el valor de capacidad máxima, consulte Elegir la configuración de capacidad de Aurora Serverless v2 máxima para un clúster.

En función de cómo configure los lectores en una implementación multi-AZ, sus capacidades se pueden vincular a la capacidad del escritor o de forma independiente. Para obtener información detallada sobre cómo hacerlo, consulte Escalado en Aurora Serverless v2.

Supervisar Aurora Serverless v2 implica medir los valores de capacidad del escritor y los lectores del clúster de bases de datos a lo largo del tiempo. Si la base de datos no se reduce a la capacidad mínima, puede realizar acciones como ajustar el mínimo y optimizar la aplicación de base de datos. Si la base de datos alcanza su capacidad máxima de forma coherente, puede realizar acciones como aumentar el máximo. También puede optimizar la aplicación de base de datos y distribuir la carga de consultas entre más lectores.

Los cargos correspondientes a la capacidad de Aurora Serverless v2 se mide en términos de horas de ACU. Para obtener información acerca de cómo se calculan los cargos para Aurora Serverless v2, consulte la página de precios de Aurora.

Supongamos que el número total de escritores y lectores del clúster es *n*. En ese caso, el clúster consume aproximadamente *n* x *minimum ACUs* cuando no se está ejecutando ninguna operación de base de datos. Aurora puede ejecutar operaciones de supervisión o mantenimiento que provocan una pequeña cantidad de carga. Ese clúster no consume más de *n* x *maximum ACUs* cuando la base de datos se ejecuta a plena capacidad.

Para obtener más información sobre cómo elegir los valores de ACU mínimos y máximos adecuados, consulte Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora. Los valores de ACU mínimo y máximo especificados también afectan a la forma en que funcionan algunos de los parámetros de configuración de Aurora para Aurora Serverless v2. Para obtener más información sobre la interacción entre el rango de capacidad y los parámetros de configuración, consulte Trabajo con los grupos de parámetros para Aurora Serverless v2.

Escalado en Aurora Serverless v2

Para cada escritor o lector Aurora Serverless v2, Aurora realiza un seguimiento continuo del uso de recursos como la CPU, la memoria y la red. Estas mediciones se denominan colectivamente carga. La carga incluye las operaciones de base de datos realizadas por la aplicación. También incluye procesamiento en segundo plano para el servidor de base de datos y tareas administrativas de Aurora. Cuando la capacidad queda limitada por cualquiera de ellos, Aurora Serverless v2 aumenta. Aurora Serverless v2 también se amplía cuando detecta problemas de rendimiento que se pueden resolver de esta manera. Puede supervisar la utilización de los recursos y cómo afecta al escalado en Aurora Serverless v2 mediante los procedimientos en Métricas importantes de Amazon CloudWatch para Aurora Serverless v2 y Supervisión del rendimiento de Aurora Serverless v2 con la información de rendimiento.

La carga puede variar según el escritor o los lectores del clúster de bases de datos. El escritor se encarga de todas las instrucciones de lenguaje de definición de datos (DDL), tales como `CREATE TABLE`, `ALTER TABLE` y `DROP TABLE`. El escritor también se encarga de todas las instrucciones de lenguaje de manipulación de datos (DML), como `INSERT` y `UPDATE`. Los lectores pueden procesar declaraciones de solo lectura, como consultas `SELECT`.

El escalado es la operación que aumenta o disminuye la capacidad de Aurora Serverless v2 para la base de datos. Con Aurora Serverless v2, escritor y lector tiene su propio valor de capacidad en uso, medida en ACU. Aurora Serverless v2 escala un escritor o lector a una mayor capacidad cuando

su capacidad en uso es demasiado baja para manejar la carga. Escala el escritor o el lector a una capacidad inferior cuando su capacidad en uso es superior a la necesaria.

A diferencia de Aurora Serverless v1, que escala duplicando las ACU cada vez que el clúster de bases de datos alcanza un límite, Aurora Serverless v2 puede aumentar la capacidad incrementalmente. Cuando la demanda de carga de trabajo comienza a alcanzar la capacidad de base de datos en uso de un escritor o un lector, Aurora Serverless v2 aumenta el número de ACU de ese escritor y lector. Aurora Serverless v2 escala la capacidad en función de los incrementos necesarios para proporcionar el mejor rendimiento de los recursos consumidos. El escalado se produce en incrementos tan pequeños como 0,5 ACU. Cuanto mayor sea la capacidad en uso, mayor será el incremento de escalado y, por lo tanto, más rápido puede producirse el escalado.

Ya que el escalado en Aurora Serverless v2 es tan frecuente, granular y no disruptivo, no genera eventos discretos en la Consola de administración de AWS igual que lo hace Aurora Serverless v1. En su lugar, puede medir las métricas de Amazon CloudWatch, como `ServerlessDatabaseCapacity` y `ACUUtilization` y hacer un seguimiento de sus valores mínimo, máximo y medio a lo largo del tiempo. Para obtener más información sobre las métricas de Aurora, consulte Supervisión de métricas en un clúster de Amazon Aurora. Para ver sobre cómo supervisar Aurora Serverless v2, consulte Métricas importantes de Amazon CloudWatch para Aurora Serverless v2.

Puede elegir escalar el lector al mismo tiempo que el escritor asociado o independientemente del escritor. Para ello, especifique el nivel de promoción de ese lector.

- Los lectores en niveles de promoción 0 y 1 se escalan al mismo tiempo que el escritor. Este comportamiento de escalado hace que los lectores de los niveles prioritarios 0 y 1 sean ideales para disponibilidad. Esto se debe a que siempre tienen el tamaño adecuado para asumir la carga de trabajo de escritura en caso de conmutación por error.
- Los lectores de los niveles de promoción 2 a 15 escalan independientemente del escritor. Cada lector se mantiene dentro de los valores de ACU mínimo y máximo especificados para el clúster. Cuando un lector escala independientemente de la base de datos de escritura asociada, puede pasar a quedar inactiva y reducirse mientras el escritor continúa procesando un gran volumen de transacciones. Sigue disponible como objetivo de conmutación por error si no hay otros lectores disponibles en niveles de promoción más bajos. No obstante, si se promueve para ser el escritor, es posible que tenga que escalar para manejar toda la carga de trabajo del escritor.

Para obtener información detallada sobre los niveles de promoción, consulte Elegir el nivel de promoción para un lector Aurora Serverless v2.

Las nociones de los puntos de escala y los periodos de tiempo de espera asociados desde Aurora Serverless v1 no son aplicables en Aurora Serverless v2. El escalado en Aurora Serverless v2 puede ocurrir mientras hay conexiones de base de datos abiertas, mientras hay transacciones SQL en proceso, mientras hay tablas bloqueadas y mientras se utilizan tablas temporales. Aurora Serverless v2 no espera a que un llegue punto de dewscanso para comenzar a escalar. El escalado no interrumpe ninguna operación de base de datos en curso.

Si la carga de trabajo requiere más capacidad de lectura de la disponible con un solo escritor y un solo lector, puede agregar varios lectores Aurora Serverless v2 al clúster. Cada lector Aurora Serverless v2 puede escalar dentro de los valores de capacidad mínimo y máximo especificados para el clúster de bases de datos. Puede utilizar el punto de conexión del lector del clúster para dirigir las sesiones de solo lectura a los lectores y reducir la carga en el escritor.

Que Aurora Serverless v2 realice un escalado y lo rápido que se produzca el escalado una vez que se inicie, también dependerá de la configuración de ACU mínima y máxima para el clúster. Además, depende de si un lector está configurado para escalar junto con el escritor o independientemente de él. Para obtener información acerca de los factores que afectan al escalado de Aurora Serverless v2, consulte Rendimiento y escalado para Aurora Serverless v2.

Escalado hasta cero

En las versiones de Aurora MySQL y Aurora PostgreSQL recientes, los escritores y los lectores de Aurora Serverless v2 pueden escalarse hasta cero ACU. Nos referimos a esta capacidad como pausa y reanudación automáticas o pausa automática. Puede elegir si desea permitir este comportamiento al especificar un valor cero o distinto de cero para la capacidad mínima. También puede elegir cuánto tiempo esperar antes de que una instancia de Aurora Serverless v2 se detenga. Para obtener información sobre las versiones que tienen esta capacidad, consulte Capacidad de Aurora Serverless v2. Para obtener información acerca de cómo utilizarlo de manera eficaz, consulte Escalado a cero ACU con pausa y reanudación automáticas para Aurora Serverless v2.

En las versiones de Aurora MySQL y Aurora PostgreSQL anteriores, los escritores y los lectores de Aurora Serverless v2 inactivos pueden reducir verticalmente hasta el valor de ACU mínimo especificado para el clúster, pero no hasta cero ACU. En ese caso, la opción de cero ACU no está disponible al establecer el rango de capacidad. Ese comportamiento es diferente de Aurora Serverless v1, que puede detenerse después de un período de inactividad, pero luego tarda algún tiempo en reanudarse al abrir una nueva conexión.

Cuando el clúster de bases de datos con capacidad de Aurora Serverless v2 no se necesita durante algún tiempo, puede también detener e iniciar todo el clúster, lo mismo con los clústeres de bases

de datos aprovisionados. Esta técnica es la más adecuada para los sistemas de desarrollo y prueba, en los que es posible que no se necesiten durante muchas horas seguidas y la velocidad de reanudación del clúster no es crucial. La característica de detener o iniciar el clúster está disponible para todas las versiones de Aurora Serverless v2. Para obtener más información acerca de esa característica, consulte Detención e inicio de un clúster de bases de datos de Amazon Aurora.

Aurora Serverless v2 y alta disponibilidad

La forma de establecer una alta disponibilidad para un clúster de bases de datos Aurora es convertirlo en un clúster de bases de datos Multi-AZ. Un clúster de bases de datos Aurora Multi-AZ tiene capacidad informática disponible en todo momento en más de una zona de disponibilidad (AZ). Esta configuración mantiene la base de datos en funcionamiento incluso en caso de una interrupción significativa. Aurora realiza una conmutación por error automática en caso de que se produzca un problema que afecte al escritor o incluso a toda la zona de disponibilidad. Con Aurora Serverless v2 puede elegir que la capacidad de computación en espera se amplíe y disminuya junto con la capacidad de escritura. De esta forma, la capacidad de computación en la segunda zona de disponibilidad queda lista para asumir la carga de trabajo actual en cualquier momento. Al mismo tiempo, la capacidad de computación de todas las zonas de disponibilidad puede reducirse cuando la base de datos está inactiva. Para obtener más información sobre cómo funciona Aurora con Regiones de AWS y las zonas de disponibilidad, consulte Alta disponibilidad para instancias de bases de datos de Aurora.

La capacidad Multi-AZ de Aurora Serverless v2 utiliza lectores además del escritor. La compatibilidad para lectores es nueva para Aurora Serverless v2 en comparación con Aurora Serverless v1. Puede añadir hasta 15 lectores de Aurora Serverless v2 distribuidos en tres zonas de disponibilidad en un clúster de bases de datos de Aurora.

Para aplicaciones críticas para el negocio que deben permanecer disponibles incluso en caso de que se produzca un problema que afecte a todo el clúster o a toda la región de AWS, puede configurar una base de datos global de Aurora. Puede usar la capacidad de Aurora Serverless v2 en los clústeres secundarios para que estén listos para asumir el control durante la recuperación ante desastres. También se pueden reducir cuando la base de datos no está ocupada. Para obtener información detallada sobre bases de datos globales de Aurora, consulte Uso de una base de datos global de Amazon Aurora.

Aurora Serverless v2 funciona como aprovisionado para conmutación por error y otras funciones de alta disponibilidad. Para obtener más información, consulte Alta disponibilidad para Amazon Aurora.

Supongamos que desea garantizar la máxima disponibilidad de su clúster de Aurora Serverless v2. Puede crear un lector además del escritor. Si asigna al lector al nivel de promoción 0 o 1, cualquier escalado que ocurra para el escritor también se producirá en el lector. De esta forma, un lector con capacidad idéntica siempre estará listo para sustituir al escritor en caso de conmutación por error.

Supongamos que desea ejecutar informes trimestrales para su empresa al mismo tiempo que el clúster continúa procesando transacciones. Si añade un lector Aurora Serverless v2 al clúster y lo asigna a un nivel de promoción del 2 al 15, puede conectarse directamente a ese lector para ejecutar los informes. Dependiendo del uso intensivo de memoria y de la CPU que hagan las consultas de informes, ese lector puede escalar para adaptarse a la carga de trabajo. A continuación, puede reducirse de nuevo cuando los informes hayan finalizado.

Aurora Serverless v2 y almacenamiento

El almacenamiento de cada clúster de bases de datos Aurora consta de seis copias de todos sus datos, repartidas en tres zonas de disponibilidad. Esta replicación de datos integrada se aplica independientemente de si el clúster de bases de datos incluye lectores además del escritor. De esta forma, sus datos están seguros, incluso ante problemas que afecten a la capacidad de computación del clúster.

El almacenamiento de Aurora Serverless v2 tiene las mismas características de fiabilidad y durabilidad que se describen en Almacenamiento de Amazon Aurora. Esto se debe a que el almacenamiento de los clústeres de bases de datos Aurora funciona igual tanto si la capacidad de computación utiliza Aurora Serverless v2 como si es aprovisionada.

Parámetros de configuración de los clústeres de Aurora

Puede ajustar los mismos parámetros de configuración de clúster y base de datos para clústeres con capacidad de Aurora Serverless v2 como para clústeres de base de datos aprovisionados. Sin embargo, algunos parámetros relacionados con la capacidad se manejan de forma diferente para Aurora Serverless v2. En un clúster de configuración mixta, los valores de los parámetros especificados para esos parámetros relacionados con la capacidad se siguen aplicando a los escritores y los lectores aprovisionados.

Casi todos los parámetros funcionan de la misma manera para los escritores y los lectores Aurora Serverless v2 que para los aprovisionados. Las excepciones son algunos parámetros que Aurora ajusta automáticamente durante el escalado y algunos parámetros que Aurora mantiene en valores fijos que dependen de la configuración de capacidad máxima.

Por ejemplo, la cantidad de memoria reservada para la memoria caché del búfer aumenta a medida que un escritor o un lector aumenta y disminuye cuando se reduce la escala. De esta forma, la memoria se puede liberar cuando la base de datos no está ocupada. Por el contrario, Aurora establece automáticamente el número máximo de conexiones en un valor adecuado según la configuración de capacidad máxima. De esta forma, las conexiones activas no se pierden si la carga cae y Aurora Serverless v2escala a menos. Para obtener información acerca de cómo Aurora Serverless v2 gestiona parámetros específicos, consulte Trabajo con los grupos de parámetros para Aurora Serverless v2.

Requisitos y limitaciones para Aurora Serverless v2

Cuando cree un clúster en el que desee utilizar instancias de bases de datos de Aurora Serverless v2, preste atención a los siguientes requisitos y limitaciones:

Temas

- Disponibilidad en regiones y versiones
- Los clústeres que utilizan Aurora Serverless v2 deben tener un rango de capacidad especificado
- Algunas características aprovisionadas no se admiten en Aurora Serverless v2
- Algunos aspectos de Aurora Serverless v2 son diferentes en Aurora Serverless v1

Disponibilidad en regiones y versiones

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos de Aurora y entre Regiones de AWS. Para obtener más información acerca de la versión y la disponibilidad de las regiones con Aurora y Aurora Serverless v2, consulte Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.

En el siguiente ejemplo se muestran los comandos de AWS CLI para confirmar los valores exactos del motor de base de datos que puede utilizar con Aurora Serverless v2 para una Región de AWS determinada. El parámetro `--db-instance-class` para Aurora Serverless v2 es siempre `db.serverless`. El parámetro `--engine` puede ser `aurora-mysql` o `aurora-postgresql`. Sustituya los valores `--region` y `--engine` correspondientes para confirmar los valores de `--engine-version` que puede usar. Si el comando no produce ningún resultado, significa que Aurora Serverless v2 no está disponible para esa combinación de Región de AWS y motor de base de datos.

```
aws rds describe-orderable-db-instance-options --engine aurora-mysql --db-instance-
class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text

aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text
```

Los clústeres que utilizan Aurora Serverless v2 deben tener un rango de capacidad especificado

Un clúster de Aurora debe tener un atributo `ServerlessV2ScalingConfiguration` para poder añadir cualquier instancia de base de datos que utilice la clase de instancia de base de datos `db.serverless`. Este atributo especifica el rango de capacidad. La capacidad de Aurora Serverless v2 oscila entre un mínimo de 0 unidades de capacidad de Aurora (ACU) hasta un máximo de 256 ACU, en incrementos de 0,5 ACU. El valor mínimo permitido depende de la versión de Aurora. Cada ACU proporciona el equivalente a aproximadamente 2 gibibytes (GiB) de RAM y la CPU y las redes asociadas. Para obtener más información sobre cómo utiliza Aurora Serverless v2 la configuración del rango de capacidad, consulte Cómo funciona Aurora Serverless v2.

Para obtener información sobre los rangos de capacidad permitidos de las distintas versiones de motores de base de datos, consulte Capacidad de Aurora Serverless v2.

Puede especificar los valores de ACU mínimos y máximos en Consola de administración de AWS cuando cree un clúster y la instancia de base de datos de Aurora Serverless v2 asociada. También puede especificar la opción `--serverless-v2-scaling-configuration` en la AWS CLI. O, si lo desea, puede especificar el parámetro `ServerlessV2ScalingConfiguration` con la API de Amazon RDS. Puede especificar este atributo cuando cree un clúster o modifique un clúster existente. Para conocer los procedimientos para establecer el rango de capacidad, consulte Configuración del rango de capacidad de Aurora Serverless v2 para un clúster. Para obtener un análisis detallado sobre cómo elegir valores de capacidad mínima y máxima y cómo estos ajustes afectan a algunos parámetros de base de datos, consulte Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora.

Algunas características aprovisionadas no se admiten en Aurora Serverless v2

Las siguientes características de las instancias de base de datos aprovisionadas de Aurora no están disponibles actualmente para Amazon Aurora Serverless v2:

- Secuencias de actividades de la base de datos (DAS)
- Administración de cachés de clústeres para Aurora PostgreSQL. El parámetro de configuración `apg_ccm_enabled` no se aplica a instancias de base de datos de Aurora Serverless v2.

Algunas características de Aurora funcionan con Aurora Serverless v2, pero esto podría causar problemas si el rango de capacidad es inferior al necesario para los requisitos de memoria de esas características con su carga de trabajo específica. En ese caso, es posible que la base de datos no funcione tan bien como de costumbre o que se produzcan errores de falta de memoria. Para obtener recomendaciones sobre cómo configurar el rango de capacidad adecuado, consulte Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora. Para obtener información sobre la solución de problemas si la base de datos tiene errores de falta de memoria debido a un rango de capacidad mal configurado, consulte Evitar errores de memoria insuficiente.

No se admite Aurora Auto Scaling. Este tipo de escalado añade lectores nuevos para gestionar la carga de trabajo adicional que requiere un uso intensivo de lectura, con base en el uso de la CPU. Sin embargo, el escalado basado en la utilización de la CPU no tiene sentido para Aurora Serverless v2. Como alternativa, puede crear instancias de base de datos del lector de Aurora Serverless v2 de antemano y dejarlas reducidas verticalmente a baja capacidad. Es una forma más rápida y menos disruptiva de escalar la capacidad de lectura de un clúster que añadir nuevas instancias de base de datos dinámicamente.

Algunos aspectos de Aurora Serverless v2 son diferentes en Aurora Serverless v1

Si es usuario de Aurora Serverless v1 y es la primera vez que usa Aurora Serverless v2, consulte las diferencias entre los requisitos de Aurora Serverless v2 y Aurora Serverless v1 para saber qué requisitos son diferentes en Aurora Serverless v1 y Aurora Serverless v2.

Creación de un clúster de base de datos que utiliza Aurora Serverless v2

Para crear un clúster de Aurora en el que pueda agregar instancias de bases de datos de Aurora Serverless v2, siga el mismo procedimiento que en Creación de un clúster de base de datos de Amazon Aurora. Con Aurora Serverless v2, los clústeres son intercambiables por clústeres aprovisionados. Puede tener clústeres donde algunas instancias de base de datos utilicen Aurora Serverless v2 y se aprovisionen algunas instancias de base de datos.

Temas

- Configuración de clústeres de bases de datos de Aurora Serverless v2
- Creación de un clúster de bases de datos de Aurora Serverless v2
- Creación de una instancia de base de datos de escritura de Aurora Serverless v2

Configuración de clústeres de bases de datos de Aurora Serverless v2

Asegúrese de que la configuración inicial del clúster cumpla los requisitos que se indican en Requisitos y limitaciones para Aurora Serverless v2. Especifique la siguiente configuración para asegurarse de que puede agregar instancias de base de datos de Aurora Serverless v2 al clúster:

Región de AWS

Cree el clúster en una Región de AWS donde haya instancias de base de datos de Aurora Serverless v2 disponibles. Para obtener más información sobre las regiones disponibles, consulte Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.

Versión del motor de base de datos

Elija una versión de motor compatible con Aurora Serverless v2. Para obtener información sobre los requisitos de versión de Aurora Serverless v2, consulte Requisitos y limitaciones para Aurora Serverless v2.

Clase de instancia de base de datos

Si crea un clúster mediante la Consola de administración de AWS, elija la clase de instancia de base de datos de la instancia de base de datos de escritura al mismo tiempo. Elija la clase de instancia de base de datos Serverless (Sin servidor). Al elegir esa clase de instancia de base de datos, también especifica el rango de capacidad de la instancia de base de datos de escritura. El

mismo rango de capacidad se aplica al resto de instancias de base de datos Aurora Serverless v2 que añada a ese clúster.

Si no ve la opción Sin servidor para la clase de instancia de base de datos, elija una versión de motor de base de datos compatible con Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.

Cuando utiliza la AWS CLI o la API de Amazon RDS, el parámetro que especifica para la clase de instancia de base de datos es `db.serverless`.

Capacity range (Rango de capacidad)

Rellene los valores mínimos y máximos de la unidad de capacidad Aurora (ACU) aplicables a todas las instancias de base de datos del clúster. Esta opción está disponible en ambas páginas de la consola Create cluster (Crear el clúster) y Add reader (Añadir lector) al elegir Serverless (Sin servidor) para la clase de instancia de base de datos.

Para obtener información sobre los rangos de capacidad permitidos de las distintas versiones de motores de base de datos, consulte Capacidad de Aurora Serverless v2.

Si no ve los campos de ACU mínima y máxima, asegúrese de elegir la clase de instancia de base de datos Sin servidor de la instancia de base de datos de escritura.

En un principio se crea el clúster con una instancia de base de datos aprovisionada, no se especifican las ACU mínimas y máximas. En ese caso puede modificar el clúster posteriormente para agregar esa opción. También puede añadir una instancia de base de datos de lectura Aurora Serverless v2 al clúster. El rango de capacidad se especifica como parte de ese proceso.

Hasta que no especifique el rango de capacidad del clúster, no podrá agregar ninguna instancia de base de datos Aurora Serverless v2 al clúster mediante la AWS CLI o la API de RDS. Si intenta añadir una instancia de base de datos de Aurora Serverless v2, aparecerá un error. En la AWS CLI o en los procedimientos la API de RDS, el rango de capacidad se representa mediante el atributo `ServerlessV2ScalingConfiguration`.

Para los clústeres que contienen más de una instancia de base de datos de lectura, la prioridad de conmutación por error de cada instancia de base de datos de lectura Aurora Serverless v2 desempeña un papel importante en la forma en que esa instancia de base de datos escala (a más o a menos). No puede especificar la prioridad en el momento de crear el clúster. Tenga en cuenta esta propiedad cuando agregue una segunda instancia de base de datos de lectura (o más) al clúster.

Para obtener más información, consulte Elegir el nivel de promoción para un lector Aurora Serverless v2.

Creación de un clúster de bases de datos de Aurora Serverless v2

Puede usar la Consola de administración de AWS, la AWS CLI o la API de RDS para crear un clúster de base de datos de Aurora Serverless v2.

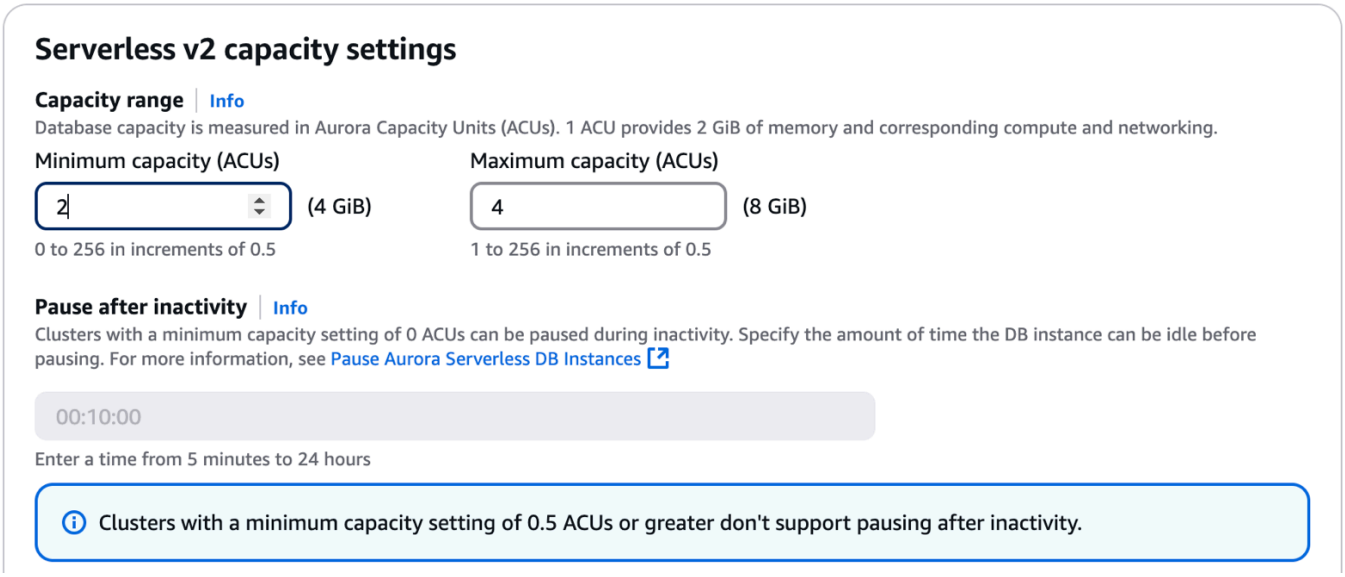
Consola

Para crear un clúster con escritor de Aurora Serverless v2

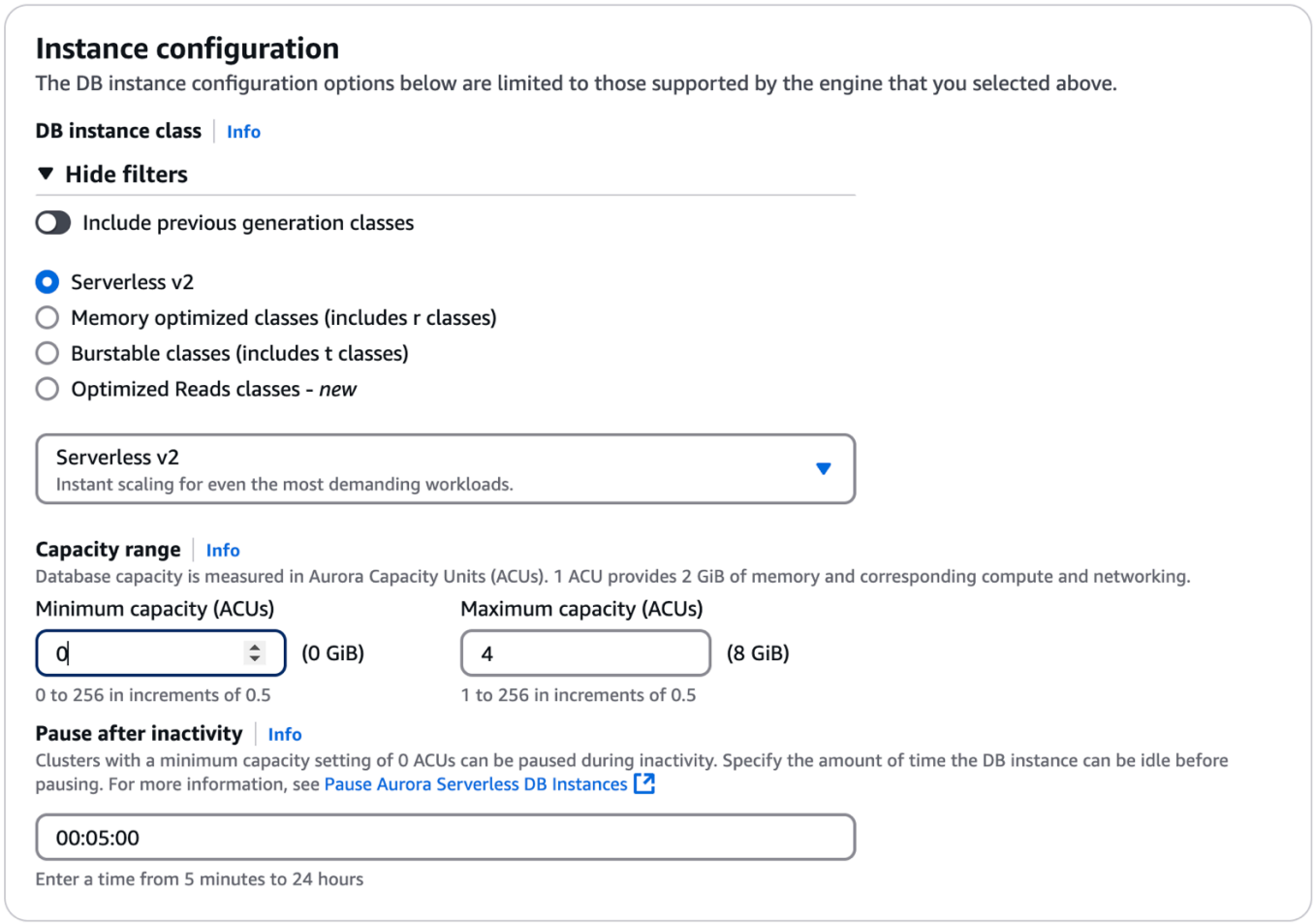
1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija Create database (Creación de base de datos). En la página que aparece, elija las opciones siguientes:
   - En Tipo de motor, elija Aurora (compatible con MySQL) o Aurora (compatible con PostgreSQL).
   - En Versión, elija una de las versiones compatibles para Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.
4. En Clase de instancia de base de datos, elija Sin servidor v2.
5. En Configuración de capacidad, puede aceptar el rango predeterminado. O puede elegir otros valores para las unidades de capacidad mínima y máxima. Puede elegir entre un ACU mínimo de 0 y un ACU máximo de 256, en incrementos de ACU de 0,5.

   Para obtener más información acerca de las unidades de capacidad de Aurora Serverless v2, consulte Capacidad de Aurora Serverless v2 y Rendimiento y escalado para Aurora Serverless v2.

   Según el motor y la versión que elija, el límite superior puede ser de 128 ACU, el límite inferior de 0,5 ACU o ambos. Para obtener más información sobre el límite de cada combinación de motor y versión de Aurora, consulte Capacidad de Aurora Serverless v2.



Si se selecciona una capacidad mínima de 0 ACU, se habilita la función de pausa y reanudación automáticas de Aurora Serverless v2. En ese caso, puede elegir además cuánto tiempo deben esperar las instancias de base de datos de Aurora Serverless v2 sin conexión a la base de datos antes de realizar una pausa automática. Para obtener información sobre la capacidad de pausa y reanudación automáticas, consulte Escalado a cero ACU con pausa y reanudación automáticas para Aurora Serverless v2.



6. Elija cualquier otra configuración de clúster de base de datos, tal y como se describe en Configuración de clústeres de bases de datos de Aurora.
7. Elija Crear base de datos para crear un clúster de base de datos de Aurora con una instancia de base de datos Aurora Serverless v2 como instancia de escritura, también llamada instancia de base de datos principal.

CLI

Para crear un clúster de bases de datos compatible con instancias de base de datos Aurora Serverless v2 con AWS CLI, siga el procedimiento de CLI Creación de un clúster de base de datos de Amazon Aurora. Incluya los siguientes parámetros en su comando `create-db-cluster`:

- --region *región_de_AWS_donde_las_instancias_de_Aurora Serverless v2_están disponibles*
- --engine-version *versión_de_motor_compatible_con_serverless_v2*

- --serverless-v2-scaling-configuration MinCapacity=*capacidad_mínima*,MaxCapacity=*capacidad_máxima*

En el siguiente ejemplo se muestra la creación de un clúster de base de datos de Aurora Serverless v2.

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.04.1 \
    --serverless-v2-scaling-configuration MinCapacity=1,MaxCapacity=4 \
    --master-username myuser \
    --manage-master-user-password
```

> **Note**
>
> Al crear un clúster de base de datos de Aurora Serverless v2 utilizando la AWS CLI, el modo de motor aparece en la salida como `provisioned` en lugar de `serverless`. El modo de motor `serverless` hace referencia a Aurora Serverless v1.

En este ejemplo se especifica la opción `--manage-master-user-password` para generar la contraseña administrativa y administrarla en Secrets Manager. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager. También puede utilizar la opción `--master-password` para especificar y administrar la contraseña usted mismo.

Para obtener información sobre los requisitos de versión de Aurora Serverless v2, consulte Requisitos y limitaciones para Aurora Serverless v2. Para obtener información sobre los números permitidos para el rango de capacidad y qué representan esos números, consulte Capacidad de Aurora Serverless v2 y Rendimiento y escalado para Aurora Serverless v2.

Para comprobar si un clúster existente tiene la configuración de capacidad especificada, compruebe la salida del comando `describe-db-clusters` para el atributo `ServerlessV2ScalingConfiguration`. Este atributo tiene un aspecto similar al siguiente.

```
"ServerlessV2ScalingConfiguration": {
```

```
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Si no especifica las ACU mínimas y máximas al crear el clúster, puede utilizar el comando `modify-db-cluster` después para agregar esa configuración. Hasta que lo haga no podrá añadir ninguna instancias de base de datos Aurora Serverless v2 al clúster. Si intenta añadir una instancia de base de datos de `db.serverless`, aparecerá un error.

API

Para crear un clúster de bases de datos compatible con instancias de base de datos Aurora Serverless v2 que utilicen la API de RDS, siga el procedimiento de API en Creación de un clúster de base de datos de Amazon Aurora. Seleccione los siguientes valores. Asegúrese de que su operación `CreateDBCluster` incluye los siguientes parámetros:

```
EngineVersion serverless_v2_compatible_engine_version
ServerlessV2ScalingConfiguration with MinCapacity=minimum_capacity and
 MaxCapacity=maximum_capacity
```

Para obtener información sobre los requisitos de versión de Aurora Serverless v2, consulte Requisitos y limitaciones para Aurora Serverless v2. Para obtener información sobre los números permitidos para el rango de capacidad y qué representan esos números, consulte Capacidad de Aurora Serverless v2 y Rendimiento y escalado para Aurora Serverless v2.

Para comprobar si un clúster existente tiene la configuración de capacidad especificada, compruebe la salida de la operación `DescribeDBClusters` para el atributo `ServerlessV2ScalingConfiguration`. Este atributo tiene un aspecto similar al siguiente.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Si no especifica las ACU mínimas y máximas al crear el clúster, puede utilizar la operación `ModifyDBCluster` después para agregar esa configuración. Hasta que lo haga no podrá añadir ninguna instancias de base de datos Aurora Serverless v2 al clúster. Si intenta añadir una instancia de base de datos de `db.serverless`, aparecerá un error.

Creación de una instancia de base de datos de escritura de Aurora Serverless v2

Aunque especifique el rango de capacidad de Aurora Serverless v2 al crear un clúster de Aurora, puede elegir si desea usar Aurora Serverless v2 o aprovisionado para cada instancia de base de datos del clúster. Para empezar a usar Aurora Serverless v2 inmediatamente en el clúster de base de datos, agregue una instancia de base de datos de escritura que utilice la clase de instancia de `db.serverless`. En la consola, normalmente se hace esta elección como parte del flujo de trabajo para crear el clúster de base de datos. Por lo tanto, este procedimiento se aplica principalmente cuando se realiza la configuración a través de la CLI.

Consola

Cuando se crea un clúster de base de datos con la Consola de administración de AWS, en ese momento se especifican las propiedades de la instancia de base de datos de escritura. Para que la instancia de base de datos de escritura utilice Aurora Serverless v2, elija la clase d instancia de base de datos Serverless (Sin servidor).

A continuación deberá establecer el rango de capacidad del clúster especificando los valores mínimos y máximos de la unidad de capacidad Aurora (ACU). Los mismos valores mínimo y máximo se aplican a cada instancia de base de datos Aurora Serverless v2 del clúster. Para ver información sobre ese procedimiento y ver información sobre la importancia de la configuración de capacidad mínima y máxima, consulte Creación de un clúster de bases de datos de Aurora Serverless v2.

Si no crea una instancia de base de datos Aurora Serverless v2 en el momento de crear el clúster por primera vez, puede agregar una o varias instancias de base de datos Aurora Serverless v2 más adelante. Para ello, siga los procedimientos en Adición de un lector Aurora Serverless v2 y en Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2. Especifique el rango de capacidad en el momento en el que añada la primera instancia de base de datos Aurora Serverless

v2 al clúster. Puede cambiar el rango de capacidad posteriormente siguiendo el procedimiento en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.

CLI

Cuando crea un clúster de base de datos de Aurora Serverless v2 mediante la AWS CLI, agrega explícitamente la instancia de base de datos del escritor mediante el comando create-db-instance. Incluya el siguiente parámetro:

- `--db-instance-class db.serverless`

En el siguiente ejemplo se muestra la creación de una instancia de base de datos del escritor de Aurora Serverless v2.

```
aws rds create-db-instance \
    --db-cluster-identifier my-serverless-v2-cluster \
    --db-instance-identifier my-serverless-v2-instance \
    --db-instance-class db.serverless \
    --engine aurora-mysql
```

Administración de clústeres de bases de datos de Aurora Serverless v2

Con Aurora Serverless v2, los clústeres son intercambiables por clústeres aprovisionados. Las propiedades Aurora Serverless v2 se aplican a una o varias instancias de base de datos dentro de un clúster. Así pues, los procedimientos para crear clústeres, modificar clústeres, crear y restaurar instantáneas, etc., son básicamente los mismos que para otros tipos de clústeres de Aurora. Para obtener información sobre procedimientos generales para administrar clústeres e instancias de base de datos de Aurora, consulte Administración de un clúster de base de datos de Amazon Aurora.

En los siguientes temas, puede obtener más información sobre elementos importantes a la hora de administrar clústeres que contengan instancias de bases de datos de Aurora Serverless v2.

Temas

- Configuración del rango de capacidad de Aurora Serverless v2 para un clúster
- Comprobación del rango de capacidad para Aurora Serverless v2
- Adición de un lector Aurora Serverless v2

- Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2
- Conversión de un escritor o un lector Aurora Serverless v2 a aprovisionado
- Elegir el nivel de promoción para un lector Aurora Serverless v2
- Uso de TLS/SSL con Aurora Serverless v2
- Visualización de instancias de Aurora Serverless v2 de escritura y lectura
- Registros en Aurora Serverless v2

Configuración del rango de capacidad de Aurora Serverless v2 para un clúster

Para modificar los parámetros de configuración u otros parámetros de clústeres que contengan instancias de base de datos Aurora Serverless v2, o las propias instancias de base de datos, siga los mismos procedimientos generales que para los clústeres aprovisionados. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora.

La opción más importante exclusiva de Aurora Serverless v2 es el rango de capacidad. Después de establecer los valores mínimos y máximos de la unidad de capacidad Aurora (ACU) para un clúster Aurora, no es necesario ajustar activamente la capacidad de las instancias de base de datos Aurora Serverless v2 del clúster. Aurora lo hace por usted. Esta opción de configuración se administra en el nivel de clúster. Los mismos valores de ACU mínima y máxima se aplican a cada instancia de base de datos Aurora Serverless v2 del clúster.

Puede establecer los valores específicos siguientes:

- Minimum ACUs (UCA mínimas): la instancia de base de datos Aurora Serverless v2 puede reducir la capacidad hasta este número de ACU.
- Maximum ACUs (UCA máximas): la instancia de base de datos Aurora Serverless v2 puede aumentar la capacidad hasta este número de ACU.

> **Note**
>
> Al modificar el rango de capacidad de un clúster de base de datos Aurora Serverless v2, el cambio se produce inmediatamente, independientemente de si decide aplicarlo de inmediato o durante el siguiente período de mantenimiento programado.

Para ver más información sobre los efectos del rango de capacidad y cómo supervisarlo y ajustarlo, consulte Métricas importantes de Amazon CloudWatch para Aurora Serverless v2 y Rendimiento y escalado para Aurora Serverless v2. Su objetivo es asegurarse de que la capacidad máxima del clúster sea lo suficientemente alta como para gestionar los picos de la carga de trabajo y que la mínima sea lo suficientemente baja como para minimizar los costos cuando el clúster no esté ocupado.

Supongamos que determina tras la supervisión que el rango de ACU para el clúster debe ser mayor, menor, o más o menos amplio. Puede establecer la capacidad de un clúster de Aurora en un rango específico mediante la Consola de administración de AWS, la AWS CLI o la API de Amazon RDS. Este rango de capacidad se aplica a cada instancia de base de datos Aurora Serverless v2 del clúster.

Por ejemplo, supongamos que el clúster tiene un rango de capacidad de 1 a 16 ACU y contiene dos instancias de base de datos Aurora Serverless v2. Luego el clúster en su conjunto consume entre 2 ACU (cuando está inactivo) y 32 ACU (cuando está en uso total). Si cambia el rango de capacidad de 8 a 20,5 ACU, ahora el clúster consume 16 ACU cuando está inactivo y hasta 41 ACU cuando están en completo uso.

Aurora establece automáticamente ciertos parámetros para instancias de base de datos Aurora Serverless v2 en valores que dependen del valor máximo de ACU en el rango de capacidad. Para ver la lista completa de estos parámetros, consulte Número máximo de conexiones para Aurora Serverless v2. Para los parámetros estáticos que se basan en este tipo de cálculo, el valor se evalúa de nuevo al reiniciar la instancia de base de datos. Por lo tanto, puede actualizar el valor de dichos parámetros reiniciando la instancia de base de datos después de cambiar el rango de capacidad. Para comprobar si necesita reiniciar la instancia de base de datos para detectar estos cambios en los parámetros, consulte el atributo `ParameterApplyStatus` de la instancia de base de datos. Un valor de `pending-reboot` indica que el reinicio aplicará cambios a los valores de algunos parámetros.

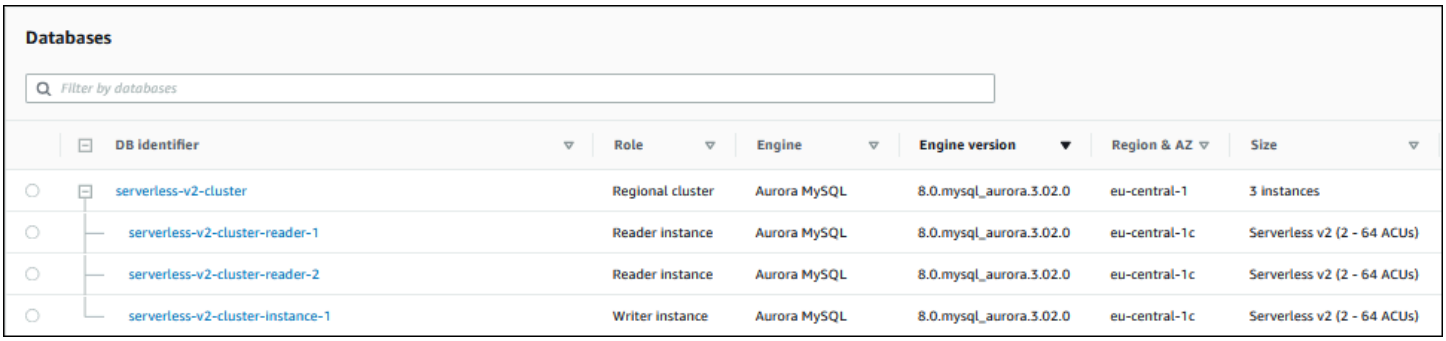
Consola

Puede establecer el rango de capacidad de un clúster que contenga bases de datos Aurora Serverless v2 mediante la Consola de administración de AWS.

Cuando utiliza la consola, establece el rango de capacidad del clúster en el momento en que agrega la primera instancia de base de datos Aurora Serverless v2 en ese clúster. Podría hacerlo al elegir la clase de instancia de base de datos Serverless v2 (Sin servidor v2) para la instancia de base de datos de escritura al crear el clúster. O podría hacerlo al elegir la clase de instancia de base

de datos Serverless v2 (Sin servidor v2) cuando se agrega una instancia de base de datos de escritura Aurora Serverless v2 al clúster. También podría hacerlo al convertir una instancia de base de datos aprovisionada existente en el clúster en la clase de instancia de base de datos Serverless (Sin servidor). Para ver las versiones completas de estos procedimientos, consulte Creación de una instancia de base de datos de escritura de Aurora Serverless v2, Adición de un lector Aurora Serverless v2 y Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2.

Cualquiera que sea el rango de capacidad que establezca en el nivel de clúster, este se aplicará a todas las instancias de base de datos Aurora Serverless v2 del clúster. La siguiente imagen muestra un clúster con varias instancias de base de datos de lectura Aurora Serverless v2. Cada uno tiene un rango de capacidad idéntico de 2 a 64 ACU.

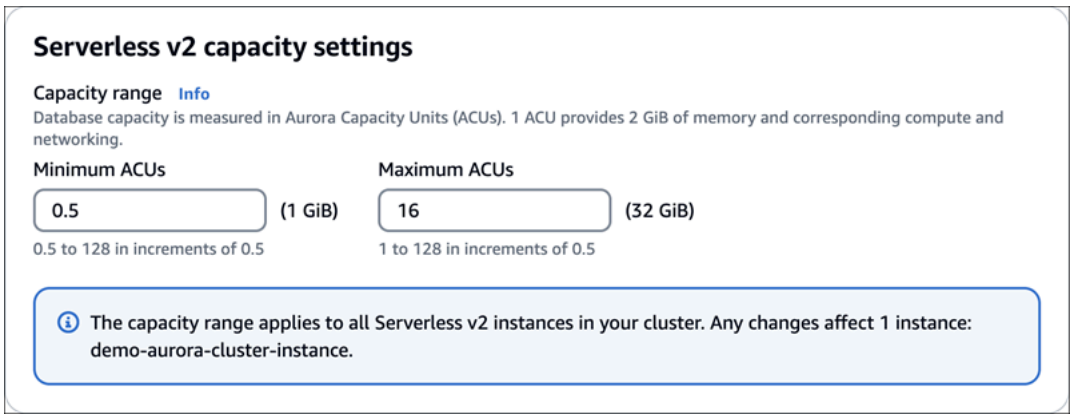


Para modificar la capacidad de un clúster de Aurora Serverless v2

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster que contiene sus instancias de base de datos Aurora Serverless v2 en la lista. El clúster debe contener ya al menos una instancia de base de datos Aurora Serverless v2. De lo contrario, Aurora no muestra la sección Capacity range (Rango de capacidad).
4. Para Actions (Acciones), elija Modify (Modificar).
5. En la sección Capacity range (Rango de capacidad), elija lo siguiente:

    a. Introduzca un valor para Minimum ACUs (UCA mínimas). La consola muestra el rango de valores permitidos. Puede elegir una capacidad mínima entre 0,5 y 128 ACU. Puede elegir una capacidad máxima entre 1 y 128 ACU. Puede ajustar los valores de capacidad en incrementos de 0,5 ACU.

    b. Introduzca un valor para Maximum ACUs (UCA máximas). Este valor debe ser igual o superior a Minimum ACUs (ACU mínimas). La consola muestra el rango de valores permitidos. En la siguiente figura se muestra esa opción.



6. Elija Continue (Continuar). Aparecerá la página Summary of modifications (Resumen de modificaciones).
7. Seleccione Apply immediately (Aplicar inmediatamente).

   La capacidad de cambio se produce inmediatamente, independientemente de si decide aplicarlo de inmediato o durante el siguiente período de mantenimiento programado.
8. Elija Modify clúster (Modificar clúster) para aceptar el resumen de las modificaciones. También puede elegir Back (Atrás) para modificar los cambios o Cancel (Cancelar) para descartar los cambios.

AWS CLI

Para configurar la capacidad de un clúster en el que se va a utilizar instancias de base de datos Aurora Serverless v2 mediante la AWS CLI, ejecute el comando modify-db-clúster de la AWS CLI. Especifique la opción `--serverless-v2-scaling-configuration`. Es posible que el clúster ya contenga una o varias instancias de base de datos Aurora Serverless v2, o puede agregar las de base de datos más adelante. Las claves y los valores válidos para los campos `MinCapacity` y `MaxCapacity` incluyen los siguientes:

- `0.5`, `1`, `1.5`, `2`, etc., en incrementos de 0,5 y hasta un máximo de 128.

En este ejemplo se establece el rango de ACU de un clúster de bases de datos Aurora llamado `sample-cluster` hasta un mínimo de `48.5` y un máximo de 64.

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
--serverless-v2-scaling-configuration MinCapacity=48.5,MaxCapacity=64
```

La capacidad de cambio se produce inmediatamente, independientemente de si decide aplicarlo de inmediato o durante el siguiente período de mantenimiento programado.

Después de hacerlo, podrá añadir las instancias de base de datos Aurora Serverless v2 al clúster, y cada nueva instancia de base de datos podrá escalarse entre 48,5 y 64 ACU. El nuevo rango de capacidad también se aplica a cualquier instancia de base de datos Aurora Serverless v2 que ya estuviera en el clúster. Las instancias de base de datos se escalan a más o menos si es necesario para situarse dentro del nuevo rango de capacidad.

Para ver más ejemplos de configuración del rango de capacidad mediante la CLI, consulte Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora.

Para modificar la configuración de escalado de un clúster de bases de datos de Aurora Serverless mediante la AWS CLI, ejecute el comando modify-db-clúster de la AWS CLI. Especifique la opción `--serverless-v2-scaling-configuration` para configurar la capacidad mínima y la capacidad máxima. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: `0.5`, `1`, `1.5`, `2`, etc., en incrementos de 0,5 ACU hasta un máximo de `128`.
- Aurora PostgreSQL: `0.5`, `1`, `1.5`, `2`, etc., en incrementos de 0,5 ACU hasta un máximo de `128`.

En el siguiente ejemplo se modifica la configuración de escalado de una base de datos Aurora Serverless v2 llamada `sample-instance` que forma parte de un clúster llamado `sample-cluster`.

Para Linux, macOS o:Unix

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

En:Windows

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster ^
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

API de RDS

Puede establecer la capacidad de una instancia de base de datos Aurora mediante la operación de la API ModifyDBclúster. Especifique el parámetro `ServerlessV2ScalingConfiguration`. Las claves y los valores válidos para los campos `MinCapacity` y `MaxCapacity` incluyen los siguientes:

- `0.5`, `1`, `1.5`, `2`, etc., en incrementos de 0,5 y hasta un máximo de 128.

Puede modificar la configuración de escalado de un clúster que contenga instancias de base de datos Aurora Serverless v2 mediante la operación de la API ModifyDBclúster. Especifique el parámetro `ServerlessV2ScalingConfiguration` para configurar la capacidad mínima y la capacidad máxima. Entre los valores de capacidad válidos se incluyen los siguientes:

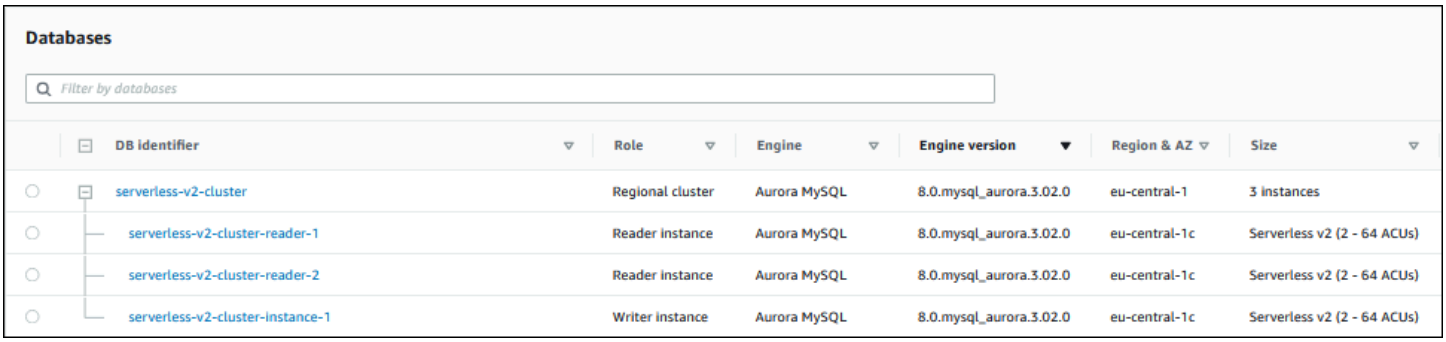
- Aurora MySQL: `0.5`, `1`, `1.5`, `2`, etc., en incrementos de 0,5 ACU hasta un máximo de `128`.
- Aurora PostgreSQL: `0.5`, `1`, `1.5`, `2`, etc., en incrementos de 0,5 ACU hasta un máximo de `128`.

La capacidad de cambio se produce inmediatamente, independientemente de si decide aplicarlo de inmediato o durante el siguiente período de mantenimiento programado.

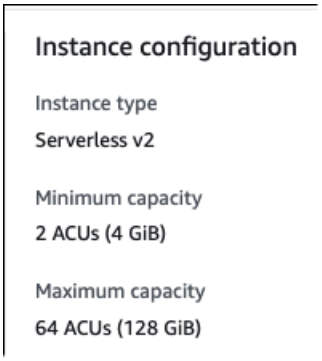
Comprobación del rango de capacidad para Aurora Serverless v2

El procedimiento para comprobar el rango de capacidad del clúster de Aurora Serverless v2requiere que primero establezca un rango de capacidad. Si aún no lo ha hecho, siga el procedimiento en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.

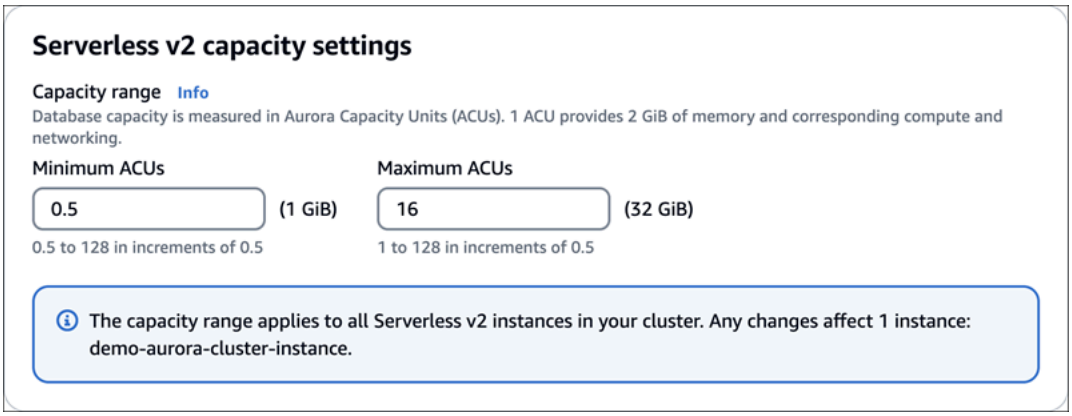
Cualquiera que sea el rango de capacidad que establezca en el nivel de clúster, este se aplicará a todas las instancias de base de datos Aurora Serverless v2 del clúster. La siguiente imagen muestra un clúster con varias instancias de base de datos Aurora Serverless v2. Cada uno tiene un rango de capacidad idéntico.



También puede ver la página de detalles de cualquier instancia de base de datos Aurora Serverless v2 en el clúster. El rango de capacidad de las instancias de base de datos aparece en la pestaña Configuration (Configuración).



También puede ver el rango de capacidad actual del clúster en la página Modify (Modificar) del clúster. En la siguiente imagen se ve cómo. En ese momento podrá cambiar el rango de capacidad. Para conocer todas las formas en que puede configurar o cambiar el rango de capacidad, consulte Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.



Comprobación del rango de capacidad en uso de un clúster Aurora

Puede comprobar el rango de capacidad configurado para instancias de base de datos Aurora Serverless v2 en un clúster examinando el atributo `ServerlessV2ScalingConfiguration` del clúster. Los siguientes ejemplos de AWS CLI muestran un clúster con una capacidad mínima de 0,5 unidades de capacidad Aurora (ACU) y una capacidad máxima de 16 ACU.

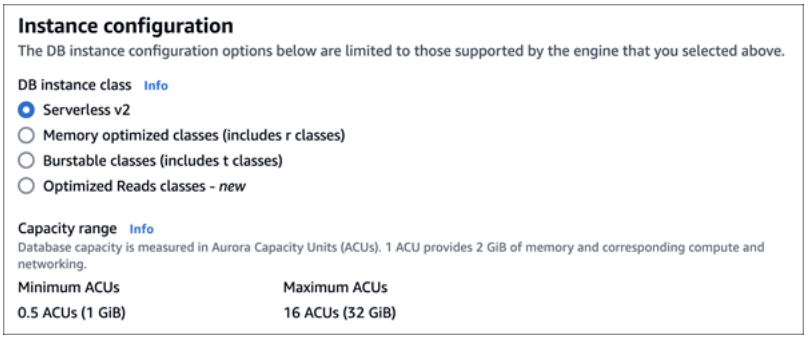
```
$ aws rds describe-db-clusters --db-cluster-identifier serverless-v2-64-acu-cluster \
  --query 'DBClusters[*].[ServerlessV2ScalingConfiguration]'
[
    [
        {
            "MinCapacity": 0.5,
            "MaxCapacity": 16.0
        }
    ]
```

```
]
```

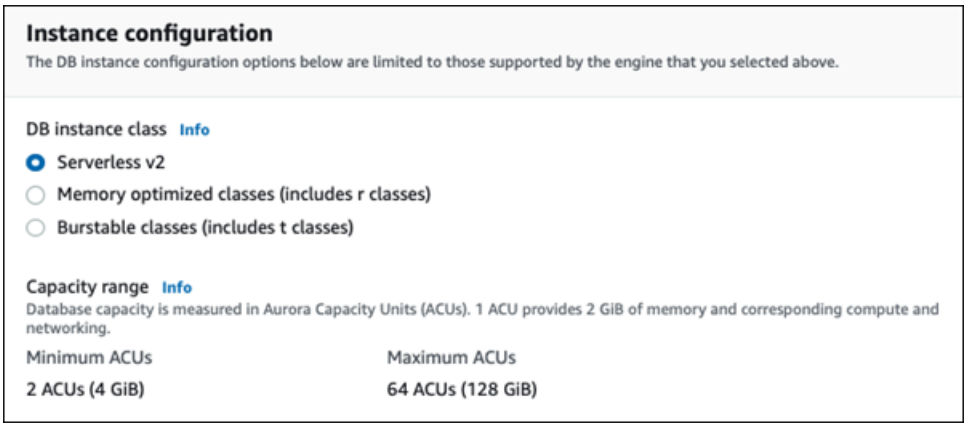
Adición de un lector Aurora Serverless v2

Para agregar una instancia de base de datos Aurora Serverless v2 de lectura al clúster, siga el mismo procedimiento general que en Adición de réplicas de Aurora a un clúster de base de datos. Elija la clase de instancia Serverless v2 (V2 sin servidor) para la nueva instancia de base de datos.

Si la instancia de base de datos de lectura es la primera instancia de base de datos Aurora Serverless v2 en el clúster, también se elige el rango de capacidad. En la imagen siguiente se muestran los controles que utiliza para especificar las unidades de capacidad (ACU) mínima y máxima de Aurora. Esta configuración se aplica a esta instancia de base de datos de lectura y a cualquier otra instancias de base de datos Aurora Serverless v2 que se añada al clúster. Cada instancia de base de datos Aurora Serverless v2 puede escalarse entre los valores de ACU mínima y máxima.



Si ya ha añadido alguna instancia de base de datos Aurora Serverless v2 al clúster, al agregar otra instancia de base de datos Aurora Serverless v2 de lectura se muestra el rango de capacidad en uso. En la imagen siguiente se muestran los controles de solo lectura.



Si desea cambiar el rango de capacidad del clúster, siga el procedimiento en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.

Para los clústeres que contienen más de una instancia de base de datos de lectura, la prioridad de conmutación por error de cada instancia de base de datos de lectura Aurora Serverless v2 desempeña un papel importante en la forma en que esa instancia de base de datos escala (a más o a menos). No puede especificar la prioridad en el momento de crear el clúster. Tenga en cuenta esta propiedad cuando agregue una segunda instancia de base de datos de lectura (o más) al clúster. Para obtener más información, consulte Elegir el nivel de promoción para un lector Aurora Serverless v2.

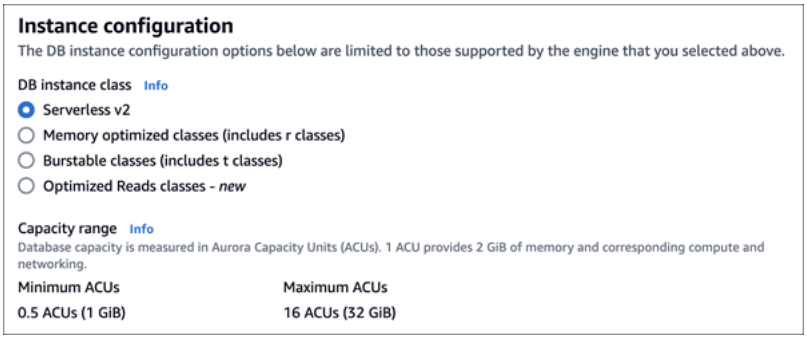
Para saber otras formas de ver el rango de capacidad en uso de un clúster, consulte Comprobación del rango de capacidad para Aurora Serverless v2.

Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2

Puede convertir una instancia de base de datos aprovisionada para usar Aurora Serverless v2. Para ello, siga el procedimiento en Modificación de una instancia de base de datos en un clúster de base de datos. El clúster debe cumplir los requisitos de Requisitos y limitaciones para Aurora Serverless v2. Por ejemplo, las instancias de base de datos Aurora Serverless v2 requieren que el clúster ejecute ciertas versiones mínimas del motor.

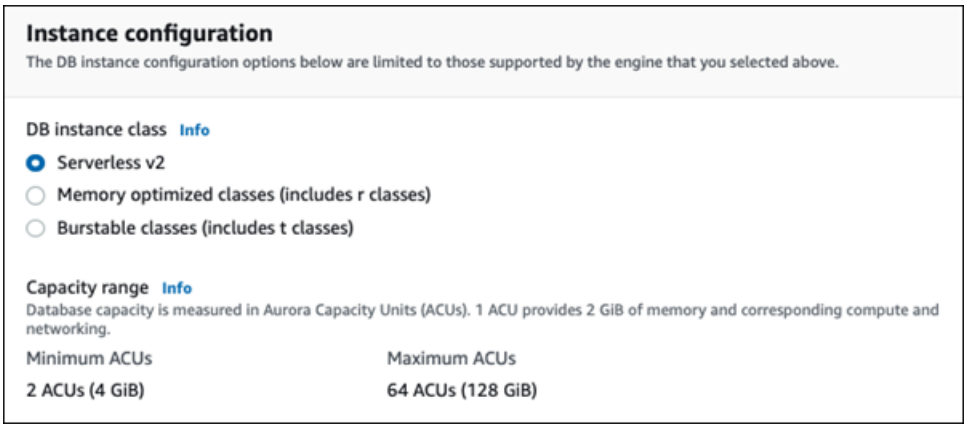
Supongamos que va a convertir un clúster aprovisionado en ejecución para aprovechar Aurora Serverless v2. En ese caso puede minimizar el tiempo de inactividad convirtiendo una instancia de base de datos a Aurora Serverless v2 como primer paso del proceso de conmutación. Para ver el procedimiento completo, consulte Cambiar de un clúster aprovisionado a Aurora Serverless v2.

Si la instancia de base de datos que convierte es la primera instancia de base de datos Aurora Serverless v2 en el clúster, elija el rango de capacidad del clúster como parte de la operación Modify (Modificar). Este rango de capacidad se aplica a cada instancia de base de datos Aurora Serverless v2 que se agregue al clúster. En la imagen siguiente se ve la página donde se especifican las unidades de capacidad (ACU) mínima y máxima de Aurora.



Para ver más información acerca de la importancia del rango de capacidad, consulte Capacidad de Aurora Serverless v2.

Si el clúster ya contiene una o más instancias de base de datos Aurora Serverless v2, verá el rango de capacidad existente durante la operación Modify (Modificar). En la siguiente imagen se muestra un ejemplo de ese panel de información.



Si desea cambiar el rango de capacidad del clúster después de agregar más instancias de base de datos Aurora Serverless v2, siga el procedimiento en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.

Conversión de un escritor o un lector Aurora Serverless v2 a aprovisionado

Puede convertir una instancia de base de datos Aurora Serverless v2 a una instancia de base de datos aprovisionada. Para ello, siga el procedimiento en Modificación de una instancia de base de datos en un clúster de base de datos. Elija la clase de instancia de base de datos distinta a Serverless (Sin servidor).

Podría convertir una instancia de base de datos Aurora Serverless v2 a aprovisionada si necesita una capacidad superior a la disponible con las unidades de capacidad máxima de Aurora (ACU) de una instancia de base de datos Aurora Serverless v2. Por ejemplo, las clases de instancia de base de datos db.r5 y db.r6g más grandes tienen una capacidad de memoria mayor que la que puede llegar a alcanzar una instancia de base de datos Aurora Serverless v2 al escalarse.

> **Tip**
>
> Algunas de las clases de instancias de base de datos anteriores, como db.r3 y db.t2, no están disponibles para las versiones de Aurora que se utilizan con Aurora Serverless v2. Para ver qué clases de instancias de base de datos puede utilizar al convertir una instancia

de base de datos Aurora Serverless v2 a una aprovisionada, consulte Motores de base de datos compatibles para clases de instancia de base de datos.

Si va a convertir la instancia de base de datos de escritura del clúster desdeAurora Serverless v2 para aprovisionarla, puede seguir el procedimiento en Cambiar de un clúster aprovisionado a Aurora Serverless v2, pero en sentido inverso. Cambie una de las instancias de base de datos de lectura del clúster de Aurora Serverless v2a aprovisionada. A continuación, realice una conmutación por error para convertir esa instancia de base de datos aprovisionada en la de escritura.

Los rangos de capacidad que haya especificado anteriormente para el clúster siguen igual, incluso si todas las instancias de base de datos Aurora Serverless v2 se eliminan del clúster. Si desea cambiar el rango de capacidad, puede modificar el clúster, tal y como se explica en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.

Elegir el nivel de promoción para un lector Aurora Serverless v2

Para clústeres que contienen varias instancias de base de datos Aurora Serverless v2 o una combinación de instancias de base de datos Aurora Serverless v2 y aprovisionadas, preste atención a la configuración del nivel de promoción para cada instancia de base de datos Aurora Serverless v2. Esta configuración controla más comportamiento para instancias de base de datos Aurora Serverless v2 que para instancias de base de datos aprovisionadas.

En la Consola de administración de AWS, especifique esta configuración mediante la opción Failover priority (Prioridad de conmutación por error) para las páginas Additional configuration (Configuración adicional), Create database (Crear base de datos), Modify instance (Modificar instancia) y Add reader (Añadir lector). Puede ver esta propiedad para las instancias de base de datos existentes en la columna opcional Priority tier (Nivel prioritario) de la página Databases (Bases de datos). También puede ver esta propiedad en la página de detalles de un clúster de bases de datos o en una instancia de base de datos.

Para las instancias de base de datos aprovisionadas, la elección de nivel 0 a 15 determina únicamente el orden en que Aurora elige qué instancia de base de datos de lectura debe promocionarse a de escritura durante una operación de conmutación por error. Para instancias de base de datos Aurora Serverless v2 de lectura, el número de nivel también determina si la instancia de base de datos se amplía para que coincida con la capacidad de la instancia de base de datos de escritura o se escala de manera independiente en función de su propia carga de trabajo. Las instancias de base de datos Aurora Serverless v2 de lectura en los niveles 0 o 1 se mantienen en una capacidad mínima, al menos tan alta como la instancia de base de datos de escritura. De esta

forma, quedan listas para tomar el relevo de la instancia de base de datos de escritura en caso de conmutación por error. Si la instancia de base de datos de escritura es una instancia de base de datos aprovisionada, Aurora estima la capacidad equivalente de Aurora Serverless v2. Utilice esa estimación como capacidad mínima para la instancia de base de datos Aurora Serverless v2 de lectura.

Las instancias de base de datos Aurora Serverless v2 de lectura de los niveles 2 a 15 no tienen la misma restricción en cuanto a capacidad mínima. Cuando están inactivas pueden escalarse hasta el valor mínimo de la unidad de capacidad (ACU) de Aurora especificado en el rango de capacidad del clúster.

El siguiente ejemplo de la AWS CLI de Linux muestra cómo examinar los niveles de promoción de todas las instancias de base de datos del clúster. El campo final incluye un valor de `True` para la instancia de base de datos de escritura y `False` para todas las instancias de base de datos de lectura.

```
$ aws rds describe-db-clusters --db-cluster-identifier promotion-tier-demo \
  --query 'DBClusters[*].DBClusterMembers[*].
[PromotionTier,DBInstanceIdentifier,IsClusterWriter]' \
  --output text

1   instance-192  True
1   tier-01-4840  False
10  tier-10-7425  False
15  tier-15-6694  False
```

El siguiente ejemplo de la AWS CLI de Linux muestra cómo cambiar el nivel de promoción de una instancia de base de datos concreta del clúster. Los comandos modifican primero la instancia de base de datos con un nuevo nivel de promoción. A continuación, esperan a que la instancia de base de datos vuelva a estar disponible y confirman el nuevo nivel de promoción de la instancia de base de datos.

```
$ aws rds modify-db-instance --db-instance-identifier instance-192 --promotion-tier 0
$ aws rds wait db-instance-available --db-instance-identifier instance-192
$ aws rds describe-db-instances --db-instance-identifier instance-192 \
  --query '*[].[PromotionTier]' --output text
0
```

Para obtener más información sobre cómo especificar niveles de promoción para diferentes casos de uso, consulte Escalado en Aurora Serverless v2.

Uso de TLS/SSL con Aurora Serverless v2

Aurora Serverless v2 utiliza el protocolo de Transport Layer Security/Secure Sockets Layer (TLS/SSL) para cifrar las comunicaciones entre los clientes y el clúster de bases de datos de Aurora Serverless v2. Admite TLS/SSL versiones 1.0, 1.1 y 1.2. Para obtener información general sobre cómo se usa de IAM con RDS y Aurora, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

Para obtener más información acerca de cómo conectarse a la base de datos de Aurora MySQL con el cliente MySQL, consulte Conexión a una instancia de base de datos que ejecuta el motor de base de datos MySQL.

Aurora Serverless v2 admite todos los modos de TLS/SSL disponibles para el cliente MySQL (`mysql`) y el cliente PostgreSQL (`psql`), incluidos los enumerados en la tabla siguiente.

Descripción del modo TLS/SSL	mysql	psql
Conéctese sin utilizar TLS/SSL.	DISABLED	disable
Intente conectarse usando TLS/SSL primero, pero vuelva a no SSL si es necesario.	PREFERRED	preferir (predeterminado)
Aplicar mediante TLS/SSL.	REQUIRED	require
Implemente TLS/SSL y verifique la entidad de certificación (CA).	VERIFY_CA	verify-ca
Aplique TLS/SSL, verifique la CA y compruebe el nombre de alojamiento de la CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v2 utiliza certificados comodín. Si especifica la opción “verify CA (verificar CA)” o “verify CA and CA hostname (verificar CA y nombre de alojamiento de CA)” al utilizar TLS/SSL, descargue primero el almacén de confianza Amazon Root CA 1 de Amazon Trust Services. Después

de hacerlo, puede identificar este archivo con formato PEM en el comando del cliente. Para hacerlo utilizando el cliente PostgreSQL, haga lo que sigue.

Para Linux, macOS o:Unix

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Para obtener más información acerca de cómo trabajar con la base de datos de Aurora PostgreSQL usando el cliente PostgreSQL, consulte Conexión a una instancia de base de datos que ejecuta el motor de base de datos PostgreSQL.

Para obtener más información acerca de cómo conectarse a los clústeres de base de datos de Aurora en general, consulte Conexión a un clúster de base de datos Amazon Aurora.

Conjuntos de cifrado compatibles para conexiones a clústeres de base de datos de Aurora Serverless v2

Mediante el uso de conjuntos de cifrado configurables, puede tener más control sobre la seguridad de las conexiones de su base de datos. Puede especificar una lista de conjuntos de cifrado que desea permitir para proteger las conexiones TLS/SSL del cliente a su base de datos. Con conjuntos de cifrado configurables, puede controlar el cifrado de conexión que acepta el servidor de base de datos. Esto evita el uso de cifrados que no son seguros o que ya no se utilizan.

Los clústeres de base de datos de Aurora Serverless v2 basados en Aurora MySQL admiten los mismos conjuntos de cifrado que los clústeres de base de datos aprovisionados de Aurora MySQL. Para obtener información sobre estos conjuntos de cifrado, consulte Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora MySQL.

Los clústeres de base de datos de Aurora Serverless v2 basados en Aurora PostgreSQL admiten los mismos conjuntos de cifrado que los clústeres de base de datos aprovisionados de Aurora PostgreSQL. Para obtener información sobre estos conjuntos de cifrado, consulte Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora PostgreSQL.

Visualización de instancias de Aurora Serverless v2 de escritura y lectura

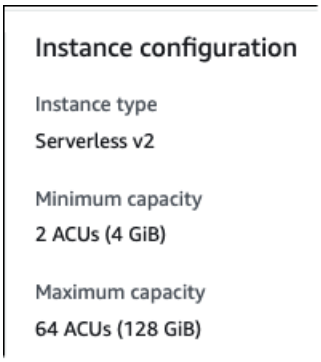
Puede ver los detalles de las instancias de base de datos Aurora Serverless v2 del mismo modo que lo hace para las instancias de base de datos aprovisionadas. Para ello, siga el procedimiento general en Visualización de un clúster de base de datos de Amazon Aurora. Un clúster podría contener

todas las instancias de base de datos Aurora Serverless v2, todas las instancias de base de datos aprovisionadas o algunas de ambas.

Después de crear una o varias instancias de base de datos Aurora Serverless v2 puede consultar cuáles de dellas son de tipo Serverless (Sin servidor) y cuáles son de tipo Instance (Instancia). También puede ver las unidades de capacidad mínima y máxima de Aurora (ACU) que puede usar la instancia de base de datos Aurora Serverless v2. Cada ACU es una combinación de capacidad de procesamiento (CPU) y de memoria (RAM). Este rango de capacidad se aplica a cada instancia de base de datos Aurora Serverless v2 del clúster. Para que el procedimiento compruebe el rango de capacidad de un clúster o de cualquier instancia de base de datos Aurora Serverless v2 en el clúster, consulte Comprobación del rango de capacidad para Aurora Serverless v2.

En la Consola de administración de AWS, las instancias de base de datos Aurora Serverless v2 aparecen marcadas bajo la columna Size (Tamaño) de la página Databases (Bases de datos). Las instancias de base de datos aprovisionadas muestran el nombre de una clase de instancia de base de datos como r6g.xlarge. Las instancias de base de datosAurora Serverless Serverless (Sin servidor) para la clase de instancia de base de datos, junto con la capacidad mínima y máxima de la instancia de base de datos. Por ejemplo, podría ver Serverless v2 (4–64 ACUs) o Serverless v2 (1–40 ACUs).

Podrá ver la misma información en la pestaña Configuration (Configuración) de cada instancia de base de datos Aurora Serverless v2 en la consola. Por ejemplo, es posible que vea una sección Instance type (Tipo de instancia) como la siguiente. Aquí, el valor Instance type (Tipo de instancia) es Serverless v2 (Sin servidor v2), el valor Minimum capacity (Capacidad mínima) es 2 ACUs (4 GiB) y el valor Maximum capacity (Capacidad máxima) es 64 ACUs (128 GiB).



Puede supervisar la capacidad de cada instancia de base de datos Aurora Serverless v2 a lo largo del tiempo. De esta forma, puede comprobar las ACU mínima, máxima y media que consume cada instancia de base de datos. También puede comprobar lo cerca que ha llegado la instancia de base de datos a su capacidad mínima o máxima. Para ver estos detalles en la Consola de administración

de AWS, examine los gráficos de las métricas de Amazon CloudWatch en la pestaña Monitoring (Supervisión) de la instancia de base de datos. Para obtener información acerca de las métricas de supervisión y cómo interpretarlas, consulte Métricas importantes de Amazon CloudWatch para Aurora Serverless v2.

Registros en Aurora Serverless v2

Para activar el registro de bases de datos, especifique qué registros que se van a habilitar mediante parámetros de configuración en el grupo de parámetros personalizado.

Para Aurora MySQL, puede habilitar los siguientes registros.

Aurora MySQL	Descripción
`general_log`	Crea el registro general. Se establece en 1 para activarlo. El valor predeterminado es desactivado (0).
`log_queries_not_using_indexes`	Registra todas las consultas en el registro de consultas lentas que no utilizan un índice. El valor predeterminado es desactivado (0). Se establece en 1 para activar este registro.
`long_query_time`	Evita que las consultas de ejecución rápida se registren en el registro de consultas lentas. Se puede establecer en un número flotante entre 0 y 31536000. El valor predeterminado es 0 (no activo).
`server_audit_events`	La lista de eventos que se deben capturar en los registros. Los valores admitidos son `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` y `QUERY_DML` y `TABLE`.
`server_audit_logging`	Se establece en 1 para activar el registro de auditoría del servidor. Si activa esta opción, puede especificar los eventos de auditoría que se enviarán a CloudWatch enumerándolos en el parámetro de `server_audit_events` .

Aurora MySQL	Descripción
`slow_query_log`	Crea un registro de consulta lento. Se establece en 1 para activar el registro de consultas lentas. El valor predeterminado es desactivado (0).

Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.

Para Aurora PostgreSQL, puede habilitar los siguientes registros en sus instancias de base de datos Aurora Serverless v2.

Aurora PostgreSQL	Descripción
`log_connections`	Registra cada conexión realizada correctam ente.
`log_disconnections`	Registra el final de una sesión, incluida su duración.
`log_lock_waits`	El valor predeterminado es 0 (desactivado). Se establece en 1 para registrar las esperas de bloqueo.
`log_min_duration_statement`	La duración mínima (en milisegundos) para que una instrucción se ejecute antes de que se registre.
`log_min_messages`	Define los niveles de los mensajes que se registran. Los valores admitidos son `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Para registrar los datos de rendimiento en el registro de `postgres`, establezca el valor en `debug1`.

Aurora PostgreSQL	Descripción
`log_temp_files`	Registra el uso de archivos temporales que superan los kilobytes especificados (kB).
`log_statement`	Controla las instrucciones SQL específicas que se registran. Los valores admitidos son `none`, `ddl`, `mod` y `all`. El valor predeterminado es `none`.

Temas

- Registro con Amazon CloudWatch
- Visualización registros de Aurora Serverless v2 en Amazon CloudWatch
- Capacidad de monitoreo con Amazon CloudWatch

Registro con Amazon CloudWatch

Después de usar el procedimiento en Registros en Aurora Serverless v2 para elegir qué registros de bases de datos activar, puede elegir qué registros cargar (“publicar”) en Amazon CloudWatch.

Amazon CloudWatch le permite analizar los datos de registro, crear alarmas y ver métricas. De forma predeterminada, los registros de errores para Aurora Serverless v2 están habilitados y se cargan automáticamente en CloudWatch. También puede cargar otros registros desde instancias de base de datos Aurora Serverless v2 a CloudWatch.

A continuación se elige cuál de esos registros desea cargar en CloudWatch con las opciones de Log exports (Exportaciones de registros) en la Consola de administración de AWS o en la opción `--enable-cloudwatch-logs-exports` de la AWS CLI.

Puede elegir cuál de sus registros de Aurora Serverless v2 cargar en CloudWatch. Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.

Igual que con cualquier tipo de clúster de base de datos de Aurora, no se puede modificar el grupo de parámetros de clúster de base de datos predeterminado. En su lugar, puede crear su propio grupo de parámetros de clúster de base de datos basado en un parámetro predeterminado para su clúster de base de datos y tipo de motor. Le recomendamos que cree su grupo de parámetros de clúster de

base de datos personalizado antes de crear el clúster de base de datos de Aurora Serverless v2, de modo que esté disponible para elegirlo cuando cree una base de datos en la consola.

> **Note**
>
> Para Aurora Serverless v2, puede crear clústeres de base de datos y grupos de parámetros de base de datos. Esto contrasta con Aurora Serverless v1, donde solo puede crear grupos de parámetros de clústeres de bases de datos.

Visualización registros de Aurora Serverless v2 en Amazon CloudWatch

Tras utilizar el procedimiento en Registro con Amazon CloudWatch para elegir qué registros de bases de datos activar, podrá ver el contenido de los registros.

Para obtener más información acerca de cómo usar CloudWatch con registros de Aurora MySQL y Aurora PostgreSQL, consulte Monitoreo de eventos de registro en Amazon CloudWatch y Publicación de registros de Aurora PostgreSQL en Amazon CloudWatch Logs.

Para ver los registros del clúster de bases de datos de Aurora Serverless v2

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija su Región de AWS.
3. Elija Log groups (Grupos de registros).
4. Seleccione el registro del clúster de bases de datos de Aurora Serverless v2 de la lista. El patrón de nomenclatura de los registros es el siguiente.

   ```
   /aws/rds/cluster/cluster-name/log_type
   ```

> **Note**
>
> Para los clústeres de bases de datos Aurora Serverless v2 compatibles con Aurora MySQL, el registro de errores incluye los eventos de escalado de grupos de búferes incluso cuando no hay errores.

Capacidad de monitoreo con Amazon CloudWatch

Aurora Serverless v2 le permite utilizar CloudWatch para supervisar la capacidad y el uso de todos las instancias de base de datos Aurora Serverless v2 del clúster. Puede ver las métricas en el nivel de instancia para comprobar la capacidad de cada instancia de base de datos Aurora Serverless v2 a medida que se amplía o reduce. También puede comparar las métricas relacionadas con la capacidad con otras métricas para ver cómo los cambios en las cargas de trabajo afectan el consumo de recursos. Por ejemplo, puede comparar `ServerlessDatabaseCapacity` con `DatabaseUsedMemory`, `DatabaseConnections` y `DMLThroughput` para evaluar cómo responde su clúster de bases de datos durante las operaciones. Para obtener más información sobre las métricas relacionadas con la capacidad que se aplican a Aurora Serverless v2, consulte Métricas importantes de Amazon CloudWatch para Aurora Serverless v2.

Para monitorear la capacidad del clúster de base de datos de Aurora Serverless v2

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Metrics (Métricas). Todas las métricas disponibles aparecen como tarjetas en la consola, agrupadas por nombre de servicio.
3. Elija RDS.
4. (Opcional) Utilice el cuadro de búsqueda encontrar las métricas que son especialmente importantes para:Aurora Serverless v2 `ServerlessDatabaseCapacity`, `ACUUtilization`, `CPUUtilization` y `FreeableMemory`.

Se recomienda configurar un panel de CloudWatch para supervisar la capacidad del clúster de bases de datos de Aurora Serverless v2 con métricas relacionadas con la capacidad. Para obtener información acerca de cómo hacerlo, consulte Creación de paneles con CloudWatch.

Para obtener más información acerca del uso de Amazon CloudWatch con Amazon Aurora, consulte Publicación de registros de Amazon Aurora MySQL en Amazon CloudWatch Logs.

Rendimiento y escalado para Aurora Serverless v2

Los siguientes procedimientos y ejemplos muestran cómo se puede establecer el rango de capacidad para clústeres de Aurora Serverless v2 y sus instancias de base de datos asociadas. También puede utilizar los siguientes procedimientos para supervisar la ocupación de las instancias de base de datos. A continuación, puede utilizar lo averiguado para determinar si necesita ajustar el rango de capacidad a más o a menos.

Antes de utilizar estos procedimientos, asegúrese de estar familiarizado con cómo funciona el escalado en Aurora Serverless v2. El mecanismo de escalado es diferente al de Aurora Serverless v1. Para obtener más información, consulte Escalado en Aurora Serverless v2.

Contenido

- Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora
  - Elegir la configuración de capacidad de Aurora Serverless v2 mínima para un clúster
  - Elegir la configuración de capacidad de Aurora Serverless v2 máxima para un clúster
  - Ejemplo: Cambiar el rango de capacidad de Aurora Serverless v2 de un clúster de Aurora MySQL
  - Ejemplo: Cambiar el rango de capacidad Aurora Serverless v2 de un clúster de Aurora PostgreSQL
- Trabajo con los grupos de parámetros para Aurora Serverless v2
  - Valores de parámetros predeterminados
  - Número máximo de conexiones para Aurora Serverless v2
  - Parámetros que Aurora ajusta cuando se escala Aurora Serverless v2 a más o a menos
  - Parámetros en los que Aurora hace sus cómputos basándose en la capacidad máxima de Aurora Serverless v2
- Evitar errores de memoria insuficiente
- Métricas importantes de Amazon CloudWatch para Aurora Serverless v2
  - Cómo se aplican las métricas de Aurora Serverless v2 en la factura de AWS
  - Ejemplos de comandos de CloudWatch para métricas de Aurora Serverless v2
- Supervisión del rendimiento de Aurora Serverless v2 con la información de rendimiento
- Solución de problemas de capacidad de Aurora Serverless v2

Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora

Con las instancias de base de datos Aurora Serverless v2, usted establece el rango de capacidad que se aplica a todas las instancias de base de datos del clúster de bases de datos al mismo tiempo que agrega la primera instancia de base de datos Aurora Serverless v2 en el clúster de bases de datos. Para ver el procedimiento para hacerlo, consulte Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.

También puede cambiar el rango de capacidad de un clúster existente. En las siguientes secciones se explica con más detalle cómo elegir los valores mínimos y máximos adecuados y qué ocurre cuando se realiza un cambio en el rango de capacidad. Por ejemplo, cambiar el rango de capacidad puede modificar los valores predeterminados de algunos parámetros de configuración. Aplicar todos los cambios de parámetros puede requerir reiniciar cada instancia de base de datos Aurora Serverless v2.

Temas

- Elegir la configuración de capacidad de Aurora Serverless v2 mínima para un clúster
- Elegir la configuración de capacidad de Aurora Serverless v2 máxima para un clúster
- Ejemplo: Cambiar el rango de capacidad de Aurora Serverless v2 de un clúster de Aurora MySQL
- Ejemplo: Cambiar el rango de capacidad Aurora Serverless v2 de un clúster de Aurora PostgreSQL

Elegir la configuración de capacidad de Aurora Serverless v2 mínima para un clúster

Es tentador elegir siempre 0,5 para la configuración de capacidad mínima de Aurora Serverless v2. Este valor permite que la instancia de base de datos reduzca verticalmente al mínimo la capacidad cuando esté completamente inactiva y, al mismo tiempo, permanezca activa. También puede activar el comportamiento de pausa automática especificando una capacidad mínima de 0 ACU, como se explica en Escalado a cero ACU con pausa y reanudación automáticas para Aurora Serverless v2. No obstante, en función de cómo se utilice ese clúster y de los demás ajustes que se configuren, un capacidad mínima diferente podría ser lo más efectivo. Debe tener en cuenta los factores siguientes al elegir la configuración de capacidad mínima:

- La tasa de escalado de una instancia de base de datos Aurora Serverless v2 depende de su capacidad actual. Cuanto mayor sea la capacidad actual, más rápido podrá ampliarse. Si necesita que la instancia de base de datos se amplíe rápidamente a una capacidad muy alta, piense en establecer la capacidad mínima en un valor en el que la tasa de escalado cumpla con sus requisitos.
- Si normalmente modifica la clase de instancia de base de datos de sus instancias de base de datos en previsión de una carga de trabajo especialmente alta o baja, puede aprovechar esa experiencia para hacer una estimación aproximada del rango de capacidad de Aurora Serverless v2 equivalente. Para determinar el tamaño de la memoria que se utilizará en momentos de poco tráfico, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

Por ejemplo, supongamos que utiliza la clase de instancia de base de datos db.r6g.xlarge cuando el clúster tiene una carga de trabajo baja. Esta clase de instancia de base de datos tiene 32 GiB de memoria. Por lo tanto, puede especificar una configuración mínima de unidad de capacidad Aurora (ACU) de 16 para configurar una instancia de base de datos Aurora Serverless v2 que puede escalarse hacia menos a aproximadamente a la misma capacidad. Esto se debe a que cada ACU corresponde a aproximadamente 2 GiB de memoria. Puede especificar un valor algo inferior para permitir que la instancia de base de datos se amplíe aún más en caso de que la instancia de base de datos db.r6g.xlarge se haya infrautilizado a veces.

- Si la aplicación funciona de la manera más eficiente cuando las instancias de base de datos tienen cierta cantidad de datos en la memoria caché del búfer, piense en especificar una configuración mínima de ACU en la que la memoria sea lo suficientemente grande como para contener los datos a los que se accede con frecuencia. De lo contrario, algunos datos se sacan de la memoria caché del búfer cuando las instancias de base de datos Aurora Serverless v2 se escalan a un tamaño de memoria inferior. Luego, cuando las instancias de base de datos se escalan a más, la información se vuelve a leer en la caché del búfer a lo largo del tiempo. Si la cantidad de E/S para devolver los datos a la caché del búfer es importante, podría resultar más eficaz elegir un valor de ACU mínimo más alto.
- Si las instancias de base de datos Aurora Serverless v2 se ejecutan la mayor parte del tiempo en una capacidad determinada, piense en especificar una configuración de capacidad mínima inferior a esa línea de base, pero no muy inferior. Aurora Serverless v2 Las instancias de base de datos pueden estimar de la manera más eficaz cuánto y con qué rapidez deben ampliarse cuando la capacidad actual no es drásticamente inferior a la capacidad requerida.
- Si la carga de trabajo aprovisionada tiene requisitos de memoria demasiado altos para clases de instancias de base de datos pequeñas como T3 o T4g, elija una configuración de ACU mínima que proporcione memoria comparable a una instancia de base de datos R5 o R6g.

  En particular, recomendamos la siguiente capacidad mínima de uso con las funciones especificadas (estas recomendaciones están sujetas a cambios):

  - Performance Insights (Información sobre rendimiento): 2 ACU
  - Bases de datos globales de Aurora: 8 ACU (se aplica solo a la Región de AWS principal)
- En Aurora, la replicación se produce en la capa de almacenamiento, por lo que la capacidad del lector no afecta directamente a la replicación. Sin embargo, en el caso de las instancias de bases de datos de lector de Aurora Serverless v2 que se escalan de forma independiente, asegúrese de que la capacidad mínima sea suficiente para gestionar las cargas de trabajo durante los periodos de escritura intensiva a fin de evitar la latencia de las consultas. Si las instancias de base de datos

de lector en los niveles de promoción 2 a 15 experimentan problemas de rendimiento, considere aumentar la capacidad mínima del clúster. Para obtener más información sobre cómo elegir si las instancias de base de datos de lectura se escalan junto con el escritor o de manera independiente, consulte Elegir el nivel de promoción para un lector Aurora Serverless v2.

- Si tiene un clúster de base de datos con instancias de base de datos lectoras de Aurora Serverless v2, los lectores no se escalan junto con la instancia de base de datos escritora cuando el nivel de promoción de los lectores no es 0 o 1. En ese caso, establecer una capacidad mínima baja puede provocar un retraso excesivo de la replicación. Esto se debe a que es posible que los lectores no tengan la capacidad suficiente para aplicar cambios de escritura cuando la base de datos esté ocupada. Se recomienda establecer la capacidad mínima en un valor que represente una cantidad de memoria y CPU comparables con la de la instancia de base de datos escritora.
- El valor del parámetro `max_connections` para las instancias de base de datos de Aurora Serverless v2 se basa en el tamaño de memoria obtenido del número máximo de ACU. Sin embargo, cuando especifique una capacidad mínima de 0 o 0,5 ACU en instancias de base de datos compatibles con PostgreSQL, el valor máximo de `max_connections` está limitado a 2000.

  Si desea utilizar el clúster de Aurora PostgreSQL para una carga de trabajo de alta conexión, piense en utilizar una configuración de ACU mínima de 1 o más. Para obtener más información acerca de cómo Aurora Serverless v2 maneja el parámetro de configuración `max_connections`, consulte Número máximo de conexiones para Aurora Serverless v2.
- El tiempo que se tarda en una instancia de base de datos Aurora Serverless v2 en escalar desde su capacidad mínima hasta su capacidad máxima depende de la diferencia entre sus valores de ACU mínima y máxima. Cuando la capacidad actual de la instancia de base de datos es grande, Aurora Serverless v2 se amplía en incrementos mayores que cuando la instancia de base de datos comienza desde una pequeña capacidad. Por lo tanto, si especifica una capacidad máxima relativamente grande y la instancia de base de datos está la mayor parte del tiempo cerca de esa capacidad, considere aumentar la configuración mínima de ACU. De esta forma, una instancia de base de datos inactiva podrá escalar de nuevo hasta la máxima capacidad más rápidamente.

Elegir la configuración de capacidad de Aurora Serverless v2 máxima para un clúster

Es tentador elegir siempre un valor alto para la configuración de capacidad de Aurora Serverless v2 máxima. Una alta capacidad máxima permite que la instancia de base de datos se amplíe al máximo cuando ejecuta una carga de trabajo intensiva. Un valor bajo evita la posibilidad de cobros inesperados. Según cómo utilice ese clúster y los demás ajustes que configure, el valor más efectivo

podría ser mayor o menor de lo que pensaba originalmente. Tenga en cuenta los factores siguientes al elegir la configuración de capacidad máxima:

- La capacidad máxima debe ser al menos tan alta como la capacidad mínima. Usted puede definir la capacidad mínima y máxima para que sean idénticas. Sin embargo, en ese caso, la capacidad nunca aumenta ni baja. Por lo tanto, utilizar valores idénticos para la capacidad mínima y máxima no es apropiado fuera de situaciones de prueba.
- La capacidad máxima debe ser superior a 0,5 ACU. Puede establecer la capacidad mínima y máxima para que sean iguales en la mayoría de los casos. No obstante, no puede especificar 0,5 tanto para el mínimo como para el máximo. Utilice un valor de 1 o superior para obtener la capacidad máxima.
- Si normalmente modifica la clase de instancia de base de datos de sus instancias de base de datos en previsión de una carga de trabajo especialmente alta o baja, puede aprovechar esa experiencia para hacer una estimación del rango de capacidad de Aurora Serverless v2 equivalente. Para determinar el tamaño de la memoria que se utilizará en momentos de mucho tráfico, consulte Especificaciones de hardware para clases de instancia de base de datos para Aurora.

  Por ejemplo, supongamos que utiliza la clase de instancia de base de datos db.r6g.4xlarge cuando el clúster tiene una carga de trabajo elevada. Esta clase de instancia de base de datos tiene 128 GiB de memoria. Por lo tanto, puede especificar una configuración de ACU máxima de 64 para configurar una instancia de base de datos Aurora Serverless v2 que pueda escalar hasta aproximadamente la misma capacidad. Esto se debe a que cada ACU corresponde a aproximadamente 2 GiB de memoria. Puede especificar un valor algo mayor para permitir que la instancia de base de datos se amplíe más en caso de que la instancia de base de datos db.r6g.4xlarge a veces no tenga la capacidad suficiente para gestionar la carga de trabajo de forma eficaz.
- Si tiene un límite presupuestario para usar su base de datos, elija un valor que se mantenga dentro de ese límite, incluso si todas las instancias de base de datos de Aurora Serverless v2 se ejecutan a máxima capacidad todo el tiempo. Recuerde eso cuando tenga n Aurora Serverless v2instancias de base de datos en el clúster, la máxima capacidad de Aurora Serverless v2 teórica que el clúster puede consumir en cualquier momento es n veces la configuración máxima de ACU para el clúster. (La cantidad real consumida podría ser menor, por ejemplo, si algunos lectores escalan independientemente del escritor).
- Si hace uso de instancias de base de datos Aurora Serverless v2 de lector para descargar parte de la carga de trabajo de solo lectura de la instancia de base de datos de escritor, es posible que

pueda elegir una configuración de capacidad máxima inferior. Esto se hace para reflejar que cada instancia de base de datos de lector no necesita escalar tan alto como si el clúster contuviera solo una instancia de base de datos.

- Supongamos que desea protegerse contra el uso excesivo por parámetros de base de datos mal configurados o consultas ineficientes en la aplicación. En ese caso, puede evitar un uso excesivo por error eligiendo un ajuste de capacidad máxima inferior al más alto absoluto que podría establecer.
- Si los picos por actividad real del usuario son poco frecuentes, pero sí ocurren, puede tener en cuenta esas ocasiones al elegir la configuración de capacidad máxima. Si la prioridad es que la aplicación siga funcionando con un rendimiento y escalabilidad completos, puede especificar una configuración de capacidad máxima superior a la observada en el uso normal. Si está bien que la aplicación se ejecute con un rendimiento reducido durante picos de actividad muy extremos, puede elegir una configuración de capacidad máxima ligeramente inferior. Asegúrese de elegir una configuración que aún tenga suficiente memoria y recursos de CPU para mantener la aplicación en ejecución.
- Si habilita opciones de configuración en el clúster que aumenten el uso de memoria para cada instancia de base de datos, tenga en cuenta esa memoria al decidir el valor máximo de ACU. Estas opciones incluyen las de información del rendimiento, consultas paralelas de Aurora MySQL, esquema de rendimiento de Aurora MySQL y replicación de registros binarios de Aurora MySQL. Asegúrese de que el valor máximo de la ACU permite que las instancias de base de datos Aurora Serverless v2 se escalen lo suficiente como para gestionar la carga de trabajo cuando se usen esas funciones. Para obtener información sobre cómo solucionar problemas causados por la combinación de una configuración de ACU máxima baja y características de Aurora que impongan sobrecarga de memoria, consulte Evitar errores de memoria insuficiente.

Ejemplo: Cambiar el rango de capacidad de Aurora Serverless v2 de un clúster de Aurora MySQL

El siguiente ejemplo de AWS CLI muestra cómo actualizar el rango de ACU para instancias de base de datos Aurora Serverless v2 en un clúster de Aurora MySQL existente. En un principio, el rango de capacidad para el clúster es 8-32 ACU.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
```

```
}
```

La instancia de base de datos está inactiva y se escala a 8 ACU. La siguiente es la configuración relacionada con la capacidad en la instancia de base de datos en este momento. Para representar el tamaño del grupo de búferes en unidades fácilmente legibles, lo dividimos por 2 a la potencia de 30, lo que da lugar a una medición en gibibytes (GiB). Esto es así porque las mediciones relacionadas con la memoria de Aurora utilizan unidades basadas en potencias de 2, no de 10.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|                9294577664 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|   8.65625 |
+-----------+
1 row in set (0.00 sec)
```

A continuación, cambiamos el rango de capacidad del clúster. Una vez que finalice el comando `modify-db-cluster`, el rango de ACU para el clúster pasa de 12,5 a 80.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
```

```
    "MaxCapacity": 80.0
}
```

El cambio del rango de capacidad provocó cambios en los valores predeterminados de algunos parámetros de configuración. Aurora puede aplicar algunos de esos nuevos valores predeterminados de manera inmediata. No obstante, algunos de los cambios en los parámetros surten efecto solo después de un reinicio. El estado `pending-reboot` indica que es necesario reiniciar para aplicar algunos cambios en los parámetros.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

En este punto, el clúster está inactivo y la instancia de base de datos `serverless-v2-instance-1`consume 12,5 ACU. El siguiente ejemplo muestra cómo el parámetro `innodb_buffer_pool_size` ya se ha ajustado en función de la capacidad actual de la instancia de base de datos. El parámetro `max_connections` sigue reflejando el valor del rango de capacidad máxima anterior. Para restablecer ese valor es necesario reiniciar la instancia de base de datos.

> **Note**
>
> Si establece el parámetro `max_connections` directamente en un grupo de parámetros de base de datos personalizado, no será necesario reiniciar.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
```

```
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               15572402176 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+---------------+
| gibibytes     |
+---------------+
| 14.5029296875 |
+---------------+
1 row in set (0.00 sec)
```

Ahora reiniciamos la instancia de base de datos y esperamos a que vuelva a estar disponible.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

El estado `pending-reboot` se borra. El valor `in-sync` confirma que Aurora ha aplicado todos los cambios de parámetros pendientes.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
```

```
}
```

El parámetro `innodb_buffer_pool_size` ha aumentado a su tamaño final para una instancia de base de datos inactiva. El parámetro `max_connections` ha aumentado para reflejar un valor derivado del valor de ACU máximo. La fórmula que utiliza Aurora para `max_connections` provoca un aumento de 1000 cuando el tamaño de la memoria se duplica.

```
mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               16139681792 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|  15.03125 |
+-----------+
1 row in set (0.00 sec)

mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              4000 |
+-------------------+
1 row in set (0.00 sec)
```

Establecemos el rango de capacidad entre 0,5 y 128 ACU y reiniciamos la instancia de base de datos. Ahora la instancia de base de datos inactiva tiene un tamaño de caché de búfer inferior a 1 GiB, por lo que la medimos en mebibytes (MiB). El valor `max_connections` de 5000 se deriva del tamaño de memoria de la configuración de capacidad máxima.

```
mysql> select @@innodb_buffer_pool_size / pow(2,20) as mebibytes, @@max_connections;
+-----------+-------------------+
| mebibytes | @@max_connections |
+-----------+-------------------+
|       672 |              5000 |
+-----------+-------------------+
```

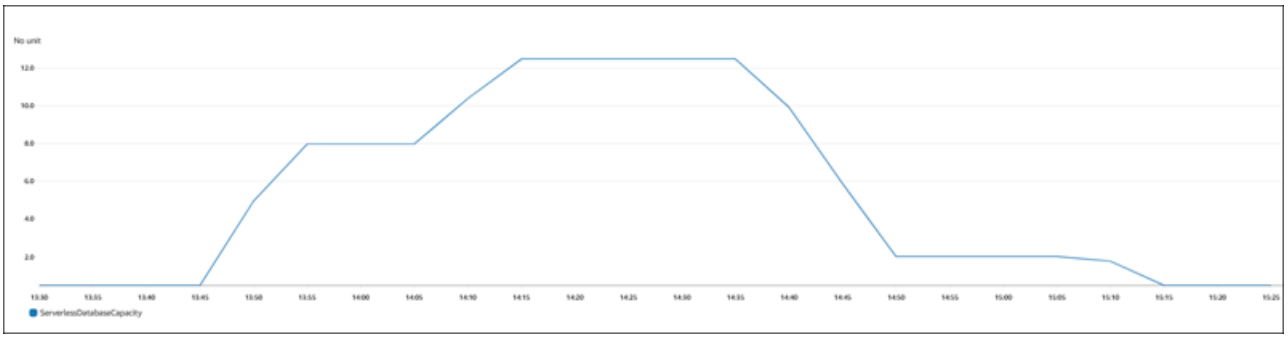
```
1 row in set (0.00 sec)
```

Ejemplo: Cambiar el rango de capacidad Aurora Serverless v2 de un clúster de Aurora PostgreSQL

Los siguientes ejemplos de la CLI muestran cómo actualizar el rango de ACU para instancias de base de datos de Aurora Serverless v2 en un clúster de Aurora PostgreSQL existente.

1. El rango de capacidad del clúster comienza en 0,5 a 1 ACU.
2. Cambie el rango de capacidad a 8-32 ACU.
3. Cambie el rango de capacidad a 12,5–80 ACU.
4. Cambie el rango de capacidad a 0,5–128 ACU.
5. Devuelva la capacidad a su rango inicial de 0,5-1 ACU.

En el siguiente gráfico, se muestran los cambios de capacidad en Amazon CloudWatch.



La instancia de base de datos está inactiva y se escala a 0,5 ACU. La siguiente es la configuración relacionada con la capacidad en la instancia de base de datos en este momento.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

A continuación, cambiamos el rango de capacidad del clúster. Una vez que finalice el comando `modify-db-cluster`, el rango de ACU para el clúster es 8,0–32.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

El cambio del rango de capacidad provoca cambios en los valores predeterminados de algunos parámetros de configuración. Aurora puede aplicar algunos de esos nuevos valores predeterminados de manera inmediata. No obstante, algunos de los cambios en los parámetros surten efecto solo después de un reinicio. El estado `pending-reboot` indica que es necesario reiniciar para aplicar algunos cambios en los parámetros.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

En este punto, el clúster está inactivo y la instancia de base de datos `serverless-v2-instance-1` consume 8,0 ACU. El siguiente ejemplo muestra cómo el parámetro `shared_buffers` ya se ha ajustado en función de la capacidad actual de la instancia de base de datos. El parámetro `max_connections` sigue reflejando el valor del rango de capacidad máxima anterior. Para restablecer ese valor es necesario reiniciar la instancia de base de datos.

> **Note**
>
> Si establece el parámetro `max_connections` directamente en un grupo de parámetros de base de datos personalizado, no será necesario reiniciar.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Reiniciamos la instancia de base de datos y esperamos a que vuelva a estar disponible.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Ahora que la instancia de base de datos se ha reiniciado, el estado `pending-reboot` queda borrado. El valor `in-sync` confirma que Aurora ha aplicado todos los cambios de parámetros pendientes.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Tras el reinicio, `max_connections` muestra el valor de la nueva capacidad máxima.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

A continuación, cambiamos el rango de capacidad del clúster a 12,5–80 ACU.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

En este punto, el clúster está inactivo y la instancia de base de datos `serverless-v2-instance-1`consume 12,5 ACU. El siguiente ejemplo muestra cómo el parámetro `shared_buffers` ya se ha ajustado en función de la capacidad actual de la instancia de base de datos. El valor de `max_connections` sigue siendo 5000.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Reiniciamos de nuevo, pero los valores de los parámetros siguen siendo los mismos. Esto se debe a que `max_connections` tiene un valor máximo de 5000 para un clúster de base de datos de Aurora Serverless v2 que ejecuta Aurora PostgreSQL.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Ahora establecemos el rango de capacidad entre 0,5 y 128 ACU. El clúster de base de datos se reduce verticalmente a 10 ACU y, luego, a 2. Reiniciamos la instancia de base de datos.

```
postgres=> show max_connections;
 max_connections
-----------------
 2000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

El valor de `max_connections` de las instancias de base de datos de Aurora Serverless v2 se basa en el tamaño de memoria obtenido del número máximo de ACU. Sin embargo, cuando especifique una capacidad mínima de 0 o 0,5 ACU en instancias de base de datos compatibles con PostgreSQL, el valor máximo de `max_connections` está limitado a 2000.

Ahora devolvemos la capacidad a su rango inicial de 0,5 a 1 ACU y reiniciamos la instancia de base de datos. El parámetro `max_connections` ha recuperado su valor original.

```
postgres=> show max_connections;
 max_connections
```

```
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Trabajo con los grupos de parámetros para Aurora Serverless v2

Cuando crea el clúster de bases de datos de Aurora Serverless v2, elige un motor de base de datos de Aurora y un grupo de parámetros de clúster de bases de datos asociado. Si no está familiarizado con cómo Aurora utiliza los grupos de parámetros para aplicar la configuración de forma coherente en clústeres, consulte Grupos de parámetros para Amazon Aurora. Todos estos procedimientos para crear, modificar, aplicar y otras acciones para grupos de parámetros son aplicables a Aurora Serverless v2.

La característica de grupo de parámetros suele funcionar igual entre clústeres aprovisionados y clústeres que contienen instancias de base de datos Aurora Serverless v2:

- Los valores de los parámetros predeterminados de todas las instancias de base de datos en el clúster los define el grupo de parámetros del clúster.
- Puede anular algunos parámetros para instancias de base de datos concretas especificando un grupo de parámetros de base de datos personalizado para esas instancias de base de datos. Puede hacerlo al ajustar las opciones de depuración o de rendimiento de instancias de base de datos específicas. Por ejemplo, suponga que tiene un clúster que contiene algunas instancias de base de datos Aurora Serverless v2 y algunas instancias de base de datos aprovisionadas. En este caso, puede especificar algunos parámetros diferentes para las instancias de base de datos aprovisionadas con un grupo de parámetros de base de datos personalizado.
- Para Aurora Serverless v2, podría utilizar todos los parámetros que tengan el valor `provisioned` en el atributo `SupportedEngineModes` del grupo de parámetros. En Aurora Serverless v1 solo puede utilizar el subconjunto de parámetros que tienen `serverless` en el atributo `SupportedEngineModes`.

Temas

- Valores de parámetros predeterminados

- Número máximo de conexiones para Aurora Serverless v2
- Parámetros que Aurora ajusta cuando se escala Aurora Serverless v2 a más o a menos
- Parámetros en los que Aurora hace sus cómputos basándose en la capacidad máxima de Aurora Serverless v2

Valores de parámetros predeterminados

Una diferencia crucial entre las instancias de base de datos aprovisionadas y las instancias de base de datos Aurora Serverless v2 es que Aurora anula cualquier valor de parámetro personalizado para determinados parámetros relacionados con la capacidad de las instancias de base de datos. Los valores de los parámetros personalizados se seguirán aplicando a cualquier instancia de base de datos aprovisionada del clúster. Para obtener más información acerca de cómo interpretan las instancias de base de datos Aurora Serverless v2 los parámetros de los grupos de parámetros Aurora, consulte Parámetros de configuración de los clústeres de Aurora. Para los parámetros específicos que Aurora Serverless v2 anula, consulte Parámetros que Aurora ajusta cuando se escala Aurora Serverless v2 a más o a menos y Parámetros en los que Aurora hace sus cómputos basándose en la capacidad máxima de Aurora Serverless v2.

Puede ver la lista de valores predeterminados de los grupos de parámetros predeterminados para los diferentes motores de base de datos de Aurora mediante el comando describe-db-cluster-parameters de la CLI y consultando la Región de AWS. A continuación se indican los valores que puede utilizar para las opciones `--db-parameter-group-family` y `-db-parameter-group-name` para versiones de motor compatibles con Aurora Serverless v2.

Motor de base de datos y versión	Familia de grupos de parámetros	Nombre del grupo de parámetros predeterminado
Aurora MySQL versión 3	`aurora-mysql8.0`	`default.aurora-mysql8.0`
Versión 13.x de Aurora PostgreSQL	`aurora-postgresql13`	`default.aurora-postgresql13`
Versión 14.x de Aurora PostgreSQL	`aurora-postgresql14`	`default.aurora-postgresql14`

Motor de base de datos y versión	Familia de grupos de parámetros	Nombre del grupo de parámetros predeterminado
Versión 15.x de Aurora PostgreSQL	`aurora-postgresql15`	`default.aurora-postgresql15`
Versión 16.x de Aurora PostgreSQL	`aurora-postgresql16`	`default.aurora-postgresql16`
Versión 17.x de Aurora PostgreSQL	`aurora-postgresql17`	`default.aurora-postgresql17`

En el ejemplo siguiente se obtiene una lista de parámetros del grupo de clústeres de base de datos predeterminado para Aurora MySQL versión 3 y Aurora PostgreSQL 13. Estas son las versiones de Aurora MySQL y Aurora PostgreSQL que se usan con Aurora Serverless v2.

Para Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql8.0 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text

aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-postgresql13 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text
```

Para Windows:

```
aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-mysql8.0 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text
```

```
aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-postgresql13 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text
```

Número máximo de conexiones para Aurora Serverless v2

Para Aurora MySQL y Aurora PostgreSQL, las instancias de base de datos Aurora Serverless v2 contienen constante el parámetro `max_connections` para que las conexiones no se pierdan cuando la instancia de base de datos se escala a menos. El valor predeterminado de este parámetro se deriva de una fórmula que se basa en el tamaño de memoria de la instancia de base de datos. Para obtener más información sobre la fórmula y los valores predeterminados de las clases de instancia de base de datos aprovisionadas, consulte Número máximo de conexiones a una instancia de base de datos Aurora MySQL y Número máximo de conexiones a una instancia de base de datos Aurora PostgreSQL.

Cuando Aurora Serverless v2 evalúa la fórmula, utiliza el tamaño de memoria en función de las unidades de capacidad máxima de Aurora (ACU) para la instancia de base de datos, no en el valor ACU actual. Si cambia el valor predeterminado, le recomendamos utilizar una variación de la fórmula en lugar de especificar un valor constante. De esa forma, Aurora Serverless v2 puede utilizar un ajuste del tamaño adecuado basado en la capacidad máxima.

Al cambiar la capacidad máxima de un clúster de base de datos de Aurora Serverless v2, debe reiniciar las instancias de base de datos de Aurora Serverless v2 para actualizar el valor de `max_connections`. Esto se debe a que `max_connections` es un parámetro estático de Aurora Serverless v2.

La siguiente tabla muestra los valores predeterminados de `max_connections` para Aurora Serverless v2 en función del valor máximo de ACU.

Cantidad máxima de ACU	Conexiones máximas predeterminadas en Aurora MySQL	Conexiones máximas predeterminadas en Aurora PostgreSQL
1	90	189

Cantidad máxima de ACU	Conexiones máximas predeterminadas en Aurora MySQL	Conexiones máximas predeterminadas en Aurora PostgreSQL
4	135	823
8	1 000	1669
16	2,000	3360
32	3000	5 000
64	4.000	5 000
128	5 000	5 000
192	6000	5 000
256	6000	5 000

> **Note**
>
> El valor de `max_connections` de las instancias de base de datos de Aurora Serverless v2 se basa en el tamaño de memoria obtenido del número máximo de ACU. Sin embargo, cuando especifique una capacidad mínima de 0 o 0,5 ACU en instancias de base de datos compatibles con PostgreSQL, el valor máximo de `max_connections` está limitado a 2000.

Para ver ejemplos específicos que muestran cómo `max_connections` cambia con el valor máximo de ACU, consulte Ejemplo: Cambiar el rango de capacidad de Aurora Serverless v2 de un clúster de Aurora MySQL y Ejemplo: Cambiar el rango de capacidad Aurora Serverless v2 de un clúster de Aurora PostgreSQL.

Parámetros que Aurora ajusta cuando se escala Aurora Serverless v2 a más o a menos

Durante el escalado automático, Aurora Serverless v2 necesita poder cambiar los parámetros de cada instancia de base de datos para que funcione mejor para el aumento o la disminución de la capacidad. Por lo tanto, no puede anular algunos parámetros relacionados con la capacidad.

En algunos parámetros que sí se pueden anular, evite codificar valores fijos. Tenga en cuenta lo siguiente en cuanto a esta configuración relacionada con la capacidad.

En Aurora MySQL, Aurora Serverless v2 cambia el tamaño de algunos parámetros dinámicamente durante el escalado. En los parámetros siguientes, Aurora Serverless v2 no utiliza ningún valor de parámetro personalizado que especifique el usuario:

- `innodb_buffer_pool_size`
- `innodb_purge_threads`
- `table_definition_cache`
- `table_open_cache`

En Aurora PostgreSQL, Aurora Serverless v2 cambia el tamaño del siguiente parámetro de forma dinámica durante el escalado. En los parámetros siguientes, Aurora Serverless v2 no utiliza ningún valor de parámetro personalizado que especifique el usuario:

- `shared_buffers`

Para todos los parámetros distintos de los enumerados aquí, las instancias de base de datos de Aurora Serverless v2 funcionan de la misma manera que las instancias de base de datos aprovisionadas. El valor predeterminado del parámetro se hereda del grupo de parámetros del clúster. Puede modificar el valor predeterminado de todo el clúster mediante un grupo de parámetros de clúster personalizado. O bien, puede modificar el valor predeterminado para determinadas instancias de base de datos mediante un grupo de parámetros de base de datos personalizado. Los parámetros dinámicos se actualizan de inmediato. Los cambios en los parámetros estáticos solo surten efecto después de reiniciar la instancia de base de datos.

Parámetros en los que Aurora hace sus cómputos basándose en la capacidad máxima de Aurora Serverless v2

Para los siguientes parámetros, Aurora PostgreSQL utiliza valores predeterminados derivados del tamaño de memoria basado en la configuración máxima de ACU, al igual que con `max_connections`:

- `autovacuum_max_workers`
- `autovacuum_vacuum_cost_limit`
- `autovacuum_work_mem`

- `effective_cache_size`
- `maintenance_work_mem`

Evitar errores de memoria insuficiente

Si una de sus instancias de base de datos Aurora Serverless v2 alcanza constantemente el límite de su capacidad máxima, Aurora lo indica estableciendo la instancia de base de datos en un estado de `incompatible-parameters`. Si bien la instancia de base de datos tiene el estado `incompatible-parameters`, algunas operaciones quedan bloqueadas. Por ejemplo, no se puede actualizar la versión del motor.

Normalmente, la instancia de base de datos entra en este estado cuando se reinicia con frecuencia debido a errores de falta de memoria. Aurora graba un evento cuando se produce este tipo de reinicio. Puede ver el evento siguiendo el procedimiento en Consulta de eventos de Amazon RDS. Los casos de uso de memoria inusualmente elevados pueden producirse debido a una sobrecarga en el ajuste de configuraciones como la de la información del rendimiento y la autenticación de IAM. También puede provenir de una carga de trabajo pesada en la instancia de base de datos o de la administración de metadatos asociados a un gran número de objetos de esquema.

Si la presión de memoria disminuye para que la instancia de base de datos no alcance su capacidad máxima muy a menudo, Aurora cambia automáticamente el estado de la instancia de base de datos a `available`.

Para recuperarse de esta situación, puede llevar a cabo algunas o todas las acciones siguientes:

- Aumentar el límite inferior de capacidad para instancias de base de datos Aurora Serverless v2 cambiando el valor mínimo de la unidad de capacidad Aurora (ACU) para el clúster. Al hacerlo, se evitan problemas en los que la capacidad de una base de datos inactiva se escala a menos memoria de la necesaria para las características activadas en el clúster. Después de cambiar la configuración de ACU del clúster, reinicie la instancia de base de datos Aurora Serverless v2. Al hacerlo, se evalúa si Aurora puede restablecer el estado de nuevo a `available`.
- Aumentar el límite superior de capacidad para instancias de base de datos Aurora Serverless v2 cambiando el valor máximo de ACU para el clúster. Al hacerlo, se evitan problemas en los que una base de datos ocupada no puede escalar a una capacidad con suficiente memoria para las funciones activadas en el clúster y la carga de trabajo de la base de datos. Después de cambiar la configuración de ACU del clúster, reinicie la instancia de base de datos Aurora Serverless v2. Al hacerlo, se evalúa si Aurora puede restablecer el estado de nuevo a `available`.

- Desactivar los ajustes de configuración que requieren sobrecarga de memoria. Por ejemplo, suponga que tiene activadas característica como AWS Identity and Access Management (IAM), información sobre el rendimiento o replicación de registros binarios de Aurora MySQL, pero no las usa. Si es así, puede desactivarlas. O bien, puede ajustar los valores de capacidad mínima y máxima del clúster para tener en cuenta la memoria utilizada por esas funciones. Para ver instrucciones sobre cómo elegir la configuración de capacidad mínima y máxima, consulte Elegir el rango de capacidad de Aurora Serverless v2 para un clúster de Aurora.
- Reducir la carga de trabajo de la instancia de base de datos. Por ejemplo, puede agregar instancias de base de datos de instancias de lector al clúster para distribuir la carga de las consultas de solo lectura entre más instancias de base de datos.
- Ajustar el código SQL utilizado por la aplicación para utilizar menos recursos. Por ejemplo, puede ver los planes de consulta, comprobar el registro de consultas lentas o ajustar los índices de las tablas. También puede aplicar otros tipos de ajustes de SQL tradicionales.

Métricas importantes de Amazon CloudWatch para Aurora Serverless v2

Para empezar a trabajar con Amazon CloudWatch para su instancia de base de datos Aurora Serverless v2, consulte Visualización registros de Aurora Serverless v2 en Amazon CloudWatch. Para obtener más información acerca de cómo monitorear los clústeres de base de datos de Aurora a través de CloudWatch, consulte Monitoreo de eventos de registro en Amazon CloudWatch.

Puede ver sus instancias de base de datos Aurora Serverless v2 en CloudWatch para supervisar la capacidad consumida por cada instancia de base de datos con la métrica `ServerlessDatabaseCapacity`. También puede supervisar todas las métricas estándar de CloudWatch para Aurora, como `DatabaseConnections` y `Queries`. Para ver la lista completa de métricas de CloudWatch que puede supervisar para Aurora, consulte Métricas de Amazon CloudWatch para Amazon Aurora. Las métricas se dividen en métricas en el nivel de clúster y de instancia, enMétricas de nivel de clúster para Amazon Aurora y Métricas de nivel de instancia para Amazon Aurora.

Es importante supervisar las siguientes métricas en el nivel de instancia de CloudWatch para comprender cómo se escalan a más o a menos las instancias de base de datos Aurora Serverless v2. Todas estas métricas se calculan cada segundo. De esta forma, puede supervisar el estado en uso de sus instancias de base de datos Aurora Serverless v2. Puede configurar alarmas para notificarle si hay alguna de las instancias de base de datos Aurora Serverless v2 se acerca a un umbral de métricas relacionadas con la capacidad. Puede determinar si los ajustes de capacidad

mínima y máxima son apropiados o si necesita ajustarlos. Puede determinar dónde debe centrar sus esfuerzos para optimizar la eficiencia de la base de datos.

- `ServerlessDatabaseCapacity`. Como métrica en el nivel de instancia, informa del número de ACU representadas por la capacidad de instancia de base de datos actual. Como métrica en el nivel de clúster, representa la media de los valores de `ServerlessDatabaseCapacity` de todas las instancias de base de datos Aurora Serverless v2 del clúster. Esta métrica es solo una métrica de nivel de clúster en Aurora Serverless v1. En Aurora Serverless v2, está disponible en el nivel de instancia de base de datos y en el nivel de clúster.
- `ACUUtilization`. Esta métrica es nueva en Aurora Serverless v2. Este valor se representa como un porcentaje. Se calcula como el valor de la métrica `ServerlessDatabaseCapacity` dividida por el valor máximo de ACU del clúster de bases de datos. Tenga en cuenta las siguientes pautas para interpretar esta métrica y tomar medidas:
  - Si esta métrica se aproxima a un valor de `100.0`, la instancia de base de datos se ha escalado al punto máximo posible. Considere aumentar la configuración máxima de ACU para el clúster. De este modo, las instancias de base de datos de escritor y lector podrán escalarse a una mayor capacidad.
  - Supongamos que una carga de trabajo de solo lectura hace que una instancia de base de datos de lector se acerque a un valor `ACUUtilization` de `100.0`, mientras que la instancia de base de datos de escritor no está cerca de su capacidad máxima. En ese caso, considere agregar instancias de base de datos de lector adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura de la carga de trabajo distribuida en más instancias de base de datos, reduciendo la carga en cada instancia de base de datos de lector.
  - Supongamos que está ejecutando una aplicación de producción, donde el rendimiento y la escalabilidad son las principales consideraciones. En ese caso, puede establecer el valor máximo de ACU del clúster en un número elevado. Su objetivo es que la métrica `ACUUtilization` quede siempre por debajo de `100.0`. Con un valor de ACU máximo alto, puede estar seguro de tener suficiente espacio en caso de que haya picos inesperados en la actividad de la base de datos. Solo se le cobrará por la capacidad de base de datos que se consuma realmente.
- `CPUUtilization`. Esta métrica se interpreta de forma diferente en Aurora Serverless v2 que en las instancias de base de datos aprovisionadas. Para Aurora Serverless v2, este valor es un porcentaje que se calcula como la cantidad de CPU que se utiliza actualmente dividida por la capacidad de CPU disponible bajo el valor máximo de ACU del clúster de bases de datos. Aurora supervisa este valor automáticamente y escala hacia arriba su instancia de base de datos Aurora

Serverless v2 cuando esta utiliza sistemáticamente una elevada proporción de su capacidad de CPU.

Si esta métrica se aproxima a un valor de `100.0`, la instancia de base de datos ha alcanzado su capacidad máxima de CPU. Considere aumentar la configuración máxima de ACU para el clúster. Si esta métrica se aproxima a un valor de `100.0` en una instancia de base de datos de lector, considere agregar instancias de base de datos de lector adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura de la carga de trabajo distribuida en más instancias de base de datos, reduciendo la carga en cada instancia de base de datos de lector.

- `FreeableMemory`. Este valor representa la cantidad de memoria sin utilizar que está disponible cuando la instancia de base de datos Aurora Serverless v2 se escala a su capacidad máxima. Por cada ACU en cuya capacidad actual esté por debajo de la capacidad máxima, este valor aumenta aproximadamente 2 GiB. Por lo tanto, esta métrica no se aproxima a cero hasta que la instancia de base de datos se amplía al máximo posible.

  Si esta métrica se aproxima a un valor de `0`, la instancia de base de datos se ha ampliado tanto como puede y se acerca al límite de memoria disponible. Considere aumentar la configuración máxima de ACU para el clúster. Si esta métrica se aproxima a un valor de `0` en una instancia de base de datos de lector, considere agregar instancias de base de datos de lector adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura entre más instancias de base de datos, reduciendo el uso de memoria en cada instancia de base de datos de lector.
- `TempStorageIOPS`. El número de IOPS realizadas en el almacenamiento local adjuntas a la instancia de base de datos. Incluye las IOPS de lecturas y escrituras. Esta métrica representa un recuento y se mide una vez por segundo. Esta es una nueva métrica en Aurora Serverless v2. Para obtener más información, consulte Métricas de nivel de instancia para Amazon Aurora.
- `TempStorageThroughput`. La cantidad de datos transferidos desde y hacia el almacenamiento local asociado a la instancia de base de datos. Esta métrica representa un número de bytes y se mide una vez por segundo. Esta es una nueva métrica en Aurora Serverless v2. Para obtener más información, consulte Métricas de nivel de instancia para Amazon Aurora.

Por lo general, la mayoría de los escalados a más para instancias de base de datos Aurora Serverless v2 son por el uso de la memoria y por la actividad. Las métricas `TempStorageIOPS` y `TempStorageThroughput` pueden ayudarle a diagnosticar los raros casos en los que la actividad de red para transferencias entre la instancia de base de datos y los dispositivos de almacenamiento locales es responsable de aumentos de capacidad no esperados. Para supervisar otra actividad de red, puede utilizar estas métricas existentes:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Puede hacer que Aurora publique algunos registros de base de datos o todos en Registros de Amazon CloudWatch. Para obtener instrucciones, consulte lo siguiente, en función del motor de base de datos:

- the section called “Publicación de registros de Aurora PostgreSQL en CloudWatch Logs”
- the section called “Publicación de registros de Aurora MySQL en CloudWatch Logs”

Cómo se aplican las métricas de Aurora Serverless v2 en la factura de AWS

Los cargos correspondientes a Aurora Serverless v2 en la factura de AWS se calcula sobre la base de la misma métrica de `ServerlessDatabaseCapacity` que el usuario puede supervisar. El mecanismo de facturación puede diferir de la media computada de CloudWatch para esta métrica en los casos en los que se utilice capacidad de Aurora Serverless v2 durante solo una parte de una hora. También puede variar si los problemas del sistema hacen que la métrica de CloudWatch no esté disponible durante periodos breves. Por lo tanto, es posible que vea un valor ligeramente diferente de las horas de ACU en su factura que si computa el número usted mismo a partir del valor de `ServerlessDatabaseCapacity` promedio.

Ejemplos de comandos de CloudWatch para métricas de Aurora Serverless v2

Los siguientes ejemplos de AWS CLI muestran cómo se pueden supervisar las métricas de CloudWatch más importantes relacionadas con Aurora Serverless v2. En cada caso, sustituya la cadena `Value=` para el parámetro `--dimensions` con el identificador de su propia instancia de base de datos Aurora Serverless v2.

En el siguiente ejemplo de Linux se muestran los valores de capacidad mínima, máxima y media de una instancia de base de datos, medidos cada 10 minutos durante una hora. El comando de Linux `date` especifica las horas de inicio y finalización en relación con la fecha y la hora en curso. La

función `sort_by` en el parámetro `--query` ordena los resultados cronológicamente basándose en el campo `Timestamp`.

```
aws cloudwatch get-metric-statistics --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Los siguientes ejemplos de Linux muestran la supervisión de la capacidad de cada instancia de base de datos de un clúster. Miden el uso de capacidad mínimo, máximo y medio de cada instancia de base de datos. Las mediciones se toman una vez por hora durante un período de tres horas. Estos ejemplos utilizan la métrica `ACUUtilization`, la cual representa un porcentaje del límite superior en las ACU, en lugar de `ServerlessDatabaseCapacity`, que representa un número fijo de ACU. De esta forma, no necesita conocer los números reales de los valores de ACU mínima y máxima en el rango de capacidad. Puede ver porcentajes que oscilan entre 0 y 100.

```
aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_writer_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table

aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_reader_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

En el siguiente ejemplo de Linux se hacen mediciones similares a las anteriores. En este caso, las medidas corresponden a la métrica `CPUUtilization`. Las mediciones se toman cada 10 minutos durante un período de 1 hora. Los números representan el porcentaje de CPU disponible utilizada, en función de los recursos de CPU disponibles para la configuración de capacidad máxima de la instancia de base de datos.

```
aws cloudwatch get-metric-statistics --metric-name "CPUUtilization" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
```

```
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

En el siguiente ejemplo de Linux se hacen mediciones similares a las anteriores. En este caso, las medidas corresponden a la métrica `FreeableMemory`. Las mediciones se toman cada 10 minutos durante un período de 1 hora.

```
aws cloudwatch get-metric-statistics --metric-name "FreeableMemory" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Supervisión del rendimiento de Aurora Serverless v2 con la información de rendimiento

Puede utilizar Performance Insights (Información de rendimiento) para supervisar el rendimiento de las instancias de base de datos Aurora Serverless v2 Para conocer los procedimientos de información del rendimiento, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

Los siguientes son contadores de información del rendimiento nuevos aplicables a instancias de base de datos Aurora Serverless v2

- `os.general.serverlessDatabaseCapacity`: la capacidad actual de la instancia de base de datos en ACU. El valor corresponde a la métrica de CloudWatch `ServerlessDatabaseCapacity` para la instancia de base de datos.
- `os.general.acuUtilization`: porcentaje de la capacidad actual fuera de la capacidad máxima configurada. El valor corresponde a la métrica de CloudWatch `ACUUtilization` para la instancia de base de datos.
- `os.general.maxConfiguredAcu`: la capacidad máxima que ha configurado para esta instancia de base de datos Aurora Serverless v2. Se mide en ACU.
- `os.general.minConfiguredAcu`: la capacidad máxima que ha configurado para esta instancia de base de datos Aurora Serverless v2. Se mide en ACU.

Para ver la lista completa de contadores de información de rendimiento, consulte Métricas de contador de Información de rendimiento.

Cuando se muestran los valores de vCPU para una instancia de base de datos Aurora Serverless v2 en la información de rendimiento, estos valores representan estimaciones basadas en el valor de ACU de la instancia de base de datos. En el intervalo predeterminado de un minuto, los valores fraccionarios de vCPU se redondean al número entero más cercano. En intervalos más largos, el valor de vCPU que se muestra es el promedio de los valores de vCPU enteros de cada minuto.

Solución de problemas de capacidad de Aurora Serverless v2

En algunos casos, Aurora Serverless v2 no se reduce verticalmente a la capacidad mínima, incluso sin carga en la base de datos. Esto puede suceder por una de las siguientes razones:

- Algunas características pueden aumentar el uso de los recursos e impedir que la base de datos se reduzca verticalmente a su capacidad mínima. Estas son algunas de ellas:
  - Bases de datos globales de Aurora
  - Exportación de registros de CloudWatch
  - Habilitación de `pg_audit` en clústeres de base de datos compatibles con Aurora PostgreSQL
  - Supervisión mejorada
  - Información de rendimiento

  Para obtener más información, consulte Elegir la configuración de capacidad de Aurora Serverless v2 mínima para un clúster.
- Si una instancia de lector no se reduce verticalmente al mínimo y permanece en la misma capacidad o más que la instancia de escritor, compruebe el nivel de prioridad de la instancia de lector. Las instancias de base de datos de lector de Aurora Serverless v2 en los niveles 0 o 1 se mantienen a una capacidad mínima al menos tan alta como la instancia de base de datos de escritor. Cambie el nivel de prioridad del lector a 2 o más para que se escale y se reduzca verticalmente independientemente del escritor. Para obtener más información, consulte Elegir el nivel de promoción para un lector Aurora Serverless v2.
- Defina los parámetros de la base de datos que afecten al tamaño de la memoria compartida en sus valores predeterminados. Si se establece un valor superior al predeterminado, se aumenta el requisito de memoria compartida y se evita que la base de datos se reduzca verticalmente a la capacidad mínima. Algunos ejemplos son `max_connections` y `max_locks_per_transaction`.

> **Note**
>
> La actualización de los parámetros de memoria compartida requiere reiniciar la base de datos para que los cambios surtan efecto.

- Las cargas de trabajo pesadas de las bases de datos pueden aumentar el uso de recursos.
- Los grandes volúmenes de bases de datos pueden aumentar el uso de recursos.

  Amazon Aurora utiliza recursos de memoria y CPU para la administración de clústeres de bases de datos. Aurora requiere más CPU y memoria para administrar los clústeres de bases de datos con volúmenes de bases de datos más grandes. Si la capacidad mínima del clúster es inferior al mínimo requerido para la administración del clúster, el clúster no se reducirá verticalmente a la capacidad mínima.
- Los procesos en segundo plano, como la purga, también pueden aumentar el uso de los recursos.

Si la base de datos sigue sin reducir verticalmente la capacidad mínima configurada, deténgala y reiníciela para recuperar los fragmentos de memoria que se hayan acumulado con el tiempo. Al detener e iniciar una base de datos, se produce un tiempo de inactividad, por lo que recomendamos hacerlo con moderación.

Escalado a cero ACU con pausa y reanudación automáticas para Aurora Serverless v2

Puede especificar que las instancias de base de datos de Aurora Serverless v2 se reduzcan verticalmente hasta cero ACU y que se pausen automáticamente si no tienen ninguna conexión iniciada por la actividad del usuario en un período de tiempo específico. Para ello, especifique un valor mínimo de ACU igual a cero para su clúster de base de datos. Mientras la instancia esté en pausa, no se le cobrará por la capacidad de las instancias. Habilitar la característica de pausa y reanudación automáticas (pausa automática) para los clústeres de Aurora que se utilizan poco o tienen períodos prolongados de inactividad puede ayudarlo a administrar los costos de su flota de bases de datos.

> **Note**
>
> La característica de pausa automática está disponible para Aurora Serverless v2 tanto con Aurora PostgreSQL como con Aurora MySQL. Es posible que necesite actualizar la versión del motor de base de datos de Aurora para aprovechar esta característica. Para ver las versiones de motor en las que hay disponible una capacidad mínima de 0 ACU, consulte Capacidad de Aurora Serverless v2.

Temas

- Descripción general de la característica de pausa automática de Aurora Serverless v2
- Requisitos previos y limitaciones de la característica de pausa automática de Aurora Serverless v2
- Activación y desactivación de la característica de pausa automática
- Funcionamiento de la característica de pausa automática de Aurora Serverless v2
- Funcionamiento de la pausa automática de Aurora Serverless v2 para diferentes tipos de clústeres de Aurora
- Supervisión de los clústeres de Aurora que utilizan la pausa automática
- Solución de problemas para la característica de pausa automática de Aurora Serverless v2
- Consideraciones sobre el diseño de la aplicación para la característica de pausa automática de Aurora Serverless v2

Descripción general de la característica de pausa automática de Aurora Serverless v2

Las instancias de base de datos de Aurora Serverless v2 pueden pausarse automáticamente después de un período sin conexiones de usuario y reanudarse automáticamente cuando llega una solicitud de conexión. La característica de pausa y reanudación automática de Aurora Serverless v2 ayuda a administrar los costos de los sistemas que no tienen un objetivo de nivel de servicio (SLO) estricto. Por ejemplo, puede habilitar esta característica para los clústeres que se utilizan para el desarrollo y las pruebas, o para las aplicaciones internas en las que se admite una breve pausa mientras se reanuda la base de datos. Si su carga de trabajo tiene períodos de inactividad y puede tolerar pequeños retrasos en la conexión mientras se reanuda la instancia, considere la posibilidad de utilizar la pausa automática con las instancias de Aurora Serverless v2 para reducir los costos.

Para controlar este comportamiento, especifique si las instancias de base de datos de Aurora Serverless v2 de un clúster se pueden pausar automáticamente o no, y cuánto tiempo debe permanecer inactiva cada instancia antes de que se detenga. Para habilitar el comportamiento de pausa automática para todas las instancias de base de datos de Aurora Serverless v2 de un clúster de Aurora, establezca el valor de capacidad mínima del clúster en cero ACU.

Si anteriormente utilizaba la característica de Aurora Serverless v1 que escalaba hasta cero ACU tras un período de inactividad, puede actualizar Aurora Serverless v2 y utilizar la característica de pausa automática correspondiente.

Los beneficios de ahorro de costos de la característica de pausa automática son similares a los del uso de la característica de parar o iniciar el clúster. La pausa automática de Aurora Serverless v2 tiene las ventajas adicionales de una reanudación más rápida que iniciar un clúster detenido y de automatizar el proceso de determinar cuándo pausar y reanudar cada instancia de base de datos.

La característica de pausa automática también proporciona mayor grado de detalle a la hora de controlar los costos de los recursos de computación del clúster. Puede activar algunas instancias de lector para que se detengan incluso mientras la instancia de escritor y otros lectores del clúster permanecen activos en todo momento. Para ello, asigne a las instancias de lector que se pueden pausar independientemente de otras instancias una prioridad de conmutación por error del orden de 2 a 15.

Las instancias de escritor y todas las instancias de lector con prioridad de conmutación por error de 0 y 1 siempre se pausan y se reanudan al mismo tiempo. Por lo tanto, las instancias de este grupo se detienen cuando ninguna de ellas tiene conexión durante el intervalo de tiempo especificado.

Los clústeres de base de datos de Aurora pueden contener una combinación de instancias de base de datos de escritor y de lector e instancias de base de datos de Aurora Serverless v2 y aprovisionadas. Por lo tanto, para utilizar esta característica de forma eficaz, es útil comprender los siguientes aspectos del mecanismo de pausa automática:

- las circunstancias en las que una instancia de base de datos puede pausarse automáticamente,
- cuándo se puede impedir que una instancia de base de datos se detenga. Por ejemplo, habilitar algunas características de Aurora o realizar ciertos tipos de operaciones en el clúster puede evitar que las instancias se detengan, incluso sin ninguna conexión a esas instancias.
- las consecuencias para la supervisión y la facturación mientras una instancia está pausada,
- qué acciones hacen que una instancia de base de datos reanude el procesamiento,
- cómo cambia la capacidad de una instancia de base de datos en torno al momento de la pausa y la reanudación de los eventos,
- cómo controlar el intervalo de inactividad antes de que una instancia de base de datos se pause,
- cómo codificar la lógica de la aplicación para gestionar el período durante el cual una instancia de base de datos reanuda el procesamiento.

Requisitos previos y limitaciones de la característica de pausa automática de Aurora Serverless v2

Antes de utilizar la característica de pausa automática, compruebe qué versiones de motor debe utilizar. Además, debe comprobar si la pausa automática funciona en combinación con las demás características de Aurora que desee utilizar. No puede activar la pausa automática si utiliza una versión del motor que no la admite. En el caso de las características incompatibles, no obtendrá ningún error si las utiliza en combinación con la pausa automática. Si el clúster utiliza características o ajustes incompatibles, las instancias de Aurora Serverless v2 no se pausarán automáticamente.

- Si utiliza Aurora PostgreSQL, el motor de base de datos debe ejecutar al menos las versiones 16.3, 15.7, 14.12 o 13.15.
- Si utiliza Aurora MySQL, el motor de base de datos debe ejecutar la versión 3.08.0 o superior.
- Para ver la lista completa de las versiones del motor y las regiones de AWS en las que está disponible esta característica, consulte Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.
- Cuando se reanuda una instancia de Aurora Serverless v2, es posible que su capacidad sea inferior a la que tenía cuando la instancia estaba en pausa. Para obtener más información,

consulte Diferencias en el comportamiento de pausa automática entre Aurora Serverless v2 y Aurora Serverless v1.

Determinadas condiciones o ajustes impiden que las instancias de Aurora Serverless v2 se detengan automáticamente. Para obtener más información, consulte Situaciones en las Aurora Serverless v2 que no se pausa automáticamente.

Activación y desactivación de la característica de pausa automática

Puede activar y desactivar la característica de pausa automática en el nivel de clúster. Para ello, debe utilizar los mismos procedimientos que cuando ajusta la capacidad mínima y máxima del clúster. La característica de pausa automática está representada por una capacidad mínima de 0 ACU.

Temas

- Activación de la pausa automática para las instancias de Aurora Serverless v2 de un clúster
- Intervalo de tiempo de espera de la pausa automática de Aurora Serverless v2 configurable
- Reanudación de una instancia de Aurora Serverless v2 pausada automáticamente
- Desactivación de la pausa automática para las instancias de Aurora Serverless v2 de un clúster

Activación de la pausa automática para las instancias de Aurora Serverless v2 de un clúster

Siga el procedimiento indicado en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster. Para obtener la capacidad mínima, elija 0 ACU. Si elige una capacidad mínima de 0 ACU, también puede especificar el tiempo que debe permanecer inactiva la instancia antes de que se pause automáticamente.

El siguiente ejemplo de la CLI muestra cómo puede crear un clúster de Aurora con la característica de pausa automática habilitada y el intervalo de pausa automática establecido en diez minutos (600 segundos).

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
```

```
    --engine-version 8.0 \
    --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=4,SecondsUntilAutoPause=600 \
    --master-username myuser \
    --manage-master-user-password
```

El siguiente ejemplo de la CLI muestra cómo puede activar la característica de pausa automática para un clúster de Aurora existente. En este ejemplo, se establece el intervalo de pausa automática en una hora (3600 segundos).

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=80,SecondsUntilAutoPause=3600

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 0,
    "MaxCapacity": 80.0,
    "SecondsUntilAutoPause": 3600
}
```

Intervalo de tiempo de espera de la pausa automática de Aurora Serverless v2 configurable

El intervalo de tiempo de espera se representa en el atributo `ServerlessV2ScalingConfiguration` del clúster de Aurora. Puede especificar este intervalo en la Consola de administración de AWS al crear o modificar un clúster de Aurora, si la capacidad mínima está establecida en cero ACU. Puede especificarlo en la AWS CLI con el parámetro `--serverless-v2-scaling-configuration` al crear o modificar un clúster de Aurora. Puede especificarlo en la API de RDS con el parámetro `ServerlessV2ScalingConfiguration` al crear o modificar un clúster de Aurora.

El intervalo mínimo que puede establecer es de 300 segundos (cinco minutos). Es el valor predeterminado si no especifica un intervalo. El intervalo máximo que puede establecer es 86 400 segundos (un día).

Suponga que desactiva la característica de pausa automática de un clúster al cambiar la capacidad mínima del clúster a un valor distinto de cero. En ese caso, la propiedad del intervalo se elimina del atributo `ServerlessV2ScalingConfiguration`. La ausencia de esa propiedad proporciona

una confirmación adicional de que la característica de pausa automática está desactivada para ese clúster. Si más adelante vuelve a activar la pausa automática, podrá volver a especificar cualquier intervalo personalizado en ese momento.

Reanudación de una instancia de Aurora Serverless v2 pausada automáticamente

- Cuando se conecta a una instancia de Aurora Serverless v2 pausada, se reanuda automáticamente y acepta la conexión.
- Si se intenta conectar con credenciales no válidas, la instancia de base de datos se reanudará igualmente.
- Si se conecta a través del punto de conexión del escritor, Aurora reanuda la instancia de base de datos de escritor si se ha pausado automáticamente. Al mismo tiempo, Aurora reanuda cualquier instancia de lector pausada automáticamente que tenga una prioridad de conmutación por error de 0 o 1, lo que significa que su capacidad está vinculada a la capacidad de la instancia de escritor.
- Si se conecta a través del punto de conexión del lector, Aurora elige una instancia de lector de forma aleatoria. Si la instancia del lector está pausada, Aurora la reanuda. Aurora también reanuda primero la instancia de escritor, pues la instancia de escritor debe estar siempre activa si hay alguna instancia de lector activa. Cuando Aurora reanuda esa instancia de escritor, también se reanudan las instancias de lector de los niveles de promoción cero y uno de conmutación por error.
- Si envía una solicitud al clúster a través de la API de datos de RDS, Aurora reanuda la instancia de escritor si está pausada. A continuación, Aurora procesa la solicitud de la API de datos.
- Al cambiar el valor de un parámetro de configuración en un grupo de parámetros de clúster de base de datos, Aurora reanuda automáticamente las instancias de Aurora Serverless v2 pausadas en todos los clústeres que utilizan ese grupo de parámetros de clúster. De igual modo, al cambiar el valor de un parámetro en un grupo de parámetros de base de datos, Aurora reanuda automáticamente las instancias de Aurora Serverless v2 pausadas que utilizan ese grupo de parámetros de base de datos. El mismo comportamiento de reanudación automática se aplica cuando se modifica un clúster para asignar un grupo de parámetros de clúster diferente o cuando se modifica una instancia para asignar un grupo de parámetros de base de datos diferente.
- Al realizar una solicitud de retroceso, se reanuda automáticamente la instancia de escritor de Aurora Serverless v2 si está pausada. Aurora procesa la solicitud de retroceso después de que se reanude la instancia de escritor. Puede retroceder hasta un momento en el que se pausó una instancia de Aurora Serverless v2.
- Si se toma una instantánea de un clúster o se elimina una instantánea, no se reanudará ninguna instancia de Aurora Serverless v2.

- La creación de un clon de Aurora hace que Aurora reanude la instancia de escritor del clúster que se está clonando.
- Si una instancia pausada recibe una gran cantidad de solicitudes de conexión antes de que termine de reanudarse, es posible que algunas sesiones no puedan conectarse. Recomendamos implementar la lógica de reintento para las conexiones a los clústeres de Aurora que tienen algunas instancias de Aurora Serverless v2 con la pausa automática activada. Por ejemplo, puede volver a intentar cualquier conexión fallida tres veces.
- Aurora puede realizar algunos tipos de mantenimiento interno menores sin activar una instancia. Sin embargo, algunos tipos de mantenimiento que se realizan durante el período de mantenimiento del clúster requieren que Aurora reanude la instancia. Cuando finaliza el mantenimiento, la instancia se vuelve a pausar automáticamente si no hay más actividad después del intervalo especificado.

> **Note**
>
> Aurora no reanuda automáticamente una instancia pausada para tareas programadas específicas del motor, como las de la extensión `pg_cron` de PostgreSQL o el programador de eventos de MySQL. Para garantizar que dichos trabajos se ejecuten, debe iniciar una conexión manual con la instancia antes de la hora programada. Aurora no pone en cola ningún trabajo en el que se cumpla la hora programada mientras la instancia de base de datos esté pausada. Estos trabajos se omiten cuando la instancia se reanuda más tarde.

- Si el clúster de Aurora se somete a una conmutación por error mientras una instancia de Aurora Serverless v2 se pausa automáticamente, Aurora podría reanudar una instancia y, a continuación, convertirla en la instancia de escritor. Lo mismo puede ocurrir si se eliminan una o más instancias de base de datos del clúster mientras una instancia está pausada. En este caso, la instancia pasa a ser de escritor inmediatamente después de reanudarse.
- Las operaciones que cambian las propiedades del clúster también hacen que se reanuden las instancias de Aurora Serverless v2 pausadas automáticamente. Por ejemplo, una instancia pausada automáticamente se reanuda para operaciones como las siguientes:
  - Cambiar el rango de escalado del clúster.
  - Actualizar la versión del motor del clúster.
  - Describir o descargar archivos de registro de una instancia pausada. Para examinar los datos de registro históricos de las instancias pausadas, habilite las cargas de registros en CloudWatch y analice los registros a través de CloudWatch.

Desactivación de la pausa automática para las instancias de Aurora Serverless v2 de un clúster

Siga el procedimiento indicado en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster. Para la capacidad mínima, elija un valor de 0,5 o superior. Al desactivar la característica de pausa automática, se restablece el intervalo de inactividad de la instancia. Si vuelve a activar la pausa automática, debe especificar un nuevo intervalo de tiempo de espera.

El siguiente ejemplo de la CLI muestra cómo puede desactivar la característica de pausa automática para un clúster de Aurora existente. El resultado `describe-db-clusters` muestra que el atributo `SecondsUntilAutoPause` se elimina cuando la capacidad mínima se establece en un valor distinto de cero.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=2,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 2,
    "MaxCapacity": 80.0
}
```

Funcionamiento de la característica de pausa automática de Aurora Serverless v2

Puede utilizar la siguiente información para planificar el uso de la característica de pausa automática. Comprender las circunstancias en las que las instancias se detienen, se reanudan o permanecen activas puede ayudarlo a equilibrar las ventajas y desventajas entre disponibilidad, capacidad de respuesta y ahorro de costos.

Temas

- Qué sucede cuando se pausan las instancias de Aurora Serverless v2
- Qué ocurre cuando se reanudan las instancias de Aurora Serverless v2 pausadas automáticamente
- Funcionamiento de la facturación de instancias para clústeres de Aurora Serverless v2 pausados automáticamente
- Situaciones en las Aurora Serverless v2 que no se pausa automáticamente

- Funcionamiento de la pausa automática con la característica de parada o inicio del clúster
- Funcionamiento del mantenimiento y las actualizaciones de los clústeres de Aurora Serverless v2 pausados automáticamente
- Diferencias en el comportamiento de pausa automática entre Aurora Serverless v2 y Aurora Serverless v1

Qué sucede cuando se pausan las instancias de Aurora Serverless v2

Cuando una instancia de base de datos de Aurora Serverless v2 se detiene tras un período sin conexiones:

- Aurora comienza a pausar la instancia una vez transcurrido el intervalo especificado sin conexiones a la instancia, independientemente del número de ACU que tenga la instancia en ese momento.
- El mecanismo de pausa no es instantáneo. Es posible que una instancia de Aurora Serverless v2 que esté a punto de pausarse automáticamente espere un momento para actualizarse con todos los cambios en el almacenamiento de Aurora.
- Los cargos de la instancia correspondientes a esa instancia se ponen en espera. La métrica `ServerlessV2Usage` tiene un valor de 0 mientras la instancia está pausada.
- El valor de estado de la instancia no cambia. El estado sigue apareciendo como disponible.
- La instancia deja de escribir en los archivos de registro de la base de datos. Deja de enviar métricas a CloudWatch, salvo registrar cero por ciento para `CPUUtilization` y `ACUUtilization`, y cero para `ServerlessDatabaseCapacity`.
- Aurora emite eventos cuando una instancia de base de datos de Aurora Serverless v2 comienza la pausa, termina la pausa y si el mecanismo de pausa se interrumpe o no funciona. Para obtener más detalles acerca de estos eventos, consulte Eventos de instancia de base de datos.

Qué ocurre cuando se reanudan las instancias de Aurora Serverless v2 pausadas automáticamente

Cuando una instancia de base de datos de Aurora Serverless v2 se reanuda tras haber sido pausada automáticamente, se aplican las siguientes condiciones:

- Todos los cambios de parámetros que estén incluidos en los cambios de `pending-reboot` se aplicarán cuando se reanude la instancia.

- Aurora emite eventos en el nivel de la instancia cuando cada instancia de base de datos de Aurora Serverless v2 comienza a reanudarse, termina de reanudarse y si la instancia no puede reanudarse por algún motivo. Para obtener más detalles acerca de estos eventos, consulte Eventos de instancia de base de datos.
- Todas las conexiones solicitadas se establecen una vez que la instancia de base de datos termina de reanudarse. Como el tiempo normal de reanudación puede ser de aproximadamente 15 segundos, le recomendamos que ajuste cualquier configuración de tiempo de espera del cliente para que sea superior a 15 segundos. Por ejemplo, en la configuración del controlador JDBC, puede ajustar los valores de la configuración de `connectTimeout` y de `sslResponseTimeout` para que sean superiores a 15 segundos.

> **Note**
>
> Si una instancia de Aurora Serverless v2 permanece pausada durante más de 24 horas, Aurora puede hacer que la instancia entre en un sueño más profundo y que tarde más en reanudarse. En ese caso, el tiempo de reanudación puede ser de 30 segundos o más, aproximadamente el equivalente a reiniciar la instancia. Si su base de datos tiene períodos de inactividad que duran más de un día, le recomendamos configurar los tiempos de espera de conexión en 30 segundos o más.

Funcionamiento de la facturación de instancias para clústeres de Aurora Serverless v2 pausados automáticamente

Mientras una instancia de Aurora Serverless v2 se pausa automáticamente, su cargo por instancia es cero. La métrica `ServerlessV2Usage` es cero durante ese período. AWS sigue cobrando por el almacenamiento de Aurora y otros aspectos del clúster que no están vinculados a esa instancia de base de datos específica.

Situaciones en las Aurora Serverless v2 que no se pausa automáticamente

- Si el valor de capacidad mínimo del clúster de base de datos es superior a cero ACU, las instancias de Aurora Serverless v2 del clúster no se pausan automáticamente. Si tiene clústeres con instancias de Aurora Serverless v2 de antes de que existiera la característica de pausa automática, la configuración de capacidad mínima más baja es de 0,5 ACU. Para utilizar la característica de pausa automática con dichos clústeres, debe modificar la configuración de capacidad mínima a cero ACU.

- Si alguna conexión iniciada por el usuario está abierta a una instancia de Aurora Serverless v2, la instancia no se pausará. La instancia tampoco se pausará mientras se realicen actividades como la aplicación de parches y las actualizaciones. Las conexiones administrativas que Aurora usa para las comprobaciones de estado no cuentan como actividad y no impiden que la instancia se pause.
- En un clúster de Aurora PostgreSQL que tenga habilitada la replicación lógica o en un clúster de Aurora MySQL que tenga habilitada la replicación binlog, la instancia de escritor y cualquier instancia de lector en los niveles cero y uno de promoción de conmutación por error no se pausan de forma automática. Aurora realiza una cantidad mínima constante de actividades para comprobar el estado de la conexión de replicación.

  En el caso de los clústeres con la replicación habilitada, puede seguir teniendo instancias de lector de Aurora en el clúster de origen que se pausa automáticamente. Para ello, establezca su prioridad de conmutación por error en un valor distinto de cero o uno. De esta forma, se pueden pausar independientemente de la instancia de escritor.
- Si el clúster de Aurora tiene un proxy RDS asociado, el proxy mantiene una conexión abierta con cada instancia de base de datos del clúster. Por lo tanto, las instancias de Aurora Serverless v2 de un clúster de este tipo no se pausarán automáticamente.
- Si el clúster es el clúster principal de una base de datos global de Aurora, Aurora no pausa automáticamente la instancia de escritor de Aurora Serverless v2. Esto se debe a que se necesita un nivel de actividad constante en la instancia de escritor para administrar los demás clústeres de la base de datos global. Como la instancia de escritor permanece activa, las instancias de lector de Aurora Serverless v2 con prioridad de conmutación por error cero o uno tampoco se pausan automáticamente.
- Las instancias de Aurora Serverless v2 de los clústeres secundarios de una base de datos global de Aurora no se pausan automáticamente. Si un clúster de base de datos se convierte en un clúster independiente, la característica de pausa automática será efectiva si se cumplen todas las demás condiciones.
- En un clúster con una integración sin ETL asociada a Redshift, la instancia de escritor y cualquier instancia de lector de los niveles cero y uno de promoción de conmutación por error no se pausan de forma automática.
- Además de la actividad en el puerto de base de datos principal de la instancia, si una instancia de Aurora PostgreSQL tiene habilitada la característica Babelfish, cualquier conexión y actividad en el puerto T-SQL impedirá que la instancia se pause automáticamente.

Funcionamiento de la pausa automática con la característica de parada o inicio del clúster

Puede detener e iniciar un clúster de Aurora cuando la característica de pausa automática esté habilitada. No importa si algunas instancias están pausadas. Al volver a iniciar el clúster, las instancias de Aurora Serverless v2 pausadas se reanudan automáticamente.

Mientras un clúster de Aurora está detenido, las instancias de Aurora Serverless v2 pausadas no se reanudan automáticamente en función de los intentos de conexión. Cuando se reinicie el clúster, se aplicarán los mecanismos habituales para pausar y reanudar las instancias de Aurora Serverless v2.

Funcionamiento del mantenimiento y las actualizaciones de los clústeres de Aurora Serverless v2 pausados automáticamente

- Mientras una instancia de Aurora Serverless v2 está pausada automáticamente, si intenta actualizar el clúster de Aurora, Aurora reanuda la instancia y la actualiza.
- Aurora reanuda periódicamente las instancias de Aurora Serverless v2 pausadas automáticamente para realizar tareas de mantenimiento, como actualizaciones de versiones secundarias y cambios en propiedades, como los grupos de parámetros.
- Cuando una instancia de Aurora Serverless v2 se activa para realizar una operación administrativa, como una actualización o una tarea de mantenimiento, Aurora espera al menos 20 minutos antes de volver a pausar la instancia. Esto es para permitir que finalicen las operaciones en segundo plano. El período de veinte minutos también evita que se pause y reanude la instancia varias veces si esta se somete a varias operaciones administrativas seguidas.

Diferencias en el comportamiento de pausa automática entre Aurora Serverless v2 y Aurora Serverless v1

- El tiempo de reanudación ha mejorado en Aurora Serverless v2 en comparación con Aurora Serverless v1. El tiempo de reanudación suele ser de aproximadamente 15 segundos si la instancia se ha pausado durante menos de 24 horas. Si la instancia permanece pausada durante más de 24 horas, es posible que el tiempo de reanudación sea mayor.
- La forma en que Aurora Serverless v2 afecta a los clústeres Multi-AZ implica que algunas instancias de base de datos del clúster pueden estar pausadas mientras que otras están activas. La instancia de escritor se reanuda siempre que se esté ejecutando un lector, ya que es necesario que el escritor coordine determinadas actividades dentro del clúster. Como Aurora Serverless v1 no utiliza instancias de lector, todo el clúster estará siempre en pausa o activo.

- Cuando el punto de conexión del lector selecciona aleatoriamente una instancia de lector a la que conectarse, es posible que esa instancia de lector ya esté activa o que esté pausada automáticamente. Por lo tanto, el tiempo de acceso a la instancia de lector puede variar y ser más difícil de predecir. Por lo tanto, los clústeres Multi-AZ que utilizan Aurora Serverless v2 y se pausan automáticamente podrían beneficiarse de la configuración de puntos de conexión personalizados para casos de uso específicos de solo lectura, en lugar de dirigir todas las sesiones de solo lectura al punto de conexión de lector.
- Las instancias de Aurora Serverless v2 se someten a operaciones de mantenimiento con la misma frecuencia que las instancias aprovisionadas. Como Aurora reanuda automáticamente las instancias cuando se necesita dicho mantenimiento, es posible que las instancias de Aurora Serverless v2 se reanuden con más frecuencia que los clústeres de Aurora Serverless v1.

Funcionamiento de la pausa automática de Aurora Serverless v2 para diferentes tipos de clústeres de Aurora

Las consideraciones sobre la característica de pausa automática dependen del número de instancias que haya en el clúster de Aurora, de los niveles de promoción de conmutación por error de las instancias de lector y de si todas las instancias son de Aurora Serverless v2 o es una combinación de instancias de Aurora Serverless v2 y aprovisionadas.

Temas

- Diseños de los clústeres de Aurora recomendados al utilizar la pausa automática
- Funcionamiento de la pausa automática de Aurora Serverless v2 para la instancia de escritor en un clúster de base de datos
- Funcionamiento de la pausa automática de Aurora Serverless v2 en los clústeres Multi-AZ
- Funcionamiento de la pausa automática de Aurora Serverless v2 en clústeres con instancias aprovisionadas

Diseños de los clústeres de Aurora recomendados al utilizar la pausa automática

Si la característica de pausa automática está habilitada, puede organizar su clúster de Aurora para lograr el equilibrio adecuado entre alta disponibilidad, respuesta rápida y escalabilidad para que se adapte a su caso de uso. Para ello, debe elegir la combinación de instancias de Aurora Serverless v2, instancias aprovisionadas y niveles de promoción de conmutación por error para las instancias de base de datos de su clúster.

Los siguientes tipos de configuraciones muestran diferentes ventajas y desventajas entre la alta disponibilidad y la optimización de costos para su clúster:

- Para un sistema de desarrollo y prueba, puede configurar un clúster de base de datos Single-AZ con una instancia de base de datos de Aurora Serverless v2. La instancia única atiende todas las solicitudes de lectura y escritura. Cuando el clúster no se utiliza durante intervalos de tiempo significativos, la instancia de base de datos se pausa. En ese momento, los costos de computación de la base de datos del clúster también se pausan.
- En el caso de un sistema que ejecute una aplicación en el que la alta disponibilidad sea una prioridad, pero el clúster aún tenga períodos en los que esté completamente inactivo, puede configurar un clúster Multi-AZ en el que las instancias de base de datos de escritor y de lector sean Aurora Serverless v2. Establezca una prioridad de conmutación por error de cero o uno para la instancia de lector, de modo que la instancia de escritor y de lector se pausen y se reanuden al mismo tiempo. Ahora podrá disfrutar de una conmutación por error rápida mientras el clúster esté activo. Cuando el clúster permanece inactivo durante más tiempo que el umbral de pausa automática, se pausan los cargos de la instancia de base de datos de ambas instancias. Cuando el clúster reanude el procesamiento, la primera sesión de base de datos tardará unos breves instantes en conectarse.
- Suponga que su clúster está constantemente activo con una cantidad mínima de actividad y que requiere una respuesta rápida para cualquier conexión. En ese caso, puede crear un clúster con más de una instancia de lector de Aurora Serverless v2 y separar las capacidades de algunas instancias de lector de las de escritor. Especifique la prioridad de conmutación por error cero o uno para la instancia de escritor y una instancia de lector. Especifique una prioridad mayor que uno para las demás instancias de lector. De esta forma, las instancias de lector en los niveles de mayor prioridad pueden pausarse automáticamente, incluso mientras el escritor y uno de los lectores permanecen activos.

  En este caso, puede emplear otras técnicas para garantizar que el clúster permanezca disponible de forma continua y, al mismo tiempo, se reduzca verticalmente hasta alcanzar una capacidad baja durante los períodos de inactividad:

  - Puede usar instancias aprovisionadas para el escritor y el lector con prioridad 0 o 1. De este modo, dos instancias de base de datos nunca se pausan automáticamente y siempre están disponibles para atender el tráfico de la base de datos y realizar conmutaciones por error.
  - Puede configurar un punto de conexión personalizado que incluya las instancias de Aurora Serverless v2 de los niveles de mayor prioridad, pero no el escritor ni los lectores de nivel de promoción 0 o 1. De esta forma, puede dirigir las sesiones de solo lectura que no sean sensibles

a la latencia hacia los lectores que podrían pausarse automáticamente. Puede evitar usar el punto de conexión de lector para este tipo de solicitudes, ya que Aurora podría dirigir las conexiones del punto de conexión de lector a la instancia de lector siempre activa o a una de las instancias pausadas automáticamente. El uso del punto de conexión personalizado le permite dirigir las conexiones hacia diferentes grupos de instancias en función de su preferencia para una respuesta rápida o una capacidad de escalado adicional.

Funcionamiento de la pausa automática de Aurora Serverless v2 para la instancia de escritor en un clúster de base de datos

Cuando un clúster de base de datos de Aurora contiene solo una instancia de base de datos, el mecanismo para pausar y reanudar automáticamente la instancia de base de datos es sencillo. Depende únicamente de la actividad de la instancia de escritor. Es posible que tenga una configuración de este tipo para los clústeres que se utilizan para el desarrollo y las pruebas, o para ejecutar aplicaciones en las que la alta disponibilidad no es crucial. Tenga en cuenta que, en un clúster de instancia única, Aurora dirige las conexiones a través del punto de conexión de lector a la instancia de base de datos de escritor. Por lo tanto, en el caso de un clúster de base de datos de instancia única, al intentar conectarse al punto de conexión de lector, se reanudará la instancia de base de datos de escritor pausada automáticamente.

Los siguientes factores adicionales se aplican a los clústeres de Aurora con varias instancias de base de datos:

- En un clúster de base de datos de Aurora, normalmente se accede con frecuencia a la instancia de base de datos de escritor. Por lo tanto, es posible que la instancia de base de datos de escritor permanezca activa incluso cuando una o más instancias de base de datos de lector se pausen automáticamente.
- Algunas actividades de las instancias de base de datos de lector requieren que la instancia de base de datos de escritor esté disponible. Por lo tanto, las instancias de base de datos de escritor no se pueden pausar hasta que todas las instancias de lector también estén pausadas. Al reanudar cualquier instancia de lector, se reanuda automáticamente la instancia de escritor, incluso si la aplicación no accede directamente a la instancia de escritor.
- Las instancias de escritor de Aurora Serverless v2 en los niveles de promoción de conmutación por error cero y uno se escalan para mantener su capacidad sincronizada con la instancia de escritor. Por lo tanto, cuando se reanuda una instancia de escritor de Aurora Serverless v2, también lo hacen las instancias de lector de Aurora Serverless v2 que estén en los niveles de promoción cero o uno.

Funcionamiento de la pausa automática de Aurora Serverless v2 en los clústeres Multi-AZ

Dentro de un clúster de base de datos de Aurora que contiene una instancia de base de datos de escritor y una o más instancias de base de datos de lector, es posible que algunas instancias de base de datos de Aurora Serverless v2 estén pausadas mientras otras están activas. La instancia de escritor y cualquier instancia de lector con prioridad de conmutación por error de 0 y 1 siempre se pausan y se reanudan al mismo tiempo. Las instancias de lector con una prioridad distinta de 0 o 1 pueden pausarse y reanudarse independientemente de las demás instancias.

Al utilizar esta característica para clústeres con varias instancias de lector, puede administrar los costos sin sacrificar la alta disponibilidad. La instancia de escritor y una o dos instancias de lector pueden permanecer activas en todo momento, mientras que las instancias de lector adicionales pueden pausarse cuando no son necesarias para administrar un gran volumen de tráfico de lectura.

El hecho de que una instancia de lector pueda pausarse automáticamente depende de si su capacidad se puede escalar de forma independiente o de si la capacidad está vinculada a la de la instancia de base de datos de escritor. Esa propiedad de escalado depende de la prioridad de conmutación por error de la instancia de base de datos de lector. Cuando la prioridad de lector es cero o uno, la capacidad del lector realiza un seguimiento de la capacidad de la instancia de base de datos del escritor. Por lo tanto, para permitir que las instancias de base de datos de lector se pausen automáticamente en la variedad más amplia de situaciones, debe definir la prioridad en un valor mayor que uno.

El tiempo necesario para reanudar una instancia de lector puede ser ligeramente superior al de reanudar una instancia de escritor. Para obtener la respuesta más rápida en caso de que las instancias estén pausadas, conéctese al punto de conexión del clúster.

Funcionamiento de la pausa automática de Aurora Serverless v2 en clústeres con instancias aprovisionadas

Las instancias de base de datos aprovisionadas en su clúster de base de datos de Aurora no se pausan automáticamente. Solo las instancias de base de datos de Aurora Serverless v2 con la clase de instancia `db.serverless` pueden usar la característica de pausa automática.

Cuando el clúster de Aurora contiene instancias de base de datos aprovisionadas, cualquier instancia de escritor de Aurora Serverless v2 se pausa automáticamente. Esto se debe al requisito de que la instancia de escritor debe permanecer disponible mientras las instancias de lector estén

activas. El hecho de que el escritor de Aurora Serverless v2 también permanezca activo significa que las instancias de lector de Aurora Serverless v2 con prioridad de conmutación por error 0 y 1 no se pausarán automáticamente en un clúster híbrido que contenga instancias aprovisionadas.

Supervisión de los clústeres de Aurora que utilizan la pausa automática

Para supervisar Aurora, ya debe estar familiarizado con los procedimientos de supervisión en Supervisión de métricas de Amazon Aurora con Amazon CloudWatch y las métricas de CloudWatch que se muestran en Referencia de métricas para Amazon Aurora. Tenga en cuenta que hay que tener en cuenta algunas consideraciones especiales al supervisar los clústeres de Aurora que utilizan la característica de pausa automática:

- Puede haber períodos en los que las instancias de Aurora Serverless v2 no registren los datos de registro y la mayoría de las métricas debido a que las instancias están pausadas. Las únicas métricas que se envían a CloudWatch mientras una instancia está pausada son el cero por ciento para `CPUUtilization` y `ACUUtilization`, y el cero para `ServerlessDatabaseCapacity`.
- Puede comprobar si las instancias de Aurora Serverless v2 se pausan con más o menos frecuencia de lo esperado. Para ello, compruebe la frecuencia con la que la métrica `ServerlessDatabaseCapacity` cambia de un valor distinto de cero a cero y durante cuánto tiempo permanece en cero. Si las instancias no permanecen pausadas el tiempo esperado, significa que no está ahorrando todo lo que podría. Si las instancias se pausan y reanudan con más frecuencia de la prevista, es posible que el clúster tenga una latencia innecesaria al responder a las solicitudes de conexión. Para obtener información sobre los factores que afectan a la frecuencia con que se pueden pausar las instancias de Aurora Serverless v2 y de qué manera, consulte Requisitos previos y limitaciones de la característica de pausa automática de Aurora Serverless v2, Situaciones en las Aurora Serverless v2 que no se pausa automáticamente y Solución de problemas para la característica de pausa automática de Aurora Serverless v2.
- También puede examinar un archivo de registro que registre las operaciones de pausa y reanudación automáticas de una instancia de Aurora Serverless v2. Si una instancia no se ha detenido después de que expirara el intervalo de tiempo de espera, este archivo de registro también incluye el motivo por el que no se ha realizado la pausa automática. Para obtener más información, consulte ???.

Temas

- Comprobación de si una instancia de Aurora Serverless v2 está pausada
- Eventos para las operaciones de pausa automática y reanudación automática

- Funcionamiento de la pausa automática con Información de rendimiento y Monitorización mejorada
- Cómo interactúa la característica de pausa automática de Aurora Serverless v2 con las métricas de Aurora

Comprobación de si una instancia de Aurora Serverless v2 está pausada

Para determinar si una instancia de Aurora Serverless v2 está pausada, puede observar la métrica `ACUUtilization` de la instancia. Esta métrica tiene un valor igual a cero mientras la instancia está pausada.

Mientras una instancia de Aurora Serverless v2 está en pausa, su valor de estado sigue apareciendo como Disponible. Lo mismo se aplica cuando una instancia de Aurora Serverless v2 pausada está en proceso de reanudación. Esto se debe a que puede conectarse correctamente a una instancia de este tipo, incluso si la conexión experimenta un ligero retraso.

Cualquier métrica relacionada con la disponibilidad de las instancias de Aurora considera el período durante el cual la instancia está en pausa como el tiempo en que la instancia ha estado disponible.

Eventos para las operaciones de pausa automática y reanudación automática

Aurora emite eventos para las instancias de Aurora Serverless v2 cuando las operaciones de pausa y reanudación automáticas se inician, finalizan o se cancelan. Los eventos relacionados con la característica de pausa automática son de `RDS-EVENT-0370` a `RDS-EVENT-0374`. Para obtener más detalles acerca de estos eventos, consulte Eventos de instancia de base de datos.

Funcionamiento de la pausa automática con Información de rendimiento y Monitorización mejorada

Mientras una instancia de Aurora Serverless v2 está en pausa, Aurora no recopila información de supervisión para esa instancia a través de Información de rendimiento o Monitorización mejorada. Cuando se reanude la instancia, es posible que se produzca un breve retraso antes de que Aurora reanude la recopilación de dicha información de supervisión.

Cómo interactúa la característica de pausa automática de Aurora Serverless v2 con las métricas de Aurora

Mientras una instancia de Aurora Serverless v2 está pausada, no emite la mayoría de las métricas de CloudWatch ni escribe información en los registros de su base de datos. Las únicas métricas

que se envían a CloudWatch mientras una instancia está pausada son el cero por ciento para `CPUUtilization` y `ACUUtilization`, y el cero para `ServerlessDatabaseCapacity`.

Cuando CloudWatch calcula las estadísticas relacionadas con la disponibilidad y el tiempo de actividad de las instancias o los clústeres, se considera que las instancias de Aurora Serverless v2 están disponibles durante el tiempo en que están pausadas.

Si inicia una acción de la AWS CLI o de la API de RDS para describir o descargar los registros de una instancia de Aurora Serverless v2 pausada, la instancia se reanuda automáticamente para que la información del registro esté disponible.

Ejemplo de métricas de CloudWatch

Los siguientes ejemplos de AWS CLI muestran cómo se puede observar de qué manera cambia la capacidad de una instancia con el paso del tiempo. Durante ese período, la instancia se pausa automáticamente y, a continuación, se reanuda. Mientras está pausada, la métrica `ServerlessDatabaseCapacity` muestra un valor de cero. Para determinar si la instancia se ha pausado en algún momento durante dicho período de tiempo, verificamos si la capacidad mínima durante ese período de tiempo ha sido igual a cero.

El siguiente ejemplo de Linux representa una instancia de Aurora Serverless v2 que ha estado pausada automáticamente durante algún tiempo. Tomamos muestras de `ServerlessDatabaseCapacity` cada minuto, durante un período de tres minutos. El valor mínimo de la ACU de 0,0 confirma que la instancia se ha pausado en algún momento en cada minuto.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '3 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:11:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:12:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
```

A continuación, intentamos establecer una conexión con la instancia de Aurora Serverless v2 pausada. En este ejemplo, utilizamos una contraseña incorrecta expresamente para que el intento

de conexión no tenga éxito. A pesar del error, el intento de conexión hace que Aurora reanude la instancia pausada.

```
$ mysql -h my_cluster_endpoint.rds.amazonaws.com -u admin -pwrong-password

ERROR 1045 (28000): Access denied for user 'admin'@'ip_address' (using password: YES)
```

El siguiente ejemplo de Linux demuestra que la instancia pausada se ha reanudado, ha permanecido inactiva durante aproximadamente cinco minutos y, después, ha vuelto a su estado de pausa. La instancia se ha reanudado con una capacidad de 2,0 ACU. Luego se ha escalado verticalmente, por ejemplo, para realizar una limpieza interna. Puesto que no ha recibido ningún intento de conexión por parte del usuario en el periodo de espera de cinco minutos, ha pasado directamente de 4,0 ACU al estado de pausa.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '8 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
DATAPOINTS 2.0 2024-08-02T22:14:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:15:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:16:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:17:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:18:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:19:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:20:00+00:00 None
```

Si el informe pretendía mostrar la rapidez con la que la instancia se ha escalado verticalmente para gestionar la carga de trabajo, podríamos especificar `Maximum` en lugar de `Minimum` para la estadística. Para planificar la capacidad y estimar los costos, podríamos especificar un período de tiempo más largo y usar la estadística `Average`. De esta forma, podríamos determinar los valores de capacidad típicos durante los períodos de alta actividad, baja actividad y estado de pausa. Para examinar el comportamiento durante los momentos precisos de pausa y reanudación, podríamos

especificar un período de un segundo y examinar un intervalo de tiempo más corto. Los valores de marca temporal de la salida, como `2024-08-02T22:13:00+00:00`, muestran el formato para especificar parámetros precisos para las opciones `--start-time` y `--end-time`.

Solución de problemas para la característica de pausa automática de Aurora Serverless v2

Si detecta que las instancias de Aurora Serverless v2 no se detienen con la frecuencia esperada, compruebe las siguientes causas posibles:

- Confirme que la versión de Aurora que está ejecutando admite una capacidad mínima de cero ACU. Para conocer los rangos de capacidad de las diferentes versiones de Aurora, consulte Capacidad de Aurora Serverless v2.
- Confirme que el valor de capacidad mínima del clúster esté establecido en cero ACU.
- Confirme que la instancia en cuestión utiliza realmente la clase de instancia `db.serverless` de Aurora Serverless v2, no una de las clases de instancias aprovisionadas.
- Confirme que el clúster no utilice ninguna de las características o configuraciones incompatibles de Requisitos previos y limitaciones de la característica de pausa automática de Aurora Serverless v2.
- Examine el archivo de registro que muestra cuándo se pausaron o se reanudaron las instancias de Aurora Serverless v2 o si Aurora no pudo pausar ni reanudar una instancia por algún motivo. Para obtener más información, consulte ???.
- Compruebe si algún cliente o aplicación mantiene las conexiones abiertas durante largos periodos de tiempo. Por el contrario, compruebe si alguna aplicación que utilice la API de datos de RDS o las funciones de Lambda envía solicitudes frecuentes para que la instancia nunca esté inactiva el tiempo suficiente como para pausarla. Puede examinar las métricas de CloudWatch, como `ConnectionAttempts` y `DatabaseConnections`. Para obtener más información, consulte Métricas de nivel de instancia para Amazon Aurora.
- Si una instancia de lector se pausa muy pocas veces, o nunca, compruebe su prioridad de conmutación por error. Si el lector se utiliza para escalar el lector y no como dispositivo de reserva en caso de conmutación por error, configúrelo en una prioridad entre 2 y 15.
- Si la instancia de escritor se detiene muy pocas veces, o nunca, compruebe el uso que hace de las instancias de lector. El escritor solo puede pausar si todo el clúster está inactivo. Esto se debe a que una conexión a cualquier instancia de lector provoca que el escritor se reanude.
- Si su aplicación recibe tiempos de espera durante las solicitudes de conexión mientras se reanudan las instancias de Aurora Serverless v2, considere la posibilidad de alargar el intervalo de

tiempo de espera utilizado por el código de la aplicación o el marco de base de datos subyacente. Los tiempos de espera de conexión más prolongados reducen la posibilidad de que se produzcan errores en las conexiones mientras se reanudan las instancias de Aurora Serverless v2. Sin embargo, los tiempos de espera más largos también pueden hacer que la aplicación detecte más lentamente los problemas de disponibilidad en el clúster.

Por el contrario, considere la posibilidad de alargar el intervalo de pausa automática para que las aplicaciones no encuentren instancias pausadas con tanta frecuencia.

Si no existe un equilibrio lógico entre el comportamiento de pausa automática y la capacidad de respuesta del clúster para su aplicación, es posible que ese clúster no sea un buen candidato para usar la característica de pausa automática.

- Si calcula cuánto tiempo permanecerán en pausa las instancias de Aurora Serverless v2, tenga en cuenta que hay factores que hacen que no sea práctico realizar predicciones precisas.
  - Es posible que las instancias se reanuden periódicamente para realizar tareas de mantenimiento, actualizaciones a versiones secundarias o aplicar cambios a los grupos de parámetros.
  - En el caso de los clústeres Multi-AZ, hay situaciones en las que la reanudación de una instancia provoca que también se reanuden otras instancias. Reanudar un lector siempre provoca que también se reanude el escritor. Al reanudar el escritor, siempre se reanudará la sesión de los lectores con prioridad de conmutación por error 0 y 1.

  Recomendamos medir el tiempo de pausa real durante un período de varios días, con una carga de trabajo realista. A continuación, use esas medidas para establecer una referencia para la proporción de tiempo que cabe esperar que esté pausada una instancia.
- Es posible que las operaciones internas, como la purga de MySQL, el programador de eventos de MySQL, el autovacuum de PostgreSQL o los trabajos de PostgreSQL programados a través de la extensión `pg_cron` no se estén ejecutando o no se estén completando. La instancia no se reanuda automáticamente para realizar operaciones que no impliquen una conexión de usuario a la base de datos. Si dichas operaciones internas están en marcha cuando se agote el tiempo de espera de la pausa automática, se cancelarán las tareas de mantenimiento, como la purga de MySQL y el autovacuum de PostgreSQL. Los trabajos programados desde el programador de eventos de MySQL o la extensión `pg_cron` de PostgreSQL también se cancelan si están en curso cuando Aurora inicia la operación de pausa.

  Si necesita asegurarse de que la instancia esté activa periódicamente para realizar las operaciones programadas, puede iniciar una conexión para reanudar la instancia antes de la

hora de inicio del trabajo. También puede aumentar el intervalo de tiempo de espera de la pausa automática para que operaciones como el autovacuum puedan ejecutarse durante más tiempo una vez finalizada la actividad del usuario. También puede usar mecanismos como las funciones de Lambda para realizar las operaciones de la base de datos según una programación, de forma que se reanude automáticamente la instancia si es necesario.

Consideraciones sobre el diseño de la aplicación para la característica de pausa automática de Aurora Serverless v2

Cuando una instancia de base de datos de Aurora Serverless v2 se reanuda tras haber sido pausada automáticamente, comienza con una capacidad relativamente pequeña y, a partir de ahí, se escala verticalmente. Esta capacidad inicial se aplica incluso si la instancia de base de datos tenía una capacidad superior inmediatamente antes de que se detuviera automáticamente.

Utilice esta característica con aplicaciones que puedan tolerar un intervalo de aproximadamente 15 segundos al establecer una conexión. Esto explica el caso típico en el que una instancia de Aurora Serverless v2 se reanuda debido a una o a un número reducido de conexiones entrantes. Si la instancia está pausada durante más de 24 horas, es posible que el tiempo de reanudación sea mayor.

Si la aplicación ya utilizaba Aurora Serverless v1 y la característica de pausa automática, es posible que ya disponga de esos intervalos de tiempo de espera para los intentos de conexión. Si ya utiliza Aurora Serverless v2 en combinación con la característica de parada e inicio en un clúster de Aurora, el tiempo de reanudación de las instancias de Aurora Serverless v2 pausadas automáticamente suele ser mucho más corto que el tiempo de inicio de un clúster detenido.

Al codificar la lógica de conexión en su aplicación, vuelva a intentar la conexión si el primer intento arroja un error de causa transitoria. (Si el error se debe a un error de autenticación, corrija las credenciales antes de volver a intentarlo). Un error que se produce inmediatamente después de la reanudación puede ser un tiempo de espera o algún error relacionado con los límites de la base de datos. Reintentarlo puede solucionar problemas en los casos más raros en los que una instancia de Aurora Serverless v2 se reanuda debido a un gran número de solicitudes de conexión simultáneas. En ese caso, es posible que algunas conexiones tarden más de lo habitual en procesarse o que superen el límite de conexiones simultáneas en el primer intento.

Durante el desarrollo y la depuración de aplicaciones, no deje abiertas a la base de datos las sesiones del cliente o las herramientas de programación con conexiones. Aurora no pausará una

instancia si hay alguna conexión iniciada por el usuario abierta, independientemente de si las conexiones no ejecutan ninguna instrucción o transacción de SQL. Cuando una instancia de Aurora Serverless v2 de un clúster de Aurora no se puede pausar, es posible que también se impida que se detengan otras instancias del clúster. Para obtener más información, consulte Situaciones en las Aurora Serverless v2 que no se pausa automáticamente.

Aurora emite eventos cuando una instancia de base de datos de Aurora Serverless v2 comienza a reanudarse, termina de reanudarse y si la instancia no puede reanudarse por algún motivo. Para obtener más detalles acerca de estos eventos, consulte Eventos de instancia de base de datos.

Migración a Aurora Serverless v2

Para convertir un clúster de base de datos existente para utilizar Aurora Serverless v2, puede hacer lo siguiente:

- Actualizar desde un clúster de base de datos de Aurora aprovisionado.
- Actualizar desde un clúster de Aurora Serverless v1.
- Migrar una base de datos en las instalaciones a un clúster de Aurora Serverless v2.

Cuando el clúster actualizado ejecuta la versión del motor adecuada, tal como se indica en Requisitos y limitaciones para Aurora Serverless v2, puede empezar a añadirle instancias de base de datos de Aurora Serverless v2. La primera instancia de base de datos que se agrega al clúster actualizado debe ser una instancia de base de datos aprovisionada. A continuación, puede cambiar el procesamiento de la carga de trabajo de escritura, la carga de trabajo de lectura o ambos a las instancias de base de datos de Aurora Serverless v2.

Contenido

- Actualización o cambio de clústeres existentes para utilizar Aurora Serverless v2
  - Rutas de actualización para clústeres compatibles con MySQL para usar Aurora Serverless v2
  - Rutas de actualización para clústeres compatibles con PostgreSQL para usar Aurora Serverless v2
- Cambiar de un clúster aprovisionado a Aurora Serverless v2
- Comparación de Aurora Serverless v2 y Aurora Serverless v1
  - Comparación de los requisitos de Aurora Serverless v2 y Aurora Serverless v1
  - Comparación del escalado y la disponibilidad de Aurora Serverless v2 y Aurora Serverless v1

  - Comparación de la compatibilidad de características de Aurora Serverless v2 y Aurora Serverless v1
  - Adaptación casos de uso de Aurora Serverless v1 para Aurora Serverless v2
- Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2
  - Clústeres de base de datos compatibles con Aurora MySQL
  - Clústeres de base de datos compatibles con Aurora PostgreSQL
- Migración de una base de datos en las instalaciones a Aurora Serverless v2

> **Note**
>
> En estos temas se describe cómo convertir un clúster de base de datos existente. Para obtener más información acerca de la creación de un clúster de bases de datos de Aurora Serverless v2 nuevo, consulte Creación de un clúster de base de datos que utiliza Aurora Serverless v2.

Actualización o cambio de clústeres existentes para utilizar Aurora Serverless v2

Si el clúster aprovisionado tiene una versión de motor que admite Aurora Serverless v2, cambiar a Aurora Serverless v2 no requiere actualización. En ese caso, puede añadir instancias de base de datos de Aurora Serverless v2 al clúster original. Puede cambiar el clúster para utilizar todas las instancias de base de datos de Aurora Serverless v2. También puede utilizar una combinación de instancias de base de datos de Aurora Serverless v2 aprovisionadas en el mismo clúster de base de datos. Para las versiones del motor Aurora compatibles con Aurora Serverless v2, consulte Regiones y motores de base de datos Aurora admitidos para Aurora Serverless v2.

Si está ejecutando una versión del motor anterior que no admite Aurora Serverless v2, siga estos pasos generales:

1. Actualice el clúster.
2. Cree una instancia de base de datos de escritor aprovisionada para el clúster actualizado.
3. Modifique el clúster para usar instancias de base de datos de Aurora Serverless v2.

> **Important**
>
> Cuando realiza una actualización de la versión principal a un versión compatible con Aurora Serverless v2 mediante la restauración o la clonación de instantáneas, la primera instancia de base de datos que agregue al nuevo clúster debe ser una instancia de base de datos aprovisionada. Esta adición inicia la etapa final del proceso de actualización.
> Hasta que se llega a esa etapa final, el clúster no tiene la infraestructura necesaria para admitir Aurora Serverless v2. Por lo tanto, estos clústeres actualizados siempre comienzan con una instancia de base de datos de escritura aprovisionada. A continuación, puede conmutar por error o convertir la instancia de base de datos aprovisionada en una instancia de Aurora Serverless v2.

La actualización de Aurora Serverless v1 en Aurora Serverless v2 implica crear un clúster aprovisionado como paso intermedio. A continuación, debe realizar los mismos pasos de actualización que cuando comienza con un clúster aprovisionado.

Rutas de actualización para clústeres compatibles con MySQL para usar Aurora Serverless v2

Si el clúster original ejecuta Aurora MySQL, elija el procedimiento adecuado según la versión del motor y el modo de motor de su clúster.

Si su clúster original de Aurora MySQL es este	Haga esto para pasar a Aurora Serverless v2
Clúster aprovisionado que ejecuta Aurora MySQL versión 3 compatible con MySQL 8.0	Esta es la etapa final de todas las conversiones de los clústeres de Aurora MySQL existentes. Si es necesario, realice una actualización de la versión menor a la versión 3.02.0 o superior. Utilice una instancia de base de datos aprovisionada para la instancia de base de datos de escritor. Añada una instancia de base de datos de lector de Aurora Serverless v2. Realice una conmutación por error para convertirla en la instancia de base de datos de escritor.

Si su clúster original de Aurora MySQL es este	Haga esto para pasar a Aurora Serverless v2
	(Opcional) Convierta otras instancias de base de datos aprovisionadas en el clúster en Aurora Serverless v2. O añada nuevas instancias de base de datos de Aurora Serverless v2 y elimine las instancias de base de datos aprovisionadas. Para ver el procedimiento completo y ejemplos, consulte Cambiar de un clúster aprovisionado a Aurora Serverless v2.
Clúster aprovisionado que ejecuta Aurora MySQL versión 2 compatible con MySQL 5.7	Realice una actualización de la versión principal a Aurora MySQL versión 3.02.0 o posterior. A continuación, siga el procedimiento para Aurora MySQL versión 3 para cambiar el clúster para que utilice instancias de base de datos de Aurora Serverless v2.
Aurora Serverless v1Clúster de que ejecuta Aurora MySQL versión 2, compatible con MySQL 5.7	Para ayudar a planificar la conversión de Aurora Serverless v1, consulte Comparación de Aurora Serverless v2 y Aurora Serverless v1 primero. A continuación, siga el procedimiento indicado en Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.

Rutas de actualización para clústeres compatibles con PostgreSQL para usar Aurora Serverless v2

Si el clúster original ejecuta Aurora PostgreSQL, elija el procedimiento adecuado según la versión del motor y el modo de motor del clúster.

Si el clúster original de Aurora PostgreSQL es este	Haga esto para cambiar a Aurora Serverless v2
Clúster aprovisionado que ejecuta Aurora PostgreSQL versión 13	Esta es la etapa final de todas las conversiones de los clústeres existentes de Aurora PostgreSQL. Si es necesario, realice una actualización de la versión secundaria a la versión 13.6 o superior. Agregue una instancia de base de datos aprovisionada para la instancia de base de datos de escritor. Añada una instancia de base de datos de lector de Aurora Serverless v2. Lleve a cabo una conmutación por error para hacer que esa instancia de Aurora Serverless v2 sea la instancia de base de datos de escritor. (Opcional) Convierta otras instancias de base de datos aprovisionadas en el clúster en Aurora Serverless v2. O añada nuevas instancias de base de datos de Aurora Serverless v2 y elimine las instancias de base de datos aprovisionadas. Para ver el procedimiento completo y ejemplos, consulte Cambiar de un clúster aprovisionado a Aurora Serverless v2.
Clúster aprovisionado que ejecuta la versión 11 o 12 de Aurora PostgreSQL	Realice una actualización de la versión principal a Aurora PostgreSQL versión 13.6 o posterior. A continuación, siga el procedimiento para Aurora PostgreSQL versión 13 para cambiar el clúster para que use instancias de base de datos de Aurora Serverless v2.
Clúster de Aurora Serverless v1 que ejecuta Aurora PostgreSQL versión 11 o 13	Para ayudar a planificar la conversión de Aurora Serverless v1, consulte Comparación

Si el clúster original de Aurora PostgreSQL es este	Haga esto para cambiar a Aurora Serverless v2
	de Aurora Serverless v2 y Aurora Serverless v1 primero. A continuación, siga el procedimiento indicado en Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.

Cambiar de un clúster aprovisionado a Aurora Serverless v2

Para cambiar un clúster aprovisionado para que utilice Aurora Serverless v2, siga estos pasos:

1. Compruebe si es necesario actualizar el clúster aprovisionado para utilizarlo con instancias de base de datos de Aurora Serverless v2. Para las versiones de Aurora compatibles con Aurora Serverless v2, consulte Requisitos y limitaciones para Aurora Serverless v2.

   Si el clúster aprovisionado ejecuta una versión de motor que no está disponible para Aurora Serverless v2, actualice la versión del motor del clúster:

   - Si tiene un clúster aprovisionado compatible con MySQL 5.7, siga las instrucciones de actualización para Aurora MySQL versión 3. Utilice el procedimiento de Pasos para realizar una actualización local.
   - Si tiene un clúster aprovisionado compatible con PostgreSQL que ejecuta PostgreSQL versiones 11 o 12, siga las instrucciones de actualización de Aurora PostgreSQL versión 13. Utilice el procedimiento de Actualización a una versión principal.
2. Configure cualquier otra propiedad del clúster para que coincida con los requisitos de Aurora Serverless v2 de Requisitos y limitaciones para Aurora Serverless v2.
3. Defina la configuración de escalado del clúster. Siga el procedimiento indicado en Configuración del rango de capacidad de Aurora Serverless v2 para un clúster.
4. Añada una o varias instancias de bases de datos de Aurora Serverless v2 al clúster. Siga el procedimiento general de Adición de réplicas de Aurora a un clúster de base de datos. Para cada nueva instancia de base de datos, especifique el nombre de clase de instancia de base de datos Sin servidor en la Consola de administración de AWS, o bien `db.serverless` en la AWS CLI o la API de Amazon RDS.

En algunos casos, es posible que ya tenga una o más instancias de base de datos de lector aprovisionadas en el clúster. De ser así, puede convertir uno de los lectores en una instancia de base de datos de Aurora Serverless v2, en lugar de crear una nueva instancia de base de datos. Para ello, siga el procedimiento en Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2.

5. Realice una operación de conmutación por error para hacer que una de las instancias de base de datos de Aurora Serverless v2 sea la instancia de base de datos de escritor del clúster.
6. (Opcional) Convierta las instancias de base de datos aprovisionadas en Aurora Serverless v2 o elimínelas del clúster. Siga el procedimiento general de Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2 o Eliminación de una instancia de base de datos de un clúster de base de datos de Aurora.

> **Tip**
>
> Eliminar las instancias de base de datos aprovisionadas no es obligatorio. Puede configurar un clúster que contenga tanto Aurora Serverless v2 como instancias de base de datos aprovisionadas. Sin embargo, hasta que conozca bien las características de rendimiento y escalado de las instancias de base de datos de Aurora Serverless v2, le recomendamos que configure los clústeres con instancias de base de datos del mismo tipo.

Los siguientes ejemplos de la AWS CLI muestran el proceso de conmutación utilizando un clúster aprovisionado que ejecuta Aurora MySQL versión 3.02.0. El clúster se denomina `mysql-80`. El clúster comienza con dos instancias de base de datos aprovisionadas llamadas `provisioned-instance-1` y `provisioned-instance-2`, escritor y lector. Ambos usan la clase instancia de base de datos de `db.r6g.large`.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' --output text
mysql-80
provisioned-instance-2     False
provisioned-instance-1     True

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-1 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
```

```
provisioned-instance-1     db.r6g.large

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-2 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-2     db.r6g.large
```

Creamos una tabla con algunos datos. De esta forma podemos confirmar que los datos y el funcionamiento del clúster son los mismos antes y después de la conmutación.

```
mysql> create database serverless_v2_demo;
mysql> create table serverless_v2_demo.demo (s varchar(128));
mysql> insert into serverless_v2_demo.demo values ('This cluster started with a
 provisioned writer.');
Query OK, 1 row affected (0.02 sec)
```

En primer lugar, añadimos un rango de capacidad al clúster. De no hacerlo, se producirá un error al añadir cualquier instancia de base de datos de Aurora Serverless v2 al clúster. Si usamos la Consola de administración de AWS para este procedimiento, ese paso es automático cuando añadimos la primera instancia de base de datos de Aurora Serverless v2.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql

An error occurred (InvalidDBClusterStateFault) when calling the CreateDBInstance
 operation:
Set the Serverless v2 scaling configuration on the parent DB cluster before creating a
 Serverless v2 DB instance.

$ # The blank ServerlessV2ScalingConfiguration attribute confirms that the cluster
 doesn't have a capacity range set yet.
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 --query
 'DBClusters[*].ServerlessV2ScalingConfiguration'
[]

$ aws rds modify-db-cluster --db-cluster-identifier mysql-80 \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
{
  "DBClusterIdentifier": "mysql-80",
  "ServerlessV2ScalingConfiguration": {
    "MinCapacity": 0.5,
    "MaxCapacity": 16
```

```
  }
}
```

Creamos dos lectores de Aurora Serverless v2 que sustituyen a las instancias de base de datos originales. Para ello, especificamos la clase de instancia de base de datos `db.serverless` para las nuevas instancias de base de datos.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 --db-
cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-2 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-2",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ # Wait for both DB instances to finish being created before proceeding.
$ aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
 && \
  aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-2
```

Realizamos una conmutación por error para que una de las instancias de base de datos de Aurora Serverless v2 sea el nuevo escritor del clúster.

```
$ aws rds failover-db-cluster --db-cluster-identifier mysql-80 \
  --target-db-instance-identifier serverless-v2-instance-1
{
  "DBClusterIdentifier": "mysql-80",
  "DBClusterMembers": [
    {
      "DBInstanceIdentifier": "serverless-v2-instance-1",
      "IsClusterWriter": false,
```

```
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "serverless-v2-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-1",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "Status": "available"
}
```

Se necesitan unos segundos para que ese cambio surta efecto. En ese momento, tenemos un escritor de Aurora Serverless v2 y lector de Aurora Serverless v2. Por lo tanto, no necesitamos ninguna de las instancias de base de datos aprovisionadas originales.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
mysql-80
serverless-v2-instance-1        True
serverless-v2-instance-2        False
provisioned-instance-2          False
provisioned-instance-1          False
```

El último paso del procedimiento de conmutación es eliminar ambas instancias de base de datos aprovisionadas.

```
$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-2 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-2",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}

$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-1 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-1",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}
```

Como comprobación final, confirmamos que la tabla original es accesible y se puede escribir desde la instancia de base de datos de escritor de Aurora Serverless v2.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
+--------------------------------------------------+
1 row in set (0.00 sec)

mysql> insert into serverless_v2_demo.demo values ('And it finished with a Serverless
 v2 writer.');
Query OK, 1 row affected (0.01 sec)

mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
```

```
2 rows in set (0.01 sec)
```

También nos conectamos a la instancia de base de datos de lector de Aurora Serverless v2 y confirmamos que los datos recién escritos también están disponibles ahí.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
2 rows in set (0.01 sec)
```

Comparación de Aurora Serverless v2 y Aurora Serverless v1

Si ya está utilizando Aurora Serverless v1, puede conocer las principales diferencias entre Aurora Serverless v1 y Aurora Serverless v2. Las diferencias de arquitectura, como la compatibilidad con instancias de base de datos de lector, abren nuevos tipos de casos de uso.

Puede utilizar las siguientes tablas para comprender las diferencias más importantes que existen entre Aurora Serverless v2 y Aurora Serverless v1.

Temas

- Comparación de los requisitos de Aurora Serverless v2 y Aurora Serverless v1
- Comparación del escalado y la disponibilidad de Aurora Serverless v2 y Aurora Serverless v1
- Comparación de la compatibilidad de características de Aurora Serverless v2 y Aurora Serverless v1
- Adaptación casos de uso de Aurora Serverless v1 para Aurora Serverless v2

Comparación de los requisitos de Aurora Serverless v2 y Aurora Serverless v1

En la siguiente tabla se resumen los diferentes requisitos para ejecutar la base de datos utilizando Aurora Serverless v2 o Aurora Serverless v1. Aurora Serverless v2 ofrece versiones posteriores de los motores de base de datos Aurora MySQL y Aurora PostgreSQL que Aurora Serverless v1.

Característica	Aurora Serverless v2Requisito de	Aurora Serverless v1Requisito de
Motores de base de datos	Aurora MySQL, Aurora PostgreSQL	Aurora MySQL, Aurora PostgreSQL
Versiones de Aurora MySQL compatibles	Consulte Aurora Serverless v2 con Aurora MySQL.	Consulte Aurora Serverless v1 con Aurora MySQL.
Versiones compatibles de Aurora PostgreSQL	Consulte Aurora Serverless v2 con Aurora PostgreSQL.	Consulte Aurora Serverless v1 con Aurora PostgreSQL.
Actualización de un clúster de base de datos	De manera similar a los clústeres de base de datos aprovisionados, puede realizar las actualizaciones manualmente sin esperar a que Aurora actualice el clúster de base de datos automáticamente. Para obtener más información, consulte Modificación de un clúster de base de datos de Amazon Aurora. **Note** Para realizar una actualización de una versión principal de 13.x a 14.x o 15.x de un clúster de base de datos compatible con Aurora PostgreSQL, la capacidad máxima del	Las actualizaciones de versiones secundarias se aplican automáticamente a medida que están disponibles. Para obtener más información, consulte Aurora Serverless v1 y versiones del motor de base de datos de Aurora. Puede realizar las actualizaciones de la versión principal manualmente. Para obtener más información, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1.

Característica	Aurora Serverless v2Requisito de	Aurora Serverless v1Requisito de
	clúster debe ser de al menos 2 ACU.	

Característica	Aurora Serverless v2Requisito de	Aurora Serverless v1Requisito de
Conversión desde un clúster de base de datos aprovisionado	Puede usar uno de los métodos siguientes: • Añada una o varias instancias de base de datos de lector de Aurora Serverless v2 al clúster aprovisionado existente. Para utilizar Aurora Serverless v2 para el escritor, realice una conmutación por error en una de los instancias de base de datos de Aurora Serverless v2. Para que todo el clúster use instancias de base de datos de Aurora Serverless v2, elimine cualquier instancia de base de datos de escritor aprovisionada después de promover la instancia de base de datos de Aurora Serverless v2 al escritor. • Cree un nuevo clúster con el motor de base de datos y la versión del motor adecuados. Utilice cualquiera de los métodos estándar. Por ejemplo, restaure una instantánea de clúster o cree un clon de un clúster existente. Elija Aurora Serverless v2 para algunas	Restaure las instantáneas del clúster aprovisionado para crear un nuevo clúster de Aurora Serverless v1.

Característica	Aurora Serverless v2Requisito de	Aurora Serverless v1Requisito de
	o todas las instancias de base de datos del nuevo clúster. Si crea el nuevo clúster mediante clonación, no podrá actualizar la versión del motor al mismo tiempo. Asegúrese de que el clúster original ya esté ejecutando una versión del motor compatible con Aurora Serverless v2.	
Conversión desde un clúster de Aurora Serverless v1	Siga el procedimiento indicado en Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.	No aplicable
Clases de instancias de base de datos disponibles	La clase de instancia de base de datos especial `db.serverless` . En la Consola de administración de AWS, está etiquetada como Sin servidor.	No aplicable. Aurora Serverless v1 usa el modo del motor `serverless` .
Puerto	Cualquier puerto compatible con MySQL o PostgreSQL	Solo el puerto MySQL o PostgreSQL predeterminado
¿Se permite una dirección IP pública?	Sí	No

Característica	Aurora Serverless v2Requisito de	Aurora Serverless v1Requisito de
¿Se requiere una VPC (Nube virtual privada)?	Sí	Sí. Cada clúster de Aurora Serverless v1 consume dos puntos de conexión de interfaz y balanceador de carga de puerta de enlace asignados a la VPC.

Comparación del escalado y la disponibilidad de Aurora Serverless v2 y Aurora Serverless v1

En la siguiente tabla se indican las diferencias de escalabilidad y disponibilidad de Aurora Serverless v2 y Aurora Serverless v1.

El escalado de Aurora Serverless v2 es más receptivo, más granular y menos disruptivo que el escalado de Aurora Serverless v1. Aurora Serverless v2 se puede escalar cambiando el tamaño de la instancia de base de datos y añadiendo más instancias de base de datos al clúster de base de datos. También se puede escalar añadiendo clústeres de otras Regiones de AWS a una base de datos global de Aurora. En cambio, Aurora Serverless v1 solo se escala aumentando o disminuyendo la capacidad del escritor. Todo la computación de un clúster de Aurora Serverless v1 se ejecuta en una única zona de disponibilidad y en una única Región de AWS.

Característica de escalado y alta disponibilidad	Comportamiento de Aurora Serverless v2	Comportamiento de Aurora Serverless v1
Unidades de capacidad mínima de Aurora (ACU) (Aurora MySQL)	0,5 cuando el clúster se está ejecutando, 0 cuando el clúster está en pausa.	1 cuando el clúster se está ejecutando, 0 cuando el clúster está en pausa.
ACU mínimas (Aurora PostgreSQL)	0,5 cuando el clúster se está ejecutando, 0 cuando el clúster está en pausa.	2 cuando el clúster se está ejecutando, 0 cuando el clúster está en pausa.
ACU máximas (Aurora MySQL)	256	256

Característica de escalado y alta disponibilidad	Comportamiento de Aurora Serverless v2	Comportamiento de Aurora Serverless v1
ACU máximas (Aurora PostgreSQL)	256	384
Detención de un clúster	Puede detener e iniciar manualmente el clúster mediante la misma característica de parada e inicio de clúster como clústeres aprovisionados.	El clúster se detiene automáticamente tras un tiempo de espera. Lleva algún tiempo hasta que está disponible cuando se reanuda la actividad.
Escalado de las instancias de base de datos	Escalado horizontal y vertical con un incremento mínimo de 0,5 ACU.	Escalado horizontal y vertical duplicando o reduciendo a la mitad las AUC.
Número de instancias de base de datos	Igual que un clúster aprovisionado: 1 instancia de base de datos de escritor, hasta 15 instancias de base de datos de lector.	1 instancia de base de datos que gestiona lecturas y escrituras.
¿El escalado puede ocurrir mientras se ejecutan las instrucciones SQL?	Sí. Aurora Serverless v2 no requiere esperar un punto de silencio.	No. Por ejemplo, el escalado espera a que se completen las transacciones de larga duración, las tablas temporales y los bloqueos de tablas.
Las instancias de base de datos de lector escalan junto con el escritor	Opcional	No aplicable
Almacenamiento máximo	128 TiB	128 TiB
Se conserva la caché de búfer al escalar	Sí. La memoria caché del búfer cambia de tamaño dinámicamente.	No. La caché de búfer se recalienta después de escalar.

Característica de escalado y alta disponibilidad	Comportamiento de Aurora Serverless v2	Comportamiento de Aurora Serverless v1
Conmutación por error	Sí, igual que con los clústeres aprovisionados.	Solo el mejor esfuerzo, sujeto a la disponibilidad de capacidad. Más lento que en Aurora Serverless v2.
Capacidad Multi-AZ	Sí, igual que para los aprovisionados. Un clúster Multi-AZ requiere una instancia de base de datos de lector en una segunda zona de disponibilidad (AZ). Para un clúster Multi-AZ, Aurora realiza una conmutación por error Multi-AZ en caso de fallo de AZ.	Aurora Serverless v1Los clústeres de ejecutan toda su computación en una única zona de disponibilidad. La recuperación en caso de error de AZ es solo un esfuerzo óptimo y está sujeta a la disponibilidad de capacidad.
Bases de datos globales de Aurora	Sí	No
Escalado basado en la presión de memoria	Sí	No
Escalado basado en la carga de la CPU	Sí	Sí
Escalado basado en el tráfico de red	Sí, según la sobrecarga de memoria y CPU del tráfico de red. El parámetro `max_connections` se mantiene constante para evitar que se caigan las conexiones al reducir la escala.	Sí, según el número de conexiones.
Acción de tiempo de espera para escalar eventos	No	Sí

Característica de escalado y alta disponibilidad	Comportamiento de Aurora Serverless v2	Comportamiento de Aurora Serverless v1
Agregar nuevas instancias de base de datos al clúster mediante AWS Auto Scaling	No se usa. Puede crear instancias de base de datos de lector de Aurora Serverless v2 en los niveles de promoción 2 a 15 y dejarlas reducidas a una capacidad baja.	No. No hay instancias de base de datos de lector disponibles.

Comparación de la compatibilidad de características de Aurora Serverless v2 y Aurora Serverless v1

Se resumen en la tabla siguiente:

- Características que están disponibles en Aurora Serverless v2 pero no en Aurora Serverless v1
- Características que funcionan de forma diferente en Aurora Serverless v1 y en Aurora Serverless v2
- Características que no están disponibles actualmente en Aurora Serverless v2

Aurora Serverless v2 incluye muchas características de clústeres aprovisionados que no están disponibles en Aurora Serverless v1.

Característica	Aurora Serverless v2Compatibilidad con	Aurora Serverless v1Compatibilidad con
Topología de clúster	Aurora Serverless v2 es una propiedad de instancias de base de datos individuales. Un clúster puede contener varias instancias de base de datos de Aurora Serverless v2 o una combinación de Aurora Serverless v2 y de instancia	Aurora Serverless v1Los clústeres de no utilizan la noción de instancias de base de datos. Una vez creado el clúster, no se puede cambiar la propiedad de Aurora Serverless v1.

Característica	Aurora Serverless v2Compatibilidad con	Aurora Serverless v1Compatibilidad con
	s de base de datos aprovisionadas.	
Parámetros de configuración	Prácticamente se pueden modificar los mismos parámetros que en los clústeres aprovisionados. Para obtener más información, consulte Trabajo con los grupos de parámetros para Aurora Serverless v2.	Solo se puede modificar un subconjunto de parámetros.
Grupos de parámetros	Grupo de parámetros de clúster y grupos de parámetros de base de datos. Están disponibles los parámetros con valor `provisioned` en el atributo `SupportedEngineModes`. Son muchos más parámetros que en Aurora Serverless v1.	Solo grupo de parámetros del clúster. Están disponibles los parámetros con el valor `serverless` en el atributo `SupportedEngineModes`.
Cifrado para volumen del clúster	Opcional	Obligatorio. Las limitaciones de Limitaciones de los clústeres de base de datos cifrados de Amazon Aurora se aplican a todos los clústeres de Aurora Serverless v1.
Instantáneas entre regiones	Sí	La instantánea debe cifrarse con su propia clave AWS Key Management Service (AWS KMS).

Característica	Aurora Serverless v2Compatibilidad con	Aurora Serverless v1Compatibilidad con
Copias de seguridad automatizadas conservadas tras la eliminación del clúster de base de datos	Sí	No
TLS/SSL	Sí. La compatibilidad es la mismo que para los clústeres aprovisionados. Para obtener más información, consulte Uso de TLS/SSL con Aurora Serverless v2.	Sí. Existen algunas diferencias con respecto a la compatibilidad TLS para clústeres aprovisionados. Para obtener más información, consulte Uso de TLS/SSL con Aurora Serverless v1.
Clonación	Solo desde y hacia las versiones del motor de base de datos compatibles con Aurora Serverless v2. No puede usar la clonación para actualizar de Aurora Serverless v1 o de una versión anterior de un clúster aprovisionado.	Solo desde y hacia las versiones del motor de base de datos compatibles con Aurora Serverless v1.
Integración con Amazon S3	Sí	Sí
Cómo integrar con AWS Secrets Manager	Sí	No
Exportación de instantáneas del clúster de bases de datos a S3	Sí	No
Asociación de un rol de IAM	Sí	No

Característica	Aurora Serverless v2Compatibilidad con	Aurora Serverless v1Compatibilidad con
Carga de registros de Amazon CloudWatch	Opcional. Usted elige qué registros activar y qué registros cargar en CloudWatch.	Todos los registros activados se cargan automáticamente en CloudWatch.
API de datos disponible	Sí	Sí
Publicador de consultas disponible	Sí	Sí
Información de rendimiento	Sí	No
Proxy de Amazon RDS disponible	Sí	No
Babelfish for Aurora PostgreSQL disponible	Sí. Compatible con las versiones de Aurora PostgreSQL que son compatibles con Babelfish y Aurora Serverless v2.	No

Adaptación casos de uso de Aurora Serverless v1 para Aurora Serverless v2

En función de su caso de uso para Aurora Serverless v1, podrá adaptar ese enfoque para aprovechar las características de Aurora Serverless v2 de la siguiente manera.

Supongamos que tiene un clúster de Aurora Serverless v1 que está ligeramente cargado y su prioridad es mantener la disponibilidad continua y minimizar los costes. Con Aurora Serverless v2, puede configurar una valor de ACU mínimo menor de 0,5, en comparación con un mínimo de 1 ACU para Aurora Serverless v1. Puede aumentar la disponibilidad creando una configuración Multi-AZ, ya que la instancia de base de datos del lector también tiene un mínimo de 0,5 ACU.

Supongamos que tiene un clúster de Aurora Serverless v1 que utiliza en un escenario de desarrollo y prueba. En este caso, el costo también es de alta prioridad, pero el clúster no necesita estar disponible en todo momento. Aurora Serverless v2 puede pausar automáticamente cada instancia cuando esté completamente inactiva. Para ello, especifique una capacidad mínima de 0 ACU para el

clúster, como se explica en Escalado a cero ACU con pausa y reanudación automáticas para Aurora Serverless v2. Puede detener también manualmente el clúster cuando no sea necesario e iniciarlo cuando llegue el momento del próximo ciclo de prueba o desarrollo.

Supongamos que tiene un clúster de Aurora Serverless v1 con una carga de trabajo pesada. Se puede escalar un clúster equivalente mediante Aurora Serverless v2 con más granularidad. Por ejemplo, Aurora Serverless v1 escala duplicando la capacidad, por ejemplo, de 64 a 128 AUC. Por el contrario, la instancia de base de datos de Aurora Serverless v2 puede reducirse horizontalmente en incrementos de 0,5 ACU.

Supongamos que su carga de trabajo requiere una capacidad total superior a la disponible en Aurora Serverless v1. Puedes usar varias instancias de base de datos de lector de Aurora Serverless v2 para descargar las partes de lectura intensiva de la carga de trabajo de la instancia de base de datos de escritor. También puede dividir la carga de trabajo de lectura intensiva entre varias instancias de base de datos de lector.

Para una carga de trabajo de escritura intensiva, puede configurar el clúster con una instancia de base de datos aprovisionada de gran tamaño como escritor. Puede hacerlo junto con una o más instancias de base de datos de lector de Aurora Serverless v2.

Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

El proceso de actualización de un clúster de base de datos de Aurora Serverless v1 a Aurora Serverless v2 tiene varios pasos. Esto se debe a que no se puede convertir directamente de Aurora Serverless v1 a Aurora Serverless v2. Siempre hay un paso intermedio que implica convertir el clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado.

Clústeres de base de datos compatibles con Aurora MySQL

Puede convertir su clúster de base de datos de Aurora Serverless v1 en un clúster de base de datos aprovisionado y, a continuación, utilizar una implementación azul/verde para actualizarlo y convertirlo en un clúster de base de datos de Aurora Serverless v2. Recomendamos este procedimiento para los entornos de producción. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Para utilizar una implementación azul/verde para actualizar un clúster de Aurora Serverless v1 que ejecuta la versión 2 de Aurora MySQL (compatible con MySQL 5.7)

1. Convierta el clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado de Aurora MySQL versión 2. Siga el procedimiento indicado en Conversión de Aurora Serverless v1 a aprovisionada.
2. Creación de una implementación azul/verde Siga el procedimiento indicado en Creación de una implementación azul/verde en Amazon Aurora.
3. Elija una versión de Aurora MySQL para el clúster verde que sea compatible con Aurora Serverless v2, por ejemplo, la 3.04.1.

   Para conocer las versiones compatibles, consulte Aurora Serverless v2 con Aurora MySQL.
4. Modifique la instancia de base de datos del escritor del clúster verde para que utilice la clase de instancia de base de datos de Serverless v2 (db.serverless).

   Para obtener más información, consulte Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2.
5. Cuando su clúster de base de datos de Aurora Serverless v2 actualizado esté disponible, cambie del clúster azul al clúster verde.

Clústeres de base de datos compatibles con Aurora PostgreSQL

Puede convertir su clúster de base de datos de Aurora Serverless v1 en un clúster de base de datos aprovisionado y, a continuación, utilizar una implementación azul/verde para actualizarlo y convertirlo en un clúster de base de datos de Aurora Serverless v2. Recomendamos este procedimiento para los entornos de producción. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon Aurora para actualizar las bases de datos.

Para utilizar una implementación azul/verde para actualizar un clúster de Aurora Serverless v1 que ejecute la versión 11 de Aurora PostgreSQL

1. Convierta el clúster de base de datos de Aurora Serverless v1 en un clúster de Aurora PostgreSQL aprovisionado. Siga el procedimiento indicado en Conversión de Aurora Serverless v1 a aprovisionada.
2. Creación de una implementación azul/verde Siga el procedimiento indicado en Creación de una implementación azul/verde en Amazon Aurora.
3. Elija una versión de Aurora PostgreSQL para el clúster verde que sea compatible con Aurora Serverless v2, por ejemplo, la 15.3.

   Para conocer las versiones compatibles, consulte Aurora Serverless v2 con Aurora PostgreSQL.
4. Modifique la instancia de base de datos del escritor del clúster verde para que utilice la clase de instancia de base de datos de Serverless v2 (db.serverless).

   Para obtener más información, consulte Conversión de un escritor o un lector aprovisionados a Aurora Serverless v2.
5. Cuando su clúster de base de datos de Aurora Serverless v2 actualizado esté disponible, cambie del clúster azul al clúster verde.

También puede actualizar el clúster de base de datos de Aurora Serverless v1 directamente desde la versión 11 de Aurora PostgreSQL a la versión 13, convertirlo en un clúster de base de datos aprovisionado y, a continuación, convertir el clúster aprovisionado en un clúster de base de datos de Aurora Serverless v2.

Para actualizar y luego convertir un clúster de Aurora Serverless v1 que ejecute la versión 11 de Aurora PostgreSQL

1. Convierta el clúster de base de datos de Aurora Serverless v1 en un clúster de Aurora PostgreSQL aprovisionado. Siga el procedimiento indicado en Conversión de Aurora Serverless v1 a aprovisionada.
2. Actualice el clúster de Aurora Serverless v1 a una versión 13 de Aurora PostgreSQL que sea compatible con Aurora Serverless v2, por ejemplo, la 13.12. Siga el procedimiento indicado en Actualización de la versión principal.

   Para conocer las versiones compatibles, consulte Aurora Serverless v2 con Aurora PostgreSQL.

3. Agregue una instancia de base de datos de lectura Aurora Serverless v2 al clúster. Para obtener más información, consulte Adición de un lector Aurora Serverless v2.
4. Conmute por error a la instancia de base de datos de Aurora Serverless v2:
    a. Seleccione la instancia de base de datos del escritor del clúster de base de datos.
    b. En Actions (Acciones), elija Failover (conmutación por error).
    c. En la página de confirmación, seleccione Conmutación por error.

Para los clústeres de base de datos de Aurora Serverless v1 que ejecuten Aurora PostgreSQL versión 13, debe convertir el clúster de Aurora Serverless v1 en un clúster de base de datos aprovisionado y, a continuación, convertir el clúster aprovisionado en un clúster de base de datos de Aurora Serverless v2.

Para actualizar un clúster de Aurora Serverless v1 que ejecute la versión 13 de Aurora PostgreSQL

1. Convierta el clúster de base de datos de Aurora Serverless v1 en un clúster de Aurora PostgreSQL aprovisionado. Siga el procedimiento indicado en Conversión de Aurora Serverless v1 a aprovisionada.
2. Agregue una instancia de base de datos de lectura Aurora Serverless v2 al clúster. Para obtener más información, consulte Adición de un lector Aurora Serverless v2.
3. Conmute por error a la instancia de base de datos de Aurora Serverless v2:
    a. Seleccione la instancia de base de datos del escritor del clúster de base de datos.
    b. En Actions (Acciones), elija Failover (conmutación por error).
    c. En la página de confirmación, seleccione Conmutación por error.
4. Elimine la instancia de lector.

Migración de una base de datos en las instalaciones a Aurora Serverless v2

Puede migrar sus bases de datos en las instalaciones a Aurora Serverless v2, de igual modo que con Aurora MySQL y Aurora PostgreSQL.

- Para las bases de datos MySQL, puede usar el comando `mysqldump`. Para obtener más información, consulte Importación de datos a una base de datos de Amazon RDS para MySQL con tiempo de inactividad reducido en la Guía del usuario de Amazon Relational Database Service.

- Para las bases de datos PostgreSQL, puede usar los comandos `pg_dump` y `pg_restore`. Para obtener más información, consulte la publicación del blog Best practices for migrating PostgreSQL databases to Amazon RDS and Amazon Aurora (Prácticas recomendadas para migrar bases de datos de PostgreSQL a Amazon RDS y Amazon Aurora).

Uso de Amazon Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

Amazon Aurora Serverless v1 (Amazon Aurora sin servidor versión 1) es una configuración de escalado automático en diferido para Amazon Aurora. Un clúster de bases de datos de Aurora Serverless v1 es un clúster de bases de datos que escala la capacidad de cómputo y la disminuye en función de las necesidades de su aplicación. Esto contrasta con los clústeres de base de datos aprovisionados de Aurora, para los que se administra la capacidad de forma manual. Aurora Serverless v1 proporciona una opción rentable y relativamente simple para las cargas de trabajo poco frecuentes, intermitentes o impredecibles. Es rentable porque se inicia automáticamente, escala la capacidad de cómputo en función del uso de la aplicación y se cierra cuando no se utiliza.

Para obtener más información sobre precios, consulte Precios sin servidor en MySQL-Compatible Edition o PostgreSQL-Compatible Edition en la página Amazon Aurora pricing.

Aurora Serverless v1Los clústeres de tienen el mismo tipo de volumen de almacenamiento distribuido de alta capacidad y disponibilidad que utilizan los clústeres de base de datos aprovisionados.

Para un clúster de Aurora Serverless v1, el volumen del clúster siempre está cifrado. Puede elegir la clave de cifrado, pero no puede desactivar el cifrado. Esto significa que puede realizar en un Aurora Serverless v1 las mismas operaciones que realiza en instantáneas cifradas. Para obtener más información, consulte Aurora Serverless v1 e instantáneas.

Temas

- Disponibilidad de regiones y versiones para Aurora Serverless v1

- Ventajas de Aurora Serverless v1
- Casos de uso de Aurora Serverless v1
- Limitaciones de Aurora Serverless v1
- Requisitos de configuración para Aurora Serverless v1
- Uso de TLS/SSL con Aurora Serverless v1
- Cómo funciona Aurora Serverless v1
- Creación de un clúster de bases de datos de Aurora Serverless v1
- Restauración de un clúster de bases de datos de Aurora Serverless v1
- Modificación de un clúster de bases de datos de Aurora Serverless v1
- Escalado manual de la capacidad del clúster de bases de datos de Aurora Serverless v1
- Visualización de los clústeres de base de datos de Aurora Serverless v1
- Eliminación de un clúster de bases de datos de Aurora Serverless v1
- Aurora Serverless v1 y versiones del motor de base de datos de Aurora

Disponibilidad de regiones y versiones para Aurora Serverless v1

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos de Aurora y entre Regiones de AWS. Para obtener más información acerca de la versión y la disponibilidad de las regiones con Aurora y Aurora Serverless v1, consulte Aurora Serverless v1.

Ventajas de Aurora Serverless v1

Aurora Serverless v1 ofrece las siguientes ventajas:

- Más simple que el aprovisionado: Aurora Serverless v1 elimina gran parte de la complejidad de administrar instancias de base de datos y capacidad.
- Escalable: Aurora Serverless v1 escala sin problemas la capacidad de memoria y de cómputo en función de las necesidades, sin interrumpir las conexiones del cliente.
- Rentable: cuando utiliza Aurora Serverless v1, solo paga los recursos de base de datos que consume, contados por segundo.
- Almacenamiento de alta disponibilidad: Aurora Serverless v1 utiliza el mismo sistema de almacenamiento distribuido, con tolerancia a errores y reproducción de seis vías que Aurora para protegerlo de la pérdida de datos.

Casos de uso de Aurora Serverless v1

Aurora Serverless v1 está diseñado para los siguientes casos de uso:

- Aplicaciones de uso no frecuente: cuando tiene una aplicación que solo se utiliza durante unos minutos varias veces al día o a la semana, como una página de blog de bajo volumen. Con Aurora Serverless v1, solo paga por los recursos de bases de datos que consume, contados por segundo.
- Aplicaciones nuevas: si implementa una nueva aplicación y no está seguro del tamaño de instancia que necesita. Con Aurora Serverless v1, puede crear un punto de enlace de base de datos y hacer que esta última escale automáticamente según los requisitos de capacidad de la aplicación.
- Cargas de trabajo variables: cuando tiene una aplicación que usa poco y que presenta picos de entre 30 minutos y varias horas algunas veces al día o varias veces al año. Este es el caso, por ejemplo, de las aplicaciones de recursos humanos, presupuestos y generación de informes operativos. Con Aurora Serverless v1, ya no tiene que aprovisionar la capacidad para los picos ni para la carga promedio.
- Cargas de trabajo impredecibles: cuando ejecuta cargas de trabajo diarias que presentan aumentos de actividad repentinos e impredecibles. Un ejemplo de ello sería un sitio dedicado al tráfico, que experimenta un aumento repentino de la actividad cuando comienza a llover. Con Aurora Serverless v1, la capacidad de la base de datos escala automáticamente para satisfacer las necesidades de los picos de carga de la aplicación y vuelve a disminuir cuando termina el aumento de actividad.
- Bases de datos de desarrollo y prueba: sus desarrolladores usan las bases de datos durante las horas de trabajo, pero no las necesitan durante las noches ni los fines de semana. Con Aurora Serverless v1, la base de datos se apaga de forma automática cuando no se utiliza.
- Aplicaciones de varios inquilinos: con Aurora Serverless v1, no es preciso administrar individualmente la capacidad de la base de datos para cada aplicación de la flota. Aurora Serverless v1 administra la capacidad de la base de datos individual por usted.

Limitaciones de Aurora Serverless v1

Las limitaciones siguientes se aplican a:Aurora Serverless v1

- > **Important**
  >
  > AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

- Aurora Serverless v1 no admite las siguientes características de :
  - Bases de datos globales de Aurora
  - Réplicas de Aurora
  - AWS Identity and Access ManagementAutenticación de base de datos de (IAM)
  - Búsqueda de datos anteriores en Aurora.
  - Secuencias de actividades de la base de datos
  - Autenticación de Kerberos
  - Información de rendimiento
  - RDS Proxy
  - Visualización de registros en la Consola de administración de AWS
- Las conexiones a un clúster de bases de datos de Aurora Serverless v1 se cierran automáticamente si se mantienen abiertas durante más de un día.
- Todos los clústeres de base de datos de Aurora Serverless v1 tienen las siguientes limitaciones:
  - No se pueden exportar las instantáneas de Aurora Serverless v1 a los buckets de Amazon S3.
  - No puede usar ni cambiar AWS Database Migration Service la captura de datos (CDC) con clústeres de base de datos de Aurora Serverless v1. Sólo los clústeres de Aurora base de datos aprovisionados admiten CDC con AWS DMS como fuente.
  - No se pueden guardar datos en archivos de texto en Amazon S3 ni cargar datos de archivos de texto en Aurora Serverless v1 desde S3.
  - No se puede adjuntar un rol de IAM a un clúster de base de datos de Aurora Serverless v1. Sin embargo, puede cargar datos en Aurora Serverless v1 desde Amazon S3 utilizando la extensión de `aws_s3` con la función de `aws_s3.table_import_from_s3` y el parámetro de

`credentials`. Para obtener más información, consulte Importación de datos de Amazon S3 en un clúster de base de datos Aurora PostgreSQL.

- Cuando se utiliza el editor de consultas, se crea un secreto de Secrets Manager para que las credenciales de la base de datos accedan a la base de datos. Si elimina las credenciales del editor de consultas, el secreto asociado también se elimina de Secrets Manager. Este secreto no se puede recuperar después de eliminarlo.

- Los clústeres de base de datos basados en Aurora MySQL que ejecutan Aurora Serverless v1 no admiten lo siguiente:
  - Invocación de funciones de AWS Lambda desde el clúster de bases de datos de Aurora MySQL. Sin embargo, las funciones de AWS Lambda pueden realizar llamadas a su clúster de bases de datos de Aurora Serverless v1.
  - Restauración de una instantánea desde una instancia de base de datos que no sea de Aurora MySQL ni de RDS for MySQL.
  - Replicación de datos mediante la replicación basada en registros binarios (binlogs). Esta limitación se cumple independientemente de si el clúster de bases de datos basado en Aurora MySQL, Aurora Serverless v1 es la fuente o el destino de la reproducción. Para replicar datos en un clúster de bases de datos de Aurora Serverless v1 desde una instancia de base de datos de MySQL fuera de Aurora, como una que se ejecuta en Amazon EC2, considere utilizar AWS Database Migration Service. Para obtener más información, consulte la Guía del usuario de AWS Database Migration Service.
  - Creación de usuarios con acceso basado en host (`'username'@'IP_address'`). Esto se debe a que Aurora Serverless v1 utiliza una flota de enrutadores entre el cliente y el host de la base de datos para lograr un escalado fluido. La dirección IP que el clúster de base de datos de Aurora Serverless ve es la del host del router y no la de su cliente. Para obtener más información, consulte Aurora Serverless v1 architecture.

    En su lugar, utilice el carácter comodín (`'username'@'%'`).

- Los clústeres de base de datos basados en Aurora PostgreSQL que ejecutan Aurora Serverless v1 tienen las siguientes limitaciones:
  - La administración de planes de consultas de Aurora PostgreSQL (extensión `apg_plan_management`) no es compatible.
  - La característica de replicación lógica disponible en Amazon RDS PostgreSQL y en Aurora PostgreSQL no es compatible.
  - Las comunicaciones salientes, como aquellas habilitadas por las extensiones de Amazon RDS for PostgreSQL no son compatibles. Por ejemplo, no se puede acceder a datos externos con la

extensión `postgres_fdw/dblink`. Para obtener más información acerca de las extensiones de RDS PostgreSQL, consulte PostgreSQL en Amazon RDS en la Guía del usuario de RDS.

- Actualmente, no se recomiendan ciertas consultas y comandos de SQL. Estos incluyen los bloqueos de asesoramiento a nivel de sesión, las relaciones temporales, las notificaciones asíncronas (`LISTEN`) y los cursores con retención (`DECLARE` *`name`* `... CURSOR WITH HOLD FOR` *`query`*). Además, los comandos de `NOTIFY` evitan el escalado y no se recomiendan.

  Para obtener más información, consulte Escalado automático para Aurora Serverless v1.

- No puede establecer el periodo de copia de seguridad automático preferido para un clúster de base de datos de Aurora Serverless v1.
- Puede configurar el periodo de mantenimiento de un clúster de base de datos de Aurora Serverless v1. Para obtener más información, consulte Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos.

Requisitos de configuración para Aurora Serverless v1

Cuando cree un clúster de bases de datos de Aurora Serverless v1, preste atención a los siguientes requisitos:

- Utilice estos números de puerto específicos para cada motor de base de datos:
  - Aurora MySQL: – `3306`
  - Aurora PostgreSQL – `5432`
- Cree sus clústeres de bases de datos de Aurora Serverless v1 en una nube virtual privada (VPC) basada en el servicio de Amazon VPC. Al crear un clúster de bases de datos de Aurora Serverless v1 en la VPC, se consumen dos (2) de los cincuenta (50) puntos de enlace de interfaz y balanceadores de carga de gateway asignados a la VPC. Estos extremos se crean automáticamente para usted. Para aumentar su cuota, puede ponerse en contacto con Soporte. Para obtener más información, consulte Amazon VPC Cupos.
- No se puede asignar una dirección IP pública a un clúster de bases de datos de Aurora Serverless v1. Sólo puede acceder a un clúster de bases de datos de Aurora Serverless v1 desde una VPC.
- Cree subredes en diferentes zonas de disponibilidad para el grupo de subredes de base de datos que utilice para el clúster de bases de datos de Aurora Serverless v1. En otras palabras, no puede tener más de una subred en la misma zona de disponibilidad.
- Los cambios realizados en un grupo de subredes utilizado por un clúster de bases de datos de Aurora Serverless v1 no se aplican al clúster.

- Puede acceder a un clúster de bases de datos de Aurora Serverless v1 desde AWS Lambda. Para hacerlo, debe configurar la función de Lambda para que se ejecute en la misma VPC que su clúster de bases de datos de Aurora Serverless v1. Para obtener más información sobre el uso de AWS Lambda, consulte Configuración de una función de Lambda para acceder a los recursos de una Amazon VPC en la Guía para desarrolladores deAWS Lambda.

Uso de TLS/SSL con Aurora Serverless v1

De forma predeterminada, Aurora Serverless v1 utiliza el protocolo de Transport Layer Security/Secure Sockets Layer (TLS/SSL) para cifrar las comunicaciones entre los clientes y el clúster de bases de datos de Aurora Serverless v1. Admite TLS/SSL versiones 1.0, 1.1 y 1.2. No necesita configurar su clúster de bases de datos de Aurora Serverless v1 para que use TLS/SSL.

Sin embargo, se aplican las siguientes limitaciones:

- Actualmente, la compatibilidad con TLS/SSL de los clústeres de base de datos de Aurora Serverless v1 no está disponible en la Región de AWS de China (Pekín).
- Cuando cree usuarios de bases de datos para un clúster de bases de datos de Aurora Serverless v1 basado en Aurora MySQL, no utilice la cláusula `REQUIRE` para los permisos SSL. Esto impide que los usuarios se conecten a la instancia de base de datos de Aurora.
- Para las utilidades de clientes MySQL y PostgreSQL, las variables de sesión que se pueden utilizar en otros entornos no tienen ningún efecto cuando se utiliza TLS/SSL entre el cliente y Aurora Serverless v1.
- Para el cliente MySQL, cuando se conecta con el modo `VERIFY_IDENTITY` de TLS/SSL, actualmente se necesita usar el comando de `mysql` compatible con MySQL 8.0. Para obtener más información, consulte Conexión a una instancia de base de datos que ejecuta el motor de base de datos MySQL.

Dependiendo del cliente que utilice para conectarse al clúster de bases de datos de Aurora Serverless v1, es posible que no necesite especificar TLS/SSL para obtener una conexión cifrada. Por ejemplo, para utilizar el cliente PostgreSQL para conectarse a un clúster de bases de datos de Aurora Serverless v1 que ejecuta la Edición ccompatible con Aurora PostgreSQL, conéctese como lo hace normalmente.

```
psql -h endpoint -U user
```

Después de escribir la contraseña, el cliente PostgreSQL le muestra los detalles de la conexión, incluida la versión de TLS/SSL y el cifrado.

```
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1), server 10.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.
```

Important

Aurora Serverless v1 utiliza el protocolo Transport Layer Security/Secure Sockets Layer (TLS/SSL) para cifrar las conexiones de forma predeterminada, a menos que la aplicación del cliente deshabilite SSL/TLS. La conexión TLS/SSL termina en la flota de enrutadores. La comunicación entre la flota de enrutadores y el clúster de bases de datos de Aurora Serverless v1 se produce dentro del límite de la red interna del servicio.
Puede verificar el estado de la conexión del cliente para verificar si la conexión a Aurora Serverless v1 está cifrada con TLS/SSL. Las tablas `pg_stat_ssl` y `pg_stat_activity` de PostgreSQL y su función `ssl_is_used` no muestran el estado TLS/SSL de la comunicación entre la aplicación del cliente y Aurora Serverless v1. Del mismo modo, el estado TLS/SSL no se puede derivar de la instrucción `status` de MySQL.
Los parámetros de clúster de Aurora `force_ssl` para PostgreSQL y `require_secure_transport` para MySQL no son compatibles con Aurora Serverless v1. Estos parámetros están disponibles ahora para Aurora Serverless v1. Para obtener una lista completa de los parámetros compatibles con Aurora sin servidor v1, llame a la operación de API DescribeEngineDefaultclústerParameters. Para obtener más información sobre los grupos de parámetros y Aurora sin servidor v1, consulte Grupos de parámetros de Aurora Serverless v1.

Para utilizar el cliente MySQL para conectarse a un clúster de bases de datos de Aurora Serverless v1 que ejecuta la Edición compatible con Aurora MySQL, especifique TLS/SSL en su solicitud. En el ejemplo siguiente se incluye el almacén de confianza Amazon Root CA 1 descargado de Amazon Trust Services, que es necesario para que esta conexión se realice de manera correcta.

```
mysql -h endpoint -P 3306 -u user -p --ssl-ca=amazon-root-CA-1.pem --ssl-mode=REQUIRED
```

Escriba la contraseña cuando se le solicite. Pronto, se abrirá el monitor MySQL. Puede confirmar que la sesión está cifrada usando el comando `status`.

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.5.62, for Linux (x86_64) using readline 5.1
Connection id:          19
Current database:
Current user:           ***@*******
SSL:                    Cipher in use is ECDHE-RSA-AES256-SHA
...
```

Para obtener más información acerca de cómo conectarse a la base de datos de Aurora MySQL con el cliente MySQL, consulte Conexión a una instancia de base de datos que ejecuta el motor de base de datos MySQL.

Aurora Serverless v1 admite todos los modos de TLS/SSL disponibles para el cliente MySQL (`mysql`) y el cliente PostgreSQL (`psql`), incluidos los enumerados en la tabla siguiente.

Descripción del modo TLS/SSL	mysql	psql
Conéctese sin utilizar TLS/SSL.	DISABLED	disable
Intente conectarse usando TLS/SSL primero, pero vuelva a no SSL si es necesario.	PREFERRED	preferir (predeterminado)
Aplicar mediante TLS/SSL.	REQUIRED	require
Aplique TLS/SSL y verifique la CA.	VERIFY_CA	verify-ca
Aplique TLS/SSL, verifique la CA y compruebe el nombre de alojamiento de la CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v1 utiliza certificados comodín. Si especifica la opción “verify CA (verificar CA)” o “verify CA and CA hostname (verificar CA y nombre de alojamiento de CA)” al utilizar TLS/SSL, descargue primero el almacén de confianza Amazon Root CA 1 de Amazon Trust Services. Después

de hacerlo, puede identificar este archivo con formato PEM en el comando del cliente. Para hacerlo utilizando el cliente PostgreSQL:

Para Linux, macOS o Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Para obtener más información acerca de cómo trabajar con la base de datos de Aurora PostgreSQL usando el cliente PostgreSQL, consulte Conexión a una instancia de base de datos que ejecuta el motor de base de datos PostgreSQL.

Para obtener más información acerca de cómo conectarse a los clústeres de base de datos de Aurora en general, consulte Conexión a un clúster de base de datos Amazon Aurora.

Conjuntos de cifrado compatibles para conexiones a clústeres de base de datos de Aurora Serverless v1

Mediante el uso de conjuntos de cifrado configurables, puede tener más control sobre la seguridad de las conexiones de su base de datos. Puede especificar una lista de conjuntos de cifrado que desea permitir para proteger las conexiones TLS/SSL del cliente a su base de datos. Con conjuntos de cifrado configurables, puede controlar el cifrado de conexión que acepta el servidor de base de datos. Esto evita el uso de cifrados que no son seguros o que ya no se utilizan.

Los clústeres de base de datos de Aurora Serverless v1 basados en Aurora MySQL admiten los mismos conjuntos de cifrado que los clústeres de base de datos aprovisionados de Aurora MySQL. Para obtener información sobre estos conjuntos de cifrado, consulte Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora MySQL.

Los clústeres de base de datos de Aurora Serverless v1 basados en Aurora PostgreSQL no admiten conjuntos de cifrado.

Cómo funciona Aurora Serverless v1

Important

AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren

antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

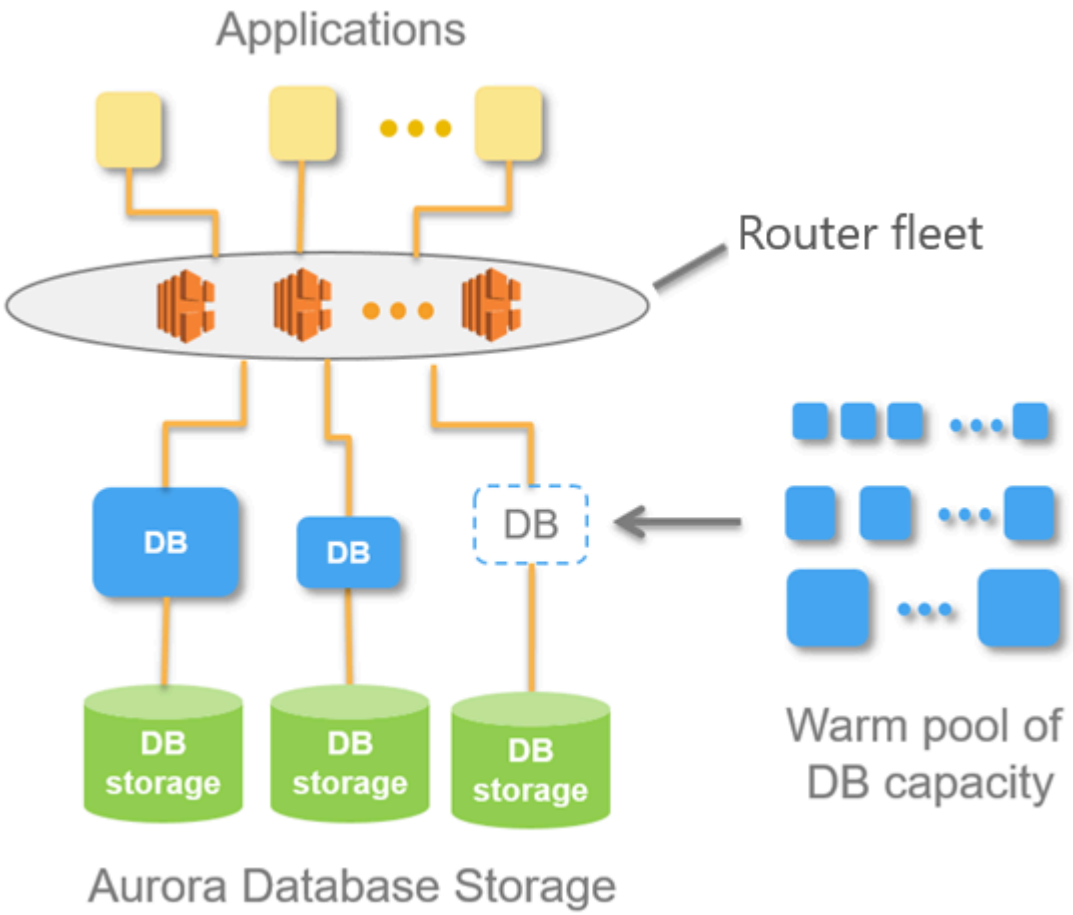
A continuación, le enseñaremos cómo funciona Aurora Serverless v1.

Temas

- Aurora Serverless v1 architecture
- Escalado automático para Aurora Serverless v1
- Acción de tiempo de espera para cambios de capacidad
- Pausar y reanudar Aurora Serverless v1
- Determinación del número máximo de conexiones de base de datos para Aurora Serverless v1
- Grupos de parámetros de Aurora Serverless v1
- Registros en Aurora Serverless v1
- Aurora Serverless v1 y mantenimiento
- Aurora Serverless v1 y conmutación por error
- Aurora Serverless v1 e instantáneas

Aurora Serverless v1 architecture

La siguiente imagen es una visión general de la arquitectura de Aurora Serverless v1.



En lugar de aprovisionar y administrar los servidores de base de datos, se especifican unidades de capacidad de Aurora (ACU). Cada ACU es una combinación de aproximadamente 2 gigabytes (GB) de memoria, la CPU correspondiente y las redes. El almacenamiento de base de datos escala automáticamente de 10 gibibytes (GiB) a 128 tebibytes (TiB), al igual que el almacenamiento en un clúster de bases de datos de Aurora estándar.

Puede especificar las ACU mínima y máxima. La unidad de capacidad de Aurora mínima es la ACU más baja a la que puede escalarse el clúster de bases de datos al reducir la capacidad. La unidad de capacidad de Aurora máxima es la ACU más baja a la que puede escalarse el clúster de bases de datos al aumentar la capacidad. En función de la configuración, Aurora Serverless v1 crea de manera automática reglas de escalado para los límites de uso de la CPU, las conexiones y la memoria disponible.

Aurora Serverless v1 administra el grupo activo de recursos de una Región de AWS para minimizar el tiempo de escalado. Cuando Aurora Serverless v1 agrega nuevos recursos al clúster de bases de datos de Aurora, utiliza la flota de enrutadores para cambiar las conexiones de clientes activas a los

nuevos recursos. En cualquier momento dado, solo se le cobrarán las ACU que se estén utilizando activamente en su clúster de base de datos de Aurora.

Escalado automático para Aurora Serverless v1

La capacidad asignada a su clúster de bases de datos de Aurora Serverless v1 se amplía y reduce sin problemas en función de la carga generada por la aplicación cliente. Aquí, la carga es la utilización de la CPU y el número de conexiones. Cuando la capacidad está limitada por cualquiera de ellos, Aurora Serverless v1 aumenta. Aurora Serverless v1 también se amplía cuando detecta problemas de rendimiento que se pueden resolver de esta manera.

Puede ver los eventos de escalado para su clúster de Aurora Serverless v1 en la Consola de administración de AWS. Durante el escalado automático, Aurora Serverless v1 restablece la métrica de `EngineUptime`. El valor de la métrica de restablecimiento no significa que el escalado perfecto haya tenido problemas ni que Aurora Serverless v1 haya eliminado conexiones. Es simplemente el punto de partida para el tiempo de actividad en la nueva capacidad. Para obtener más información sobre las métricas, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Cuando el clúster de base de datos de Aurora Serverless v1 no tiene conexiones activas, puede reducir la capacidad a cero (0 ACU). Para obtener más información, consulte Pausar y reanudar Aurora Serverless v1.

Cuando necesita realizar una operación de escalado, Aurora Serverless v1 primero intenta identificar un punto de escalado, un momento en el que no se estén procesando consultas. Aurora Serverless v1 podría no encontrar un punto de escalado por las siguientes razones:

- Consultas de larga duración
- Transacciones en curso
- Tablas temporales o bloqueos de tabla

Para aumentar la tasa de éxito del clúster de bases de datos de Aurora Serverless v1 al encontrar un punto de escalado, le recomendamos que evite las consultas y transacciones de larga duración. Para obtener más información sobre las operaciones que bloquean el escalado y cómo evitarlas, consulte Best practices for working with Aurora Serverless v1.

De forma predeterminada, Aurora Serverless v1 intenta encontrar un punto de escalado durante 5 minutos (300 segundos). Puede especificar un período de tiempo de espera distinto al crear o modificar el clúster. El tiempo de espera puede ser de 60 segundos a 10 minutos (600 segundos).

Si Aurora Serverless v1 no puede encontrar ningún punto de escalado en el período especificado, el tiempo de espera de la operación de escalado automático se agotará.

De forma predeterminada, si el escalado automático no encuentra un punto de escalado antes de agotar el tiempo de espera, Aurora Serverless v1 mantiene el clúster en la capacidad actual. Puede cambiar este comportamiento predeterminado al crear o modificar el clúster de bases de datos de Aurora Serverless v1 al seleccionar la opción Force capacity change (Forzar el cambio de capacidad). Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.

Acción de tiempo de espera para cambios de capacidad

Si se agota el tiempo de espera del escalado automático y no se encuentra ningún punto de escalado, Aurora mantendrá la capacidad actual de forma predeterminada. Puede elegir que Aurora fuerce el cambio al seleccionar la opción Force the capacity change (Forzar cambio de capacidad). Esta opción está disponible en la sección Autoscaling timeout and action (Acción y tiempo de espera de escalado automático) de la página Create database (Crear base de datos) cuando se crea el clúster.

De manera predeterminada, la opción Force the capacity change (Forzar cambio de capacidad) no está seleccionada. Deje esta opción desactivada para que la capacidad del clúster de base de datos de Aurora Serverless v1 permanezca sin cambios si la operación de escalado agota el tiempo de espera sin encontrar un punto de escalado.

Al elegir esta opción, el clúster de base de datos de Aurora Serverless v1 aplica el cambio de capacidad, incluso sin un punto de escalado. Antes de seleccionar esta opción, tenga en cuenta las consecuencias:

- Cualquier transacción en proceso se interrumpe y aparece el siguiente mensaje de error.

  Aurora MySQL versión 2: ERROR 1105 (HY000): La última transacción se ha interrumpido debido a la escalación sin interrupciones. Vuelva a intentarlo.

  Podrá volver a enviar la transacción tan pronto como el clúster de bases de datos de Aurora Serverless v1 esté disponible.
- Se eliminan las conexiones a las tablas y bloqueos temporales.

  Recomendamos que elija la opción Force the capacity change (Forzar cambio de capacidad) solo si su aplicación puede recuperarse de conexiones caídas o transacciones incompletas.

Las opciones que elige en la Consola de administración de AWS cuando crea un clúster de base de datos de Aurora Serverless v1 se almacenan en el objeto `ScalingConfigurationInfo` y en las propiedades `SecondsBeforeTimeout` y `TimeoutAction`. El valor de la propiedad `TimeoutAction` se establece en uno de los siguientes valores al crear el clúster:

- `RollbackCapacityChange`: este valor se establece cuando se elige la opción Roll back the capacity change (Revertir el cambio de capacidad). Este es el comportamiento predeterminado.
- `ForceApplyCapacityChange`: este valor se establece cuando se elige la opción Force the capacity change (Forzar el cambio de capacidad).

Puede obtener el valor de esta propiedad en un clúster de bases de datos de Aurora Serverless v1 existente mediante el comando de la AWS CLI describe-db-clusters, como se muestra a continuación.

Para Linux, macOS o Unix:

```
aws rds describe-db-clusters --region region \
  --db-cluster-identifier your-cluster-name \
  --query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'
```

Para Windows:

```
aws rds describe-db-clusters --region region ^
  --db-cluster-identifier your-cluster-name ^
  --query "*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}"
```

A modo de ejemplo, a continuación se muestra la consulta y la respuesta para un clúster de bases de datos de Aurora Serverless v1 denominado `west-coast-sles` en la región de EE. UU Oeste (N. California).

```
$ aws rds describe-db-clusters --region us-west-1 --db-cluster-identifier west-coast-
sles
--query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'

[
    {
        "ScalingConfigurationInfo": {
            "MinCapacity": 1,
            "MaxCapacity": 64,
```

```
            "AutoPause": false,
            "SecondsBeforeTimeout": 300,
            "SecondsUntilAutoPause": 300,
            "TimeoutAction": "RollbackCapacityChange"
        }
    }
]
```

Como muestra la respuesta, este clúster de bases de datos de Aurora Serverless v1 utiliza la configuración predeterminada.

Para obtener más información, consulte Creación de un clúster de bases de datos de Aurora Serverless v1. Después de crear su Aurora Serverless v1, puede modificar la acción de tiempo de espera y otros ajustes de capacidad en cualquier momento. Para saber cómo hacerlo, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1.

Pausar y reanudar Aurora Serverless v1

Puede optar por pausar el clúster de bases de datos de Aurora Serverless v1 después de transcurrido un tiempo determinado sin actividad. Usted especifica el periodo sin actividad que deberá transcurrir antes de poner en pausa el clúster de bases de datos. Al seleccionar esta opción, el tiempo de inactividad predeterminado es de cinco minutos, pero puede cambiar este valor. Se trata de una configuración opcional.

Cuando el clúster de bases de datos está en pausa, no se produce ningún tipo de actividad de computación ni de memoria y solamente se le cobra el almacenamiento. Si se solicitan conexiones a la base de datos mientras un clúster de bases de datos de Aurora Serverless v1 está en pausa, este clúster se reanuda automáticamente y atiende las solicitudes de conexión.

Cuando el clúster de bases de datos reanuda la actividad, tiene la misma capacidad que tenía cuando Aurora detuvo el clúster. El número de ACU depende de cuánto aumentó o redujo Aurora la escala del clúster antes de ponerlo en pausa.

> **Note**
>
> Si un clúster de bases de datos está en pausa durante más de siete días, puede hacerse una copia de seguridad de él mediante una instantánea. En este caso, Aurora restaura el clúster de bases de datos desde la instantánea cuando hay una solicitud para conectarse al mismo.

Determinación del número máximo de conexiones de base de datos para Aurora Serverless v1

Los siguientes ejemplos corresponden a un clúster de base de datos de Aurora Serverless v1 que es compatible con MySQL 5.7. Puede utilizar un cliente MySQL o el editor de consultas, si ha configurado el acceso a él. Para obtener más información, consulte Ejecución de consultas en el editor de consultas.

Para encontrar el número máximo de conexiones de base de datos

1. Busque el rango de capacidad de su cluster de base de datos de Aurora Serverless v1 mediante la AWS CLI.

   ```
   aws rds describe-db-clusters \
       --db-cluster-identifier my-serverless-57-cluster \
       --query 'DBClusters[*].ScalingConfigurationInfo|[0]'
   ```

   El resultado muestra que su rango de capacidad es de 1 a 4 ACU.

   ```
   {
       "MinCapacity": 1,
       "AutoPause": true,
       "MaxCapacity": 4,
       "TimeoutAction": "RollbackCapacityChange",
       "SecondsUntilAutoPause": 3600
   }
   ```

2. Ejecute la siguiente consulta SQL para encontrar el número máximo de conexiones.

   ```
   select @@max_connections;
   ```

   El resultado mostrado es para la capacidad mínima del clúster, 1 ACU.

   ```
   @@max_connections
   90
   ```

3. Escale el clúster de 8 a 32 ACU.

   Para obtener más información sobre el escalado, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1.

4. Confirme el rango de capacidad.

```
{
    "MinCapacity": 8,
    "AutoPause": true,
    "MaxCapacity": 32,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

5. Busque el número máximo de conexiones.

```
select @@max_connections;
```

   El resultado mostrado corresponde a la capacidad mínima del clúster, 8 ACU.

```
@@max_connections
1000
```

6. Escale el clúster al máximo posible, 256 a 256 ACU.
7. Confirme el rango de capacidad.

```
{
    "MinCapacity": 256,
    "AutoPause": true,
    "MaxCapacity": 256,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

8. Busque el número máximo de conexiones.

```
select @@max_connections;
```

   El resultado mostrado corresponde a 256 ACU.

```
@@max_connections
6000
```

> **Note**
>
> El valor de `max_connections` no se escala linealmente con el número de ACU.

9. Vuelva a escalar el clúster a 1 a 4 ACU.

```
{
    "MinCapacity": 1,
    "AutoPause": true,
    "MaxCapacity": 4,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

   Esta vez, el valor de `max_connections` corresponde a 4 ACU.

```
@@max_connections
270
```

10. Vuelva a reducir verticalmente el clúster a 2 ACU.

```
@@max_connections
180
```

    Si ha configurado el clúster para que se detenga después de un tiempo de inactividad, se reduce verticalmente a 0 ACU. Sin embargo,`max_connections` no cae por debajo del valor de 1 ACU.

```
@@max_connections
90
```

Grupos de parámetros de Aurora Serverless v1

Cuando crea el clúster de bases de datos de Aurora Serverless v1, elige un motor de base de datos de Aurora y un grupo de parámetros de clúster de bases de datos asociado. A diferencia de los clústeres de base de datos aprovisionados de Aurora, un clúster de bases de datos de Aurora Serverless v1 tiene una única instancia de base de datos de lectura y escritura configurada con un grupo de parámetros de clúster de bases de datos— no tiene un grupo de parámetros de base de

datos aparte. Durante el escalado automático, Aurora Serverless v1 necesita poder cambiar los parámetros para que el clúster funcione mejor para el aumento o la disminución de la capacidad. Por lo tanto, con un clúster de base de datos de Aurora Serverless v1, es posible que no se apliquen algunos de los cambios que puede realizar en los parámetros de un tipo de motor de base de datos determinado.

Por ejemplo, un clúster de bases de datos de Aurora Serverless v1 basado en –Aurora PostgreSQL no puede usar `apg_plan_mgmt.capture_plan_baselines` ni otros parámetros que podrían utilizarse en clústeres de base de datos de Aurora PostgreSQL para la administración del plan de consultas.

Puede obtener una lista de valores predeterminados de los grupos de parámetros predeterminados para los diferentes motores de base de datos de Aurora mediante el comando describe-engine-default-cluster-parameters de la CLI y consultando la Región de AWS. Estos son los valores que puede utilizar para la opción `--db-parameter-group-family`.

Aurora MySQL versión 2	`aurora-mysql5.7`
Aurora PostgreSQL versión 11	`aurora-postgresql11`
Aurora PostgreSQL versión 13	`aurora-postgresql13`

Recomendamos que configure la AWS CLI con su ID de clave de acceso de AWS y la clave de acceso secreta de AWS y que establezca su Región de AWS antes de usar los comandos de la AWS CLI. Si proporciona la región a la configuración de la CLI, se evita ingresar al parámetro `--region` cuando ejecuta comandos. Para obtener más información sobre cómo configurar la AWS CLI, consulte Conceptos básicos de la configuración en la Guía del usuario de AWS Command Line Interface.

En el ejemplo siguiente, se obtiene una lista de parámetros del grupo de clústeres de base de datos predeterminado para la versión 2 de Aurora MySQL.

Para Linux, macOS o Unix:

```
aws rds describe-engine-default-cluster-parameters \
  --db-parameter-group-family aurora-mysql5.7 --query \
  'EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, `serverless`) == `true`] | [*].{param:ParameterName}' \
```

```
  --output text
```

Para Windows:

```
aws rds describe-engine-default-cluster-parameters ^
   --db-parameter-group-family aurora-mysql5.7 --query ^
   "EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, 'serverless') == `true`] | [*].{param:ParameterName}" ^
   --output text
```

Modificar los valores de parámetros para Aurora Serverless v1

Como se explica en Grupos de parámetros para Amazon Aurora, no puede cambiar directamente los valores de un grupo de parámetros predeterminado, independientemente de su tipo (grupo de parámetros de clúster de bases de datos, grupo de parámetros de base de datos). En su lugar, se crea un grupo de parámetros personalizado basado en el grupo de parámetros de clúster de bases de datos predeterminado para su motor de base de datos de Aurora y se cambia la configuración según sea necesario en ese grupo de parámetros. Por ejemplo, es posible que desee cambiar algunos de los ajustes de su clúster de bases de datos de Aurora Serverless v1 para registrar consultas o cargar registros específicos del motor de base de datos a Amazon CloudWatch.

Para crear un grupo de parámetros de clúster de bases de datos personalizado

1. Inicie sesión en Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Parameter groups (Grupos de parámetros).
3. Elija Create parameter group (Crear grupo de parámetros) para abrir el panel “Parameter group details (Detalles del grupo de parámetros)”.
4. Elija el grupo de clúster de bases de datos predeterminado adecuado para el motor de base de datos que desea utilizar para su clúster de bases de datos de Aurora Serverless v1. Asegúrese de elegir las siguientes opciones:

   a. En Parameter group family (Familia de grupos de parámetros), elija la familia adecuada para el motor de base de datos elegido. Asegúrese de que su selección tenga el prefijo `aurora-` en el nombre.

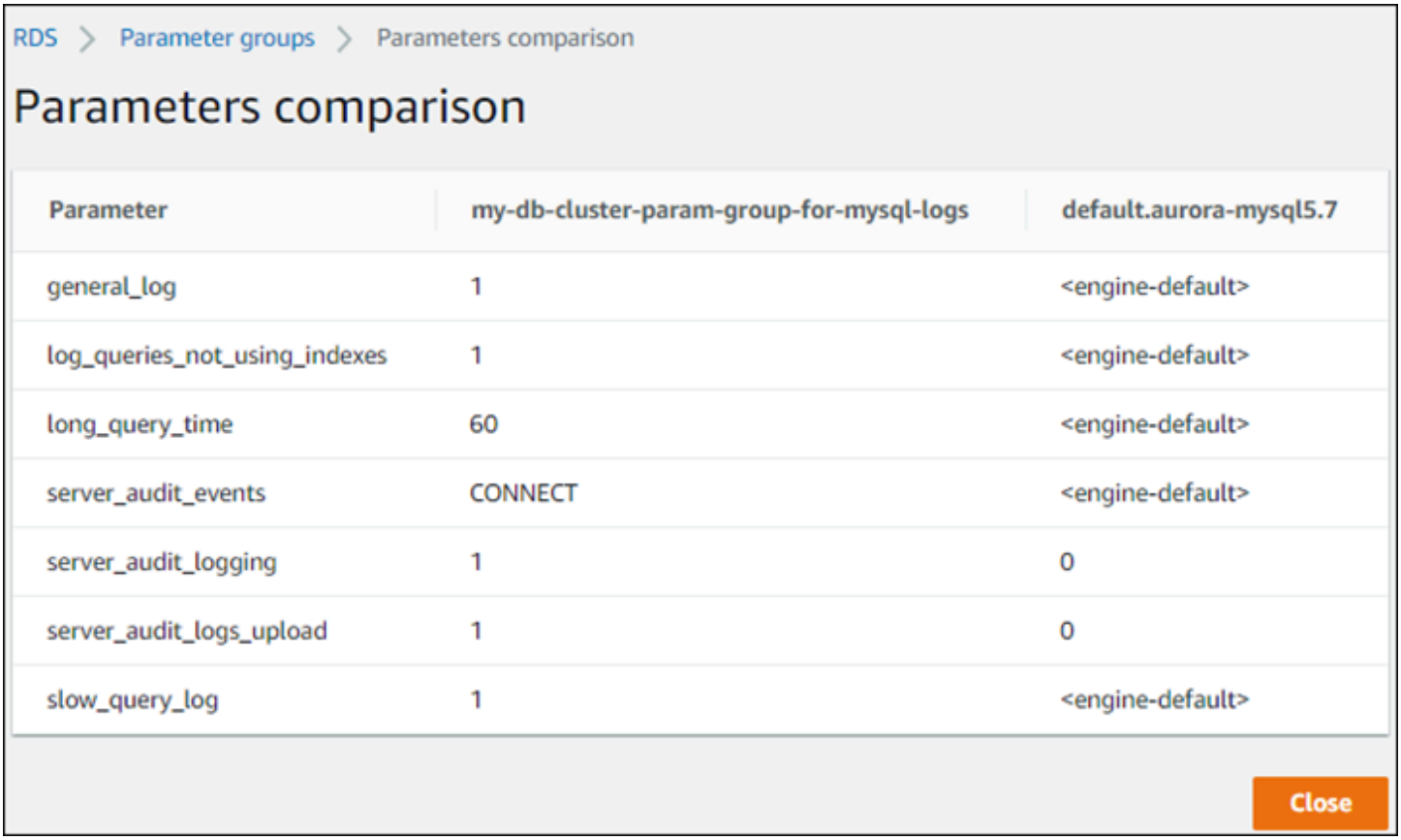
   b. En Type (Tipo), elija DB Cluster Parameter Group (Grupo de parámetros de clúster de bases de datos).

c. En Group name (Nombre de grupo) y Description (Descripción), escriba nombres significativos para usted u otras personas que puedan necesitar trabajar con el clúster de bases de datos de Aurora Serverless v1 y sus parámetros.

d. Seleccione Create (Crear).

El grupo de parámetros de clúster de base de datos personalizado se añade a la lista de grupos de parámetros disponibles en su Región de AWS. Puede utilizar su grupo de parámetros de clúster de base de datos personalizado al crear nuevos clústeres de base de datos de Aurora Serverless v1. También puede modificar un clúster de base de datos de Aurora Serverless v1 existente para utilizar su grupo de parámetros de clúster de base de datos personalizado. Cuando su clúster de base de datos de Aurora Serverless v1 comienza a utilizar el grupo de parámetros de clúster de base de datos personalizado, puede cambiar los valores de los parámetros dinámicos mediante la Consola de administración de AWS o la AWS CLI.

También puede utilizar la consola para ver una comparación paralela de los valores del grupo de parámetros de clúster de base de datos personalizado y el grupo de parámetros de clúster de base de datos predeterminado, como se muestra en la siguiente captura de pantalla.



Cuando cambia los valores de los parámetros en un clúster de bases de datos activo, Aurora Serverless v1 inicia un escalado perfecto para aplicar los cambios de parámetro. Si el clúster de base de datos de Aurora Serverless v1 está en un estado en pausa, se reanuda y comienza a escalarse para que pueda realizar el cambio. La operación de escalado para un cambio de grupo de parámetros siempre fuerza el cambio de capacidad , así que tenga en cuenta que la modificación de los parámetros podría causar caídas en la conexión si no se puede encontrar un punto de escalado durante el periodo de escalado.

Registros en Aurora Serverless v1

De forma predeterminada, los registros de errores están habilitados para Aurora Serverless v1 y se cargan automáticamente a Amazon CloudWatch. También puede hacer que su clúster de base de datos de Aurora Serverless v1 cargue los registros específicos del motor de base de datos de Aurora en CloudWatch Para ello, habilite los parámetros de configuración en el grupo de parámetros del clúster de base de datos personalizado. Su clúster de base de datos de Aurora Serverless v1 carga todos los registros disponibles en Amazon CloudWatch. En este punto, Amazon CloudWatch le permite analizar los datos de registro, crear alarmas y ver métricas.

Para Aurora MySQL, la siguiente tabla muestra los registros que puede habilitar. Cuando se habilitan, se cargan automáticamente desde su clúster de base de datos de Aurora Serverless v1 a Amazon CloudWatch.

Registro de Aurora MySQL	Descripción
`general_log`	Crea el registro general. Se establece en 1 para activarlo. El valor predeterminado es desactivado (0).
`log_queries_not_using_indexes`	Registra todas las consultas en el registro de consultas lentas que no utilizan un índice. El valor predeterminado es desactivado (0). Se establece en 1 para activar este registro.
`long_query_time`	Evita que las consultas de ejecución rápida se registren en el registro de consultas lentas. Se puede establecer en un número flotante entre 0 y 3,1536,000. El valor predeterminado es 0 (no activo).

Registro de Aurora MySQL	Descripción
`server_audit_events`	La lista de eventos que se deben capturar en los registros. Los valores admitidos son `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` y `QUERY_DML` y `TABLE`.
`server_audit_logging`	Se establece en 1 para activar el registro de auditoría del servidor. Si activa esta opción, puede especificar los eventos de auditoría que se enviarán a CloudWatch enumerándolos en el parámetro de `server_audit_events` .
`slow_query_log`	Crea un registro de consulta lento. Se establece en 1 para activar el registro de consultas lentas. El valor predeterminado es desactivado (0).

Para obtener más información, consulte Uso de auditorías avanzadas con un clúster de base de datos de Amazon Aurora MySQL.

Para Aurora PostgreSQL, la siguiente tabla muestra los registros que puede habilitar. Cuando se habilitan, se cargan automáticamente desde su clúster de base de datos de Aurora Serverless v1 a Amazon CloudWatch junto con los registros de errores regulares.

Registro de Aurora PostgreSQL	Descripción
`log_connections`	Activado de forma predeterminada y no se puede cambiar. Registra los detalles de todas las conexiones de cliente nuevas.
`log_disconnections`	Activado de forma predeterminada y no se puede cambiar. Registra todas las desconexi ones de cliente.

Registro de Aurora PostgreSQL	Descripción
`log_hostname`	Desactivado de forma predeterminada y no se puede cambiar. Los nombres de host no se registran.
`log_lock_waits`	El valor predeterminado es 0 (desactivado). Se establece en 1 para registrar las esperas de bloqueo.
`log_min_duration_statement`	La duración mínima (en milisegundos) para que una instrucción se ejecute antes de que se registre.
`log_min_messages`	Define los niveles de los mensajes que se registran. Los valores admitidos son `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Para registrar los datos de rendimiento en el registro de `postgres`, establezca el valor en `debug1`.
`log_temp_files`	Registra el uso de archivos temporales que superan los kilobytes especificados (kB).
`log_statement`	Controla las instrucciones SQL específicas que se registran. Los valores admitidos son `none`, `ddl`, `mod` y `all`. El valor predeterminado es `none`.

Después de activar los registros para Aurora MySQL o Aurora PostgreSQL para su clúster de base de datos de Aurora Serverless v1, puede ver los registros en CloudWatch.

Visualización registros de Aurora Serverless v1 con Amazon CloudWatch

Aurora Serverless v1 carga automáticamente ("publica") en Amazon CloudWatch todos los registros habilitados en el grupo de parámetros de clúster de bases de datos personalizado. No es necesario elegir o especificar los tipos de registro. La carga de registros se inicia tan pronto como se habilita el parámetro de configuración del registro. Si posteriormente deshabilita el parámetro de registro, las cargas futuras se detienen. Sin embargo, todos los registros que ya se hayan publicado en CloudWatch permanecerán hasta que los elimine.

Para obtener más información acerca de cómo usar CloudWatch con registros de Aurora MySQL, consulte Monitoreo de eventos de registro en Amazon CloudWatch.

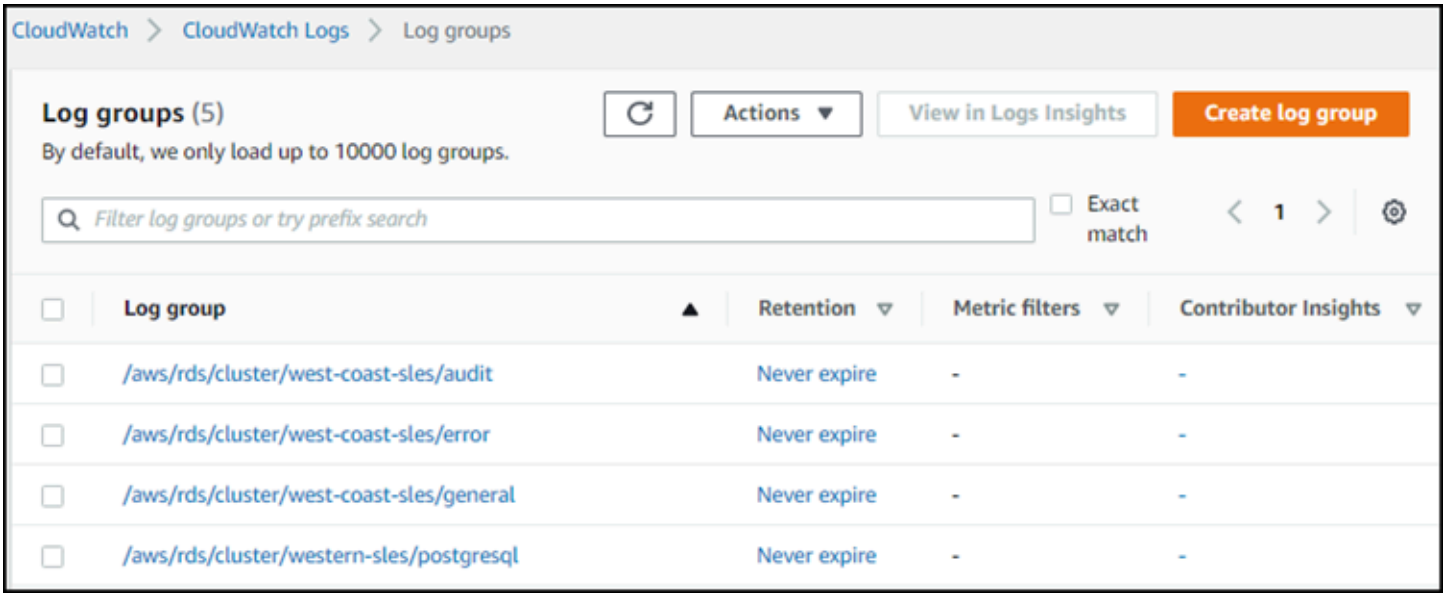
Para obtener más información sobre CloudWatch y Aurora PostgreSQL, consulte Publicación de registros de Aurora PostgreSQL en Amazon CloudWatch Logs.

Para ver los registros del clúster de bases de datos de Aurora Serverless v1

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija su Región de AWS.
3. Elija Log groups (Grupos de registros).
4. Seleccione el registro del clúster de bases de datos de Aurora Serverless v1 de la lista. Para los registros de errores, el patrón de nomenclatura es el siguiente.

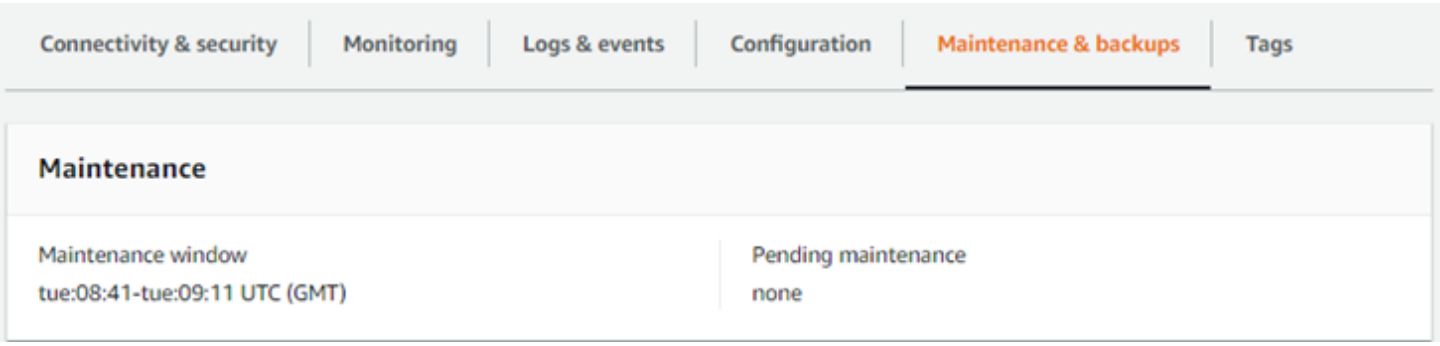
```
/aws/rds/cluster/cluster-name/error
```

Por ejemplo, en la siguiente captura de pantalla encontrará descripciones de registros publicados para un clúster de base de datos de Aurora Serverless v1 de Aurora PostgreSQL llamado `western-sles`. También puede encontrar varias descripciones del clúster de base de datos de Aurora Serverless v1 de Aurora MySQL, `west-coast-sles`. Elija el registro que le interesa para empezar a explorar su contenido.



Aurora Serverless v1 y mantenimiento

El mantenimiento del clúster de base de datos de Aurora Serverless v1, como la aplicación de las últimas características, correcciones y actualizaciones de seguridad, se realiza automáticamente. Aurora Serverless v1 tiene un periodo de mantenimiento que puede consultar en la Consola de administración de AWS, en Mantenimiento y copias de seguridad, para su clúster de base de datos de Aurora Serverless v1. Puede encontrar la fecha y la hora en que se puede realizar el mantenimiento y si hay algún mantenimiento pendiente para su clúster de bases de datos de Aurora Serverless v1, como se muestra en la siguiente figura.



Puede configurar el periodo de mantenimiento al crear el clúster de base de datos de Aurora Serverless v1 y modificarlo posteriormente. Para obtener más información, consulte Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos.

Los periodos de mantenimiento se utilizan para las actualizaciones programadas de las versiones principales. Las actualizaciones de versiones secundarias y las revisiones se aplican inmediatamente

durante el escalado. El escalado se realiza de acuerdo con la configuración establecida para `TimeoutAction`:

- `ForceApplyCapacityChange`: el cambio se aplica inmediatamente.
- `RollbackCapacityChange`: Aurora actualiza el clúster de manera forzada 3 días después del primer intento de aplicación de la revisión.

Al igual que con cualquier cambio forzado sin un punto de escalado apropiado, esto podría interrumpir la carga de trabajo.

Cuando es posible, Aurora Serverless v1 realiza el mantenimiento de una manera no disruptiva. Cuando se requiere mantenimiento, el clúster de bases de datos de Aurora Serverless v1 escala su capacidad para gestionar las operaciones necesarias. Antes de escalar, Aurora Serverless v1 busca un punto de escalado. Lo hace durante un máximo de tres días, si es necesario.

Al final de cada día que Aurora Serverless v1 no puede encontrar un punto de escalado, crea un evento de clúster. Este evento lo notifica sobre el mantenimiento pendiente y la necesidad de escalar para realizar el mantenimiento. La notificación incluye la fecha en la que Aurora Serverless v1 puede forzar al clúster de base de datos para que se escale.

Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.

Aurora Serverless v1 y conmutación por error

Si la instancia de base de datos de un clúster de base de datos de Aurora Serverless v1 deja de estar disponible o se produce un error en la zona de disponibilidad (AZ) en la que se encuentra, Aurora vuelve a crear la instancia de base de datos en una AZ diferente. Sin embargo, el clúster de Aurora Serverless v1 no es un clúster Multi-AZ. Esto se debe a que consta de una única instancia de base de datos en una sola AZ. Por tanto, el mecanismo de conmutación por error tarda más que en un clúster de Aurora con instancias aprovisionadas o de Aurora Serverless v2. El tiempo de conmutación por error de Aurora Serverless v1 está sin definir, ya que depende de la demanda y de la disponibilidad de capacidad de otras zonas de disponibilidad de la Región de AWS específica.

Debido a que Aurora separa la capacidad de computación y el almacenamiento, el volumen de almacenamiento para el clúster se distribuye por varias AZ. Sus datos permanecen disponibles si alguna interrupción afecta la instancia de base de datos o la AZ asociada.

Aurora Serverless v1 e instantáneas

El volumen de un clúster de Aurora Serverless v1 siempre está cifrado. Puede elegir la clave de cifrado, pero no puede desactivar el cifrado. Para copiar o compartir una instantánea de un clúster de Aurora Serverless v1, cifre la instantánea mediante su propia AWS KMS key. Para obtener más información, consulte Copia de una instantánea de clúster de base de datos. Para obtener más información sobre el cifrado y Amazon Aurora, consulte Cifrar un clúster de bases de datos de Amazon Aurora

Creación de un clúster de bases de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

El procedimiento siguiente crea un clúster de Aurora Serverless v1 sin ninguno de los objetos o datos de esquema. Si desea crear un clúster de Aurora Serverless v1 que es un duplicado de un aprovisionado de un clúster aprovisionado o de Aurora Serverless v1, puede realizar una operación de restauración o clonación de instantáneas. Para obtener más detalles, consulte Restauración de una instantánea de clúster de base de datos y Clonación de un volumen de clúster de base de datos de Amazon Aurora. No se puede convertir un clúster aprovisionado existente en Aurora Serverless v1. No se puede convertir un clúster de Aurora Serverless v1 existente en un clúster aprovisionado.

Cuando crea un clúster de bases de datos de Aurora Serverless v1, puede establecer la capacidad mínima y máxima del clúster. Cada unidad de capacidad equivale a una configuración de computación y memoria específicas. Aurora Serverless v1 crea automáticamente reglas de escalado para los límites de uso de la CPU, las conexiones y la memoria disponible, y se escala sin problemas al rango de unidades de capacidad que necesiten las aplicaciones. Para obtener más información, consulte Aurora Serverless v1 architecture.

Puede establecer los siguientes valores específicos para el clúster de bases de datos de Aurora Serverless v1:

- Unidad de capacidad mínima de Aurora: Aurora Serverless v1 puede reducir la capacidad hasta esta unidad de capacidad.
- Unidad de capacidad máxima de Aurora: Aurora Serverless v1 puede aumentar la capacidad hasta esta unidad de capacidad.

También puede elegir las siguientes opciones de configuración de escalado opcionales:

- Forzar el escalado de la capacidad a los valores especificados cuando se alcance el tiempo de espera: puede elegir esta configuración si desea que Aurora Serverless v1 fuerce el escalado de Aurora Serverless v1 incluso si no puede encontrar un punto de escalado antes de que se agote el tiempo de espera. Si desea que Aurora Serverless v1 cancele los cambios de capacidad si no puede encontrar un punto de escalado, no elija esta opción de configuración. Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.
- Pausar la capacidad de cómputo después de minutos consecutivos de inactividad: puede elegir esta configuración si desea que Aurora Serverless v1 escale a cero cuando no haya actividad en el clúster de bases de datos durante el periodo de tiempo que especifique. Con esta configuración habilitada, el clúster de bases de datos de Aurora Serverless v1 reanuda automáticamente el procesamiento y se escala a la capacidad necesaria para gestionar la carga de trabajo cuando se reanuda el tráfico de la base de datos. Para obtener más información, consulte Pausar y reanudar Aurora Serverless v1.

Antes de crear un clúster de bases de datos de Aurora Serverless v1, necesita una cuenta de AWS. También tiene que haber completado las tareas de configuración para trabajar con Amazon Aurora. Para obtener más información, consulte Configuración del entorno para Amazon Aurora. También debe completar otros pasos preliminares para crear cualquier clúster de bases de datos de Aurora. Para obtener más información, consulte Creación de un clúster de base de datos de Amazon Aurora.

Aurora Serverless v1 está disponible en determinadas Regiones de AWS y solo para versiones de Aurora MySQL y Aurora PostgreSQL específicas. Para obtener más información, consulte Aurora Serverless v1.

> **Note**
>
> El volumen de un clúster de Aurora Serverless v1 siempre está cifrado. Cuando crea el clúster de bases de datos de Aurora Serverless v1, no puede desactivar el cifrado, pero puede elegir usar su propia clave de cifrado. Con Aurora Serverless v2, puede elegir si desea cifrar el volumen del clúster.

Puede crear un clúster de base de datos de Aurora Serverless v1 con la AWS CLI o la API de RDS.

> **Note**
>
> Si recibe el siguiente mensaje de error al intentar crear el clúster, su cuenta necesita permisos adicionales.
> `Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.`
> Para obtener más información, consulte Uso de roles vinculados a servicios de Amazon Aurora.

No puede conectarse directamente a la instancia de base de datos en su clúster de bases de datos de Aurora Serverless v1. Para conectarse al clúster de bases de datos de Aurora Serverless v1, utilice el punto de enlace de la base de datos. Puede encontrar el punto de enlace del clúster de bases de datos de Aurora Serverless v1 en la pestaña Connectivity & security (Conectividad y seguridad) del clúster en la Consola de administración de AWS. Para obtener más información, consulte Conexión a un clúster de base de datos Amazon Aurora.

AWS CLI

Para crear un nuevo clúster de bases de datos de Aurora Serverless v1 mediante la AWS CLI, ejecute el comando create-db-cluster y especifique `serverless` en la opción `--engine-mode`.

Si lo desea, puede especificar la opción `--scaling-configuration` para configurar la capacidad mínima, la capacidad máxima y la pausa automática cuando no haya conexiones.

Los siguientes ejemplos de comandos crean un nuevo clúster de bases de datos Serverless estableciendo la opción `--engine-mode` en `serverless`. Los ejemplos también especifican valores para la opción `--scaling-configuration`.

Ejemplo de Aurora MySQL

El siguiente comando crea un nuevo clúster de base de datos sin servidor compatible con Aurora MySQL. Los valores de capacidad válidos para Aurora MySQL son 1, 2, 4, 8, 16, 32, 64, 128 y 256.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

Ejemplo de Aurora PostgreSQL

El siguiente comando crea un nuevo clúster de base de datos sin servidor compatible con PostgreSQL 13.9. Los valores de capacidad válidos para Aurora PostgreSQL son 2, 4, 8, 16, 32, 64, 192 y 384.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-postgresql --engine-version 13.9 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Para Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
```

```
    --engine aurora-postgresql --engine-version 13.9 ^
    --engine-mode serverless ^
    --scaling-configuration
MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

API de RDS

Para crear un nuevo clúster de bases de datos de Aurora Serverless v1 mediante la API de RDS, ejecute la operación CreateDBCluster y especifique `serverless` en el parámetro `EngineMode`.

Si lo desea, puede especificar el parámetro `ScalingConfiguration` para configurar la capacidad mínima, la capacidad máxima y la pausa automática cuando no haya conexiones. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 y 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 y 384.

Restauración de un clúster de bases de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

Puede configurar un clúster de bases de datos de Aurora Serverless v1 al restaurar la instantánea de un clúster de base de datos aprovisionado con la AWS CLI o la API de RDS.

Cuando restaure una instantánea en un clúster de bases de datos de Aurora Serverless v1, podrá establecer los siguientes valores específicos:

- Unidad de capacidad mínima de Aurora: Aurora Serverless v1 puede reducir la capacidad hasta esta unidad de capacidad.
- Unidad de capacidad máxima de Aurora: Aurora Serverless v1 puede aumentar la capacidad hasta esta unidad de capacidad.
- Acción de tiempo de espera: la acción que se debe realizar cuando se agota el tiempo de una modificación de capacidad porque no puede encontrar un punto de escalado. Aurora Serverless v1 El clúster de bases de datos puede forzar su clúster de bases de datos a la nueva configuración de capacidad si establece la opción Force scaling the capacity to the specified values... (Forzar el escalado de la capacidad a los valores especificados…). O bien, puede revertir el cambio de capacidad para cancelarlo si no elige la opción. Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.
- Pause after inactivity (Pausa tras inactividad): el tiempo sin tráfico de base de datos que ha de transcurrir para escalar hasta la capacidad de procesamiento cero. Cuando se reanude el tráfico de la base de datos, Aurora reanudará automáticamente la capacidad de procesamiento y se escalará para controlar el tráfico.

Para obtener información general acerca de cómo restaurar un clúster de bases de datos a partir de una instantánea, consulte Restauración de una instantánea de clúster de base de datos.

AWS CLI

Puede configurar un clúster de bases de datos de Aurora Serverless al restaurar la instantánea de un clúster de bases de datos aprovisionado mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Cuando restaure una instantánea en un clúster de bases de datos de Aurora Serverless, podrá establecer los siguientes valores específicos:

- Unidad de capacidad mínima de Aurora: Aurora Serverless puede reducir la capacidad hasta esta unidad de capacidad.
- Unidad de capacidad máxima de Aurora: Aurora Serverless puede aumentar la capacidad hasta esta unidad de capacidad.
- Acción de tiempo de espera: la acción que se debe realizar cuando se agota el tiempo de una modificación de capacidad porque no puede encontrar un punto de escalado. Aurora Serverless v1 El clúster de bases de datos puede forzar su clúster de bases de datos a la nueva configuración de capacidad si establece la opción Force scaling the capacity to the specified values... (Forzar el escalado de la capacidad a los valores especificados…). O bien, puede revertir el cambio de

capacidad para cancelarlo si no elige la opción. Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.

- Pause after inactivity (Pausa tras inactividad): el tiempo sin tráfico de base de datos que ha de transcurrir para escalar hasta la capacidad de procesamiento cero. Cuando se reanude el tráfico de la base de datos, Aurora reanudará automáticamente la capacidad de procesamiento y se escalará para controlar el tráfico.

> **Note**
>
> La versión de la instantánea del clúster de base de datos debe ser compatible con Aurora Serverless v1. Para ver una lista de las versiones admitidas, consulte Aurora Serverless v1.

Para restaurar una instantánea en un clúster de Aurora Serverless v1 con compatibilidad con MySQL 5.7, incluya los siguientes parámetros adicionales:

- `--engine` **`aurora-mysql`**
- `--engine-version` **`5.7`**

Los parámetros `--engine` y `--engine-version` le permiten crear un clúster de Aurora Serverless v1 compatible con MySQL 5.7 a partir de una instantánea de Aurora Serverless v1 o Aurora compatible con MySQL 5.6. En el siguiente ejemplo, se restaura una instantánea de un clúster compatible con MySQL 5.6 denominado *`mydbclustersnapshot`* en un clúster de Aurora Serverless v1 compatible con MySQL 5.7 denominado *`mynewdbcluster`*.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless \
    --engine aurora-mysql \
    --engine-version 5.7
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
```

```
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql ^
    --engine-version 5.7
```

Si lo desea, puede especificar la opción `--scaling-configuration` para configurar la capacidad mínima, la capacidad máxima y la pausa automática cuando no haya conexiones. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 y 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 y 384.

En el ejemplo siguiente, se restaura desde una instantánea de clúster de bases de datos creada anteriormente denominada *mydbclustersnapshot* a un nuevo clúster de bases de datos denominado *mynewdbcluster*. Defina el `--scaling-configuration` para que el nuevo clúster de bases de datos de Aurora Serverless v1 pueda escalar de 8 ACU a 64 ACU (unidades de capacidad Aurora) según sea necesario para procesar la carga de trabajo. Una vez finalizado el procesamiento y después de 1000 segundos sin conexiones que admitir, el clúster se cierra hasta que las solicitudes de conexión soliciten que se reinicie.

Para Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

Para Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

API de RDS

Para configurar un clúster de bases de datos de Aurora Serverless v1 cuando realiza una restauración a partir de un clúster de bases de datos mediante la API de RDS, ejecute la operación RestoreDBClusterFromSnapshot y especifique `serverless` en el parámetro `EngineMode`.

Si lo desea, puede especificar el parámetro `ScalingConfiguration` para configurar la capacidad mínima, la capacidad máxima y la pausa automática cuando no haya conexiones. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: `1`, `2`, `4`, `8`, `16`, `32`, `64`, `128` y `256`.
- Aurora PostgreSQL: `2`, `4`, `8`, `16`, `32`, `64`, `192` y `384`.

Modificación de un clúster de bases de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

Después de configurar un clúster de base de datos de Aurora Serverless v1, puede modificar ciertas propiedades con la Consola de administración de AWS, la AWS CLI o la API de RDS. La mayoría de las propiedades que puede modificar son las mismas que para otros tipos de clústeres de Aurora.

A continuación, se indican los cambios más importantes de Aurora Serverless v1:

- Modificación de la configuración de escalado
- Actualización de la versión principal
- Conversión de Aurora Serverless v1 a aprovisionada

Modificación de la configuración de escalado de un clúster de base de datos de Aurora Serverless v1

Puede establecer la capacidad mínima y máxima del clúster de bases de datos. Cada unidad de capacidad equivale a una configuración de computación y memoria específicas. Aurora Serverless crea automáticamente reglas de escalado para los límites de uso de la CPU, las conexiones y la memoria disponible. También puede establecer si Aurora Serverless debe pausar la base de datos cuando no haya actividad y reanudarla cuando vuelva a haberla.

Puede establecer los siguientes valores específicos para la configuración de escalado:

- Unidad de capacidad mínima de Aurora: Aurora Serverless puede reducir la capacidad hasta esta unidad de capacidad.
- Unidad de capacidad máxima de Aurora: Aurora Serverless puede aumentar la capacidad hasta esta unidad de capacidad.
- Autoscaling timeout and action (Acción y tiempo de espera de escalado automático): esta sección especifica cuánto tiempo espera Aurora Serverless para buscar un punto de escalado antes de que se agote el tiempo de espera. También especifica la acción que se debe realizar cuando se agota el tiempo de una modificación de capacidad porque no se puede encontrar ningún punto de escalado. Aurora puede forzar el cambio de capacidad para establecer la capacidad en el valor especificado lo antes posible. También puede revertir el cambio de capacidad para cancelarla. Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.
- Pausa después de la inactividad: utilice la opción Escalar la capacidad a 0 ACU cuando el clúster esté inactivo para escalar la base de datos a una capacidad de procesamiento de cero mientras esté inactiva. Cuando se reanude el tráfico de la base de datos, Aurora reanudará automáticamente la capacidad de procesamiento y se escalará para controlar el tráfico.

> **Note**
>
> Al modificar el rango de capacidad de un clúster de base de datos Aurora Serverless, el cambio se produce inmediatamente, independientemente de si decide aplicarlo de inmediato o durante el siguiente período de mantenimiento programado.

Consola

Puede modificar la configuración de escalado de un clúster de bases de datos de Aurora mediante la Consola de administración de AWS.

Para modificar un clúster de bases de datos de Aurora Serverless v1

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster de bases de datos de Aurora Serverless v1 que desea modificar.
4. Para Actions (Acciones), elija Modify cluster (Modificar clúster).
5. En la sección Capacity settings (Configuración de capacidad), modifique la configuración de escalado.
6. Elija Continuar.
7. En la página Modificar el clúster de base de datos, revise las modificaciones y elija cuándo aplicarlas.
8. Elija Modificar clúster.

AWS CLI

Para modificar la configuración de escalado de un clúster de bases de datos de Aurora Serverless v1 mediante la AWS CLI, ejecute el comando modify-db-cluster de la AWS CLI. Especifique la opción `--scaling-configuration` para configurar la capacidad mínima, la capacidad máxima y la pausa automática cuando no haya conexiones. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 y 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 y 384.

En este ejemplo, se modifica la configuración de escalado de un clúster de bases de datos de Aurora Serverless v1 denominado *`sample-cluster`*.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
```

```
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

API de RDS

Puede modificar la configuración de escalado de un clúster de bases de datos de Aurora mediante la operación ModifyDBCluster de la API. Especifique el parámetro `ScalingConfiguration` para configurar la capacidad mínima, la capacidad máxima y la pausa automática cuando no haya conexiones. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 y 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 y 384.

Actualización de la versión principal de un clúster de base de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

Puede actualizar la versión principal de un clúster de base de datos de Aurora Serverless v1 compatible con PostgreSQL 11 a la versión correspondiente compatible con PostgreSQL 13.

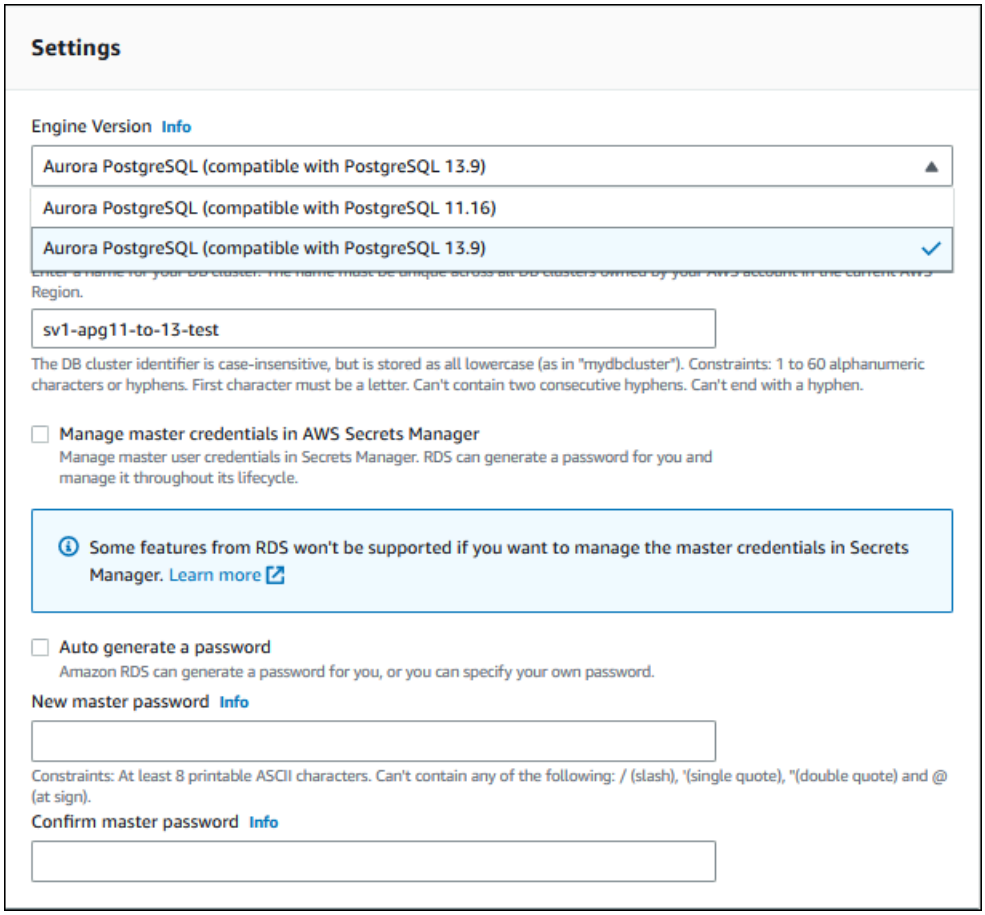
Consola

Realice una actualización in situ del clúster de base de datos de Aurora Serverless v1 utilizando la Consola de administración de AWS.

Para actualizar un clúster de base de datos de Aurora Serverless v1

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).
3. Elija el clúster de base de datos de Aurora Serverless v1 que desea actualizar.
4. Para Actions (Acciones), elija Modify cluster (Modificar clúster).
5. Para la versión, elija un número de versión de Aurora PostgreSQL versión 13.

   El siguiente ejemplo muestra una actualización in situ de Aurora PostgreSQL 11.16 a 13.9.



   Si realiza una actualización de la versión principal, deje todas las demás propiedades igual. Para cambiar cualquiera de las demás propiedades, realice otra operación Modificar una vez finalizada la actualización.

6. Elija Continuar.
7. En la página Modificar el clúster de base de datos, revise las modificaciones y elija cuándo aplicarlas.
8. Elija Modificar clúster.

AWS CLI

Para realizar una actualización in situ de un clúster de base de datos de Aurora Serverless v1 compatible con PostgreSQL 11 a uno compatible con PostgreSQL 13, especifique el parámetro `--engine-version` con un número de versión de Aurora MySQL versión 13 compatible con Aurora Serverless v1. Incluya también el parámetro `--allow-major-version-upgrade`.

En este ejemplo, modifique la versión principal de un clúster de base de datos de Aurora Serverless v1 compatible con PostgreSQL 11 llamado `sample-cluster`. Al hacerlo, se realiza una actualización in situ a un clúster de base de datos de Aurora Serverless v1 compatible con PostgreSQL 13.

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --engine-version 13.serverless_12 \
    --allow-major-version-upgrade
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --engine-version 13.serverless_12 ^
    --allow-major-version-upgrade
```

API de RDS

Para realizar una actualización in situ de un clúster de base de datos de Aurora Serverless v1 compatible con PostgreSQL 11 a uno compatible con PostgreSQL 13, especifique el parámetro `EngineVersion` con un número de versión de Aurora MySQL versión 13 compatible con Aurora Serverless v1. Incluya también el parámetro `AllowMajorVersionUpgrade`.

Conversión de un clúster de base de datos de Aurora Serverless v1 a un clúster aprovisionado

Puede convertir un clúster de base de datos de Aurora Serverless v1 en un clúster de base de datos aprovisionado. Para realizar la conversión, use la CLI de AWS o la API de Amazon RDS para cambiar la clase de instancia de base de datos a Aprovisionada. Siga los pasos que se indican a continuación para modificar la clase de instancia de base de datos.

El siguiente ejemplo muestra cómo utilizar la CLI de AWS para convertir un clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado.

AWS CLI

Para convertir un clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado, ejecute el comando de la AWS CLI modify-db-cluster.

Se requieren los siguientes parámetros:

- `--db-cluster-identifier`: el clúster de base de datos de Aurora Serverless v1 que está convirtiendo en aprovisionado.
- `--engine-mode`: use el valor `provisioned`.
- `--allow-engine-mode-change`
- `--db-cluster-instance-class`: elija la clase de instancia de base de datos para el clúster de base de datos aprovisionado en función de la capacidad del clúster de base de datos de Aurora Serverless v1.

En este ejemplo, se convierte un clúster de base de datos de Aurora Serverless v1 denominado `sample-cluster` y se utiliza la clase de instancia de base de datos de `db.r5.xlarge`.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier sample-cluster \
--engine-mode provisioned \
--allow-engine-mode-change \
--db-cluster-instance-class db.r5.xlarge
```

Para Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--engine-mode provisioned ^
--allow-engine-mode-change ^
--db-cluster-instance-class db.r5.xlarge
```

El siguiente ejemplo muestra cómo utilizar la API de Amazon RDS para convertir un clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado.

API de RDS

Para convertir un clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado, utilice la operación de la API ModifyDBCluster.

Se requieren los siguientes parámetros:

- `DBClusterIdentifier`: el clúster de base de datos de Aurora Serverless v1 que está convirtiendo en aprovisionado.
- `EngineMode`: use el valor `provisioned`.
- `AllowEngineModeChange`
- `DBClusterInstanceClass`: elija la clase de instancia de base de datos para el clúster de base de datos aprovisionado en función de la capacidad del clúster de base de datos de Aurora Serverless v1.

Consideraciones a la hora de convertir un clúster de base de datos de Aurora Serverless v1 en un clúster aprovisionado

Las siguientes consideraciones se aplican cuando un clúster de base de datos de Aurora Serverless v1 se convierte en un clúster aprovisionado:

- Puede utilizar esta conversión como parte de la actualización de su clúster de base de datos de Aurora Serverless v1 a Aurora Serverless v2. Para obtener más información, consulte Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.
- El proceso de conversión crea una instancia de base de datos de lector en el clúster de base de datos, la promociona a una instancia de escritor y, a continuación, elimina la instancia de Aurora Serverless v1 original. Cuando convierta el clúster de base de datos, no podrá realizar ninguna otra modificación al mismo tiempo, como cambiar la versión del motor de base de datos o el

grupo de parámetros del clúster de base de datos. La operación de conversión se aplica de forma inmediata y no se puede deshacer.

- Durante la conversión, se realiza una instantánea del clúster de base de datos de copia de seguridad del clúster de base de datos por si se produce un error. El identificador de la instantánea del clúster de base de datos tiene el formato `pre-modify-engine-mode-`*`DB_cluster_identifier`*`-`*`timestamp`*.
- Aurora usa la versión actual secundaria predeterminada del motor de base de datos para el clúster de base de datos aprovisionado.
- Si no proporciona una clase de instancia de base de datos para el clúster de base de datos convertido, Aurora recomienda una en función de la capacidad máxima del clúster de base de datos de Aurora Serverless v1 original. La capacidad recomendada para las asignaciones de clases de instancia se muestra en la siguiente tabla.

Capacidad máxima (ACU) de Serverless	Clase de instancia de base de datos aprovisionada
1	db.t3.small
2	db.t3.medium
4	db.t3.large
8	db.r5.large
16	db.r5.xlarge
32	db.r5.2xlarge
64	db.r5.4xlarge
128	db.r5.8xlarge
192	db.r5.12xlarge
256	db.r5.16xlarge
384	db.r5.24xlarge

> **Note**
>
> Según la clase de instancia de base de datos que elija y el uso de la base de datos, es posible que vea diferentes costes para un clúster de base de datos aprovisionado en comparación con Aurora Serverless v1.
> Si convierte su clúster de base de datos de Aurora Serverless v1 en una clase de instancia de base de datos ampliable (db.t*), podría incurrir en costes adicionales por el uso del clúster de base de datos. Para obtener más información, consulte Tipos de clase de instancia de base de datos.

Escalado manual de la capacidad del clúster de bases de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

Por lo general, los clústeres de base de datos de Aurora Serverless v1 escalan sin problemas en función de la carga de trabajo. Sin embargo, es posible que la capacidad no siempre escale lo suficientemente rápido como para cumplir con los extremos repentinos, como un aumento exponencial de las transacciones. En tales casos, puede iniciar la operación de escalado de forma manual estableciendo un nuevo valor de capacidad. Después de establecer la capacidad de forma explícita, Aurora Serverless v1 podrá escalar automáticamente el clúster de bases de datos. Realiza la acción según el periodo de recuperación del escalado descendente.

Puede establecer de manera explícita la capacidad de un clúster de bases de datos de Aurora Serverless v1 en un valor específico mediante la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Puede establecer la capacidad de un clúster de bases de datos Aurora mediante la Consola de administración de AWS.

Para modificar un clúster de bases de datos de Aurora Serverless v1

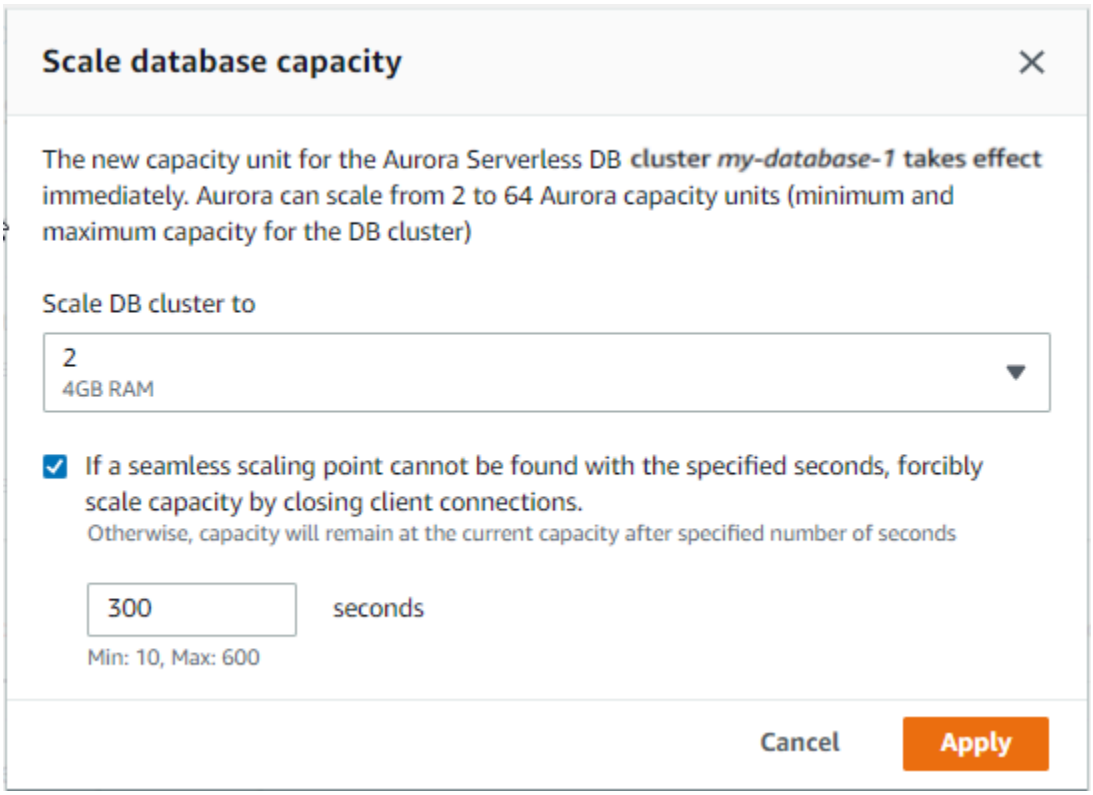
1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el clúster de bases de datos de Aurora Serverless v1 que desea modificar.
4. En Actions (Acciones), elija Set capacity (Establecer capacidad).
5. En la ventana Scale database capacity (Escalar la capacidad de la base de datos), elija lo siguiente:

    a. Para el selector desplegable Scale DB cluster to (Escalar clúster de bases de datos a), elija la nueva capacidad que desee para el clúster de bases de datos.

    b. Para la casilla de verificación If a seamless scaling point cannot be found… (Si no se puede encontrar un punto de escalado constante…), elija el comportamiento que desee para la configuración de `TimeoutAction` de su clúster de base de datos de Aurora Serverless v1, de la siguiente manera:

    - Quite la marca de esta opción si desea que su capacidad permanezca sin cambios si Aurora Serverless v1 no encuentra un punto de escalado antes de agotar el tiempo de espera.
    - Seleccione esta opción si desea forzar a su clúster de bases de datos de Aurora Serverless v1 a cambiar su capacidad incluso si no puede encontrar un punto de escalado antes de que se agote el tiempo de espera. Esta opción puede resultar en la caída de las conexiones de Aurora Serverless v1 que le impiden encontrar un punto de escalado.

    c. En seconds (segundos), introduzca la cantidad de tiempo que desea permitir que el clúster de bases de datos de Aurora Serverless v1 busque un punto de escalado antes de agotar el tiempo de espera. Puede especificar entre 10 segundos y 600 segundos (10 minutos). El valor predeterminado es de cinco minutos (300 segundos). El ejemplo siguiente obliga al clúster de bases de datos de Aurora Serverless v1 a reducir a 2 ACU, incluso si no puede encontrar un punto de escalado en cinco minutos.



6. Seleccione Apply.

Para obtener más información acerca de los puntos de escalado, la `TimeoutAction` y los periodos de recuperación, consulte Escalado automático para Aurora Serverless v1.

AWS CLI

Para establecer la capacidad de un clúster de bases de datos de Aurora Serverless v1 mediante la AWS CLI, ejecute el comando modify-current-db-cluster-capacity de la AWS CLI y especifique la opción `--capacity`. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 y 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 y 384.

En este ejemplo, se establece la capacidad de un clúster de bases de datos de Aurora Serverless v1 denominado *`sample-cluster`* en *`64`*.

```
aws rds modify-current-db-cluster-capacity --db-cluster-identifier sample-cluster --
capacity 64
```

API de RDS

Puede establecer la capacidad de un clúster de bases de datos de Aurora mediante la operación ModifyCurrentDBClusterCapacity de la API. Especifique el parámetro `Capacity`. Entre los valores de capacidad válidos se incluyen los siguientes:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 y 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 y 384.

Visualización de los clústeres de base de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

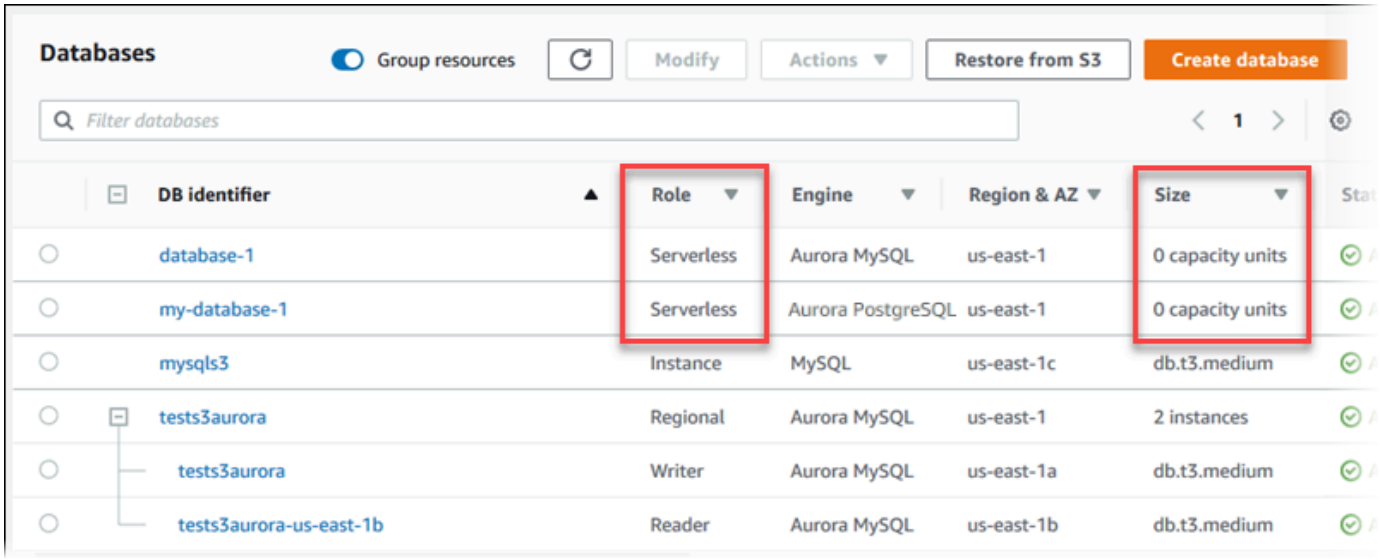
Después de crear uno o más clústeres de base de datos de Aurora Serverless v1, puede consultar cuáles de ellos son de tipo Serverless (Sin servidor) y cuáles son de tipo Instance (Instancia). También puede ver el número actual de unidades de capacidad de Aurora (ACU) que cada clúster de bases de datos de Aurora Serverless v1 utiliza. Cada ACU es una combinación de capacidad de procesamiento (CPU) y de memoria (RAM).

Para visualizar los clústeres de base de datos de Aurora Serverless v1

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

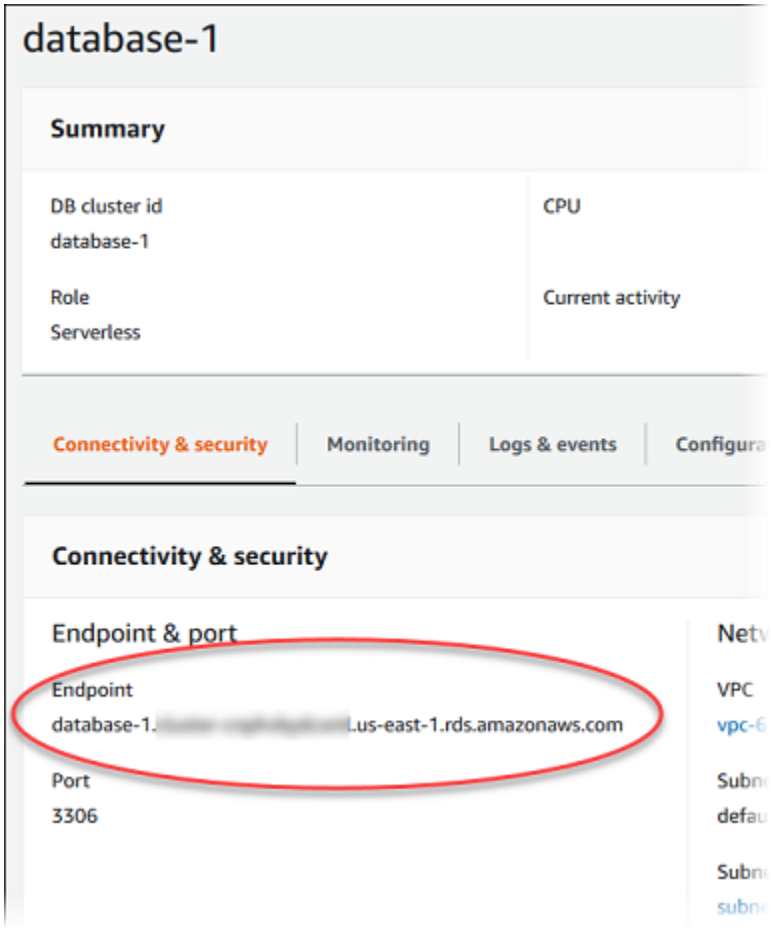
2. En la esquina superior derecha de la Consola de administración de AWS, elija la Región de AWS en la que creó los clústeres de base de datos de Aurora Serverless v1.
3. En el panel de navegación, seleccione Databases (Bases de datos).

   Puede ver el tipo de cada clúster de bases de datos en el campo Role (Rol). Los clústeres de base de datos de Aurora Serverless v1 muestran el tipo Serverless (Sin servidor). Puede consultar la capacidad actual de un clúster de bases de datos de Aurora Serverless v1 en Size (Tamaño).

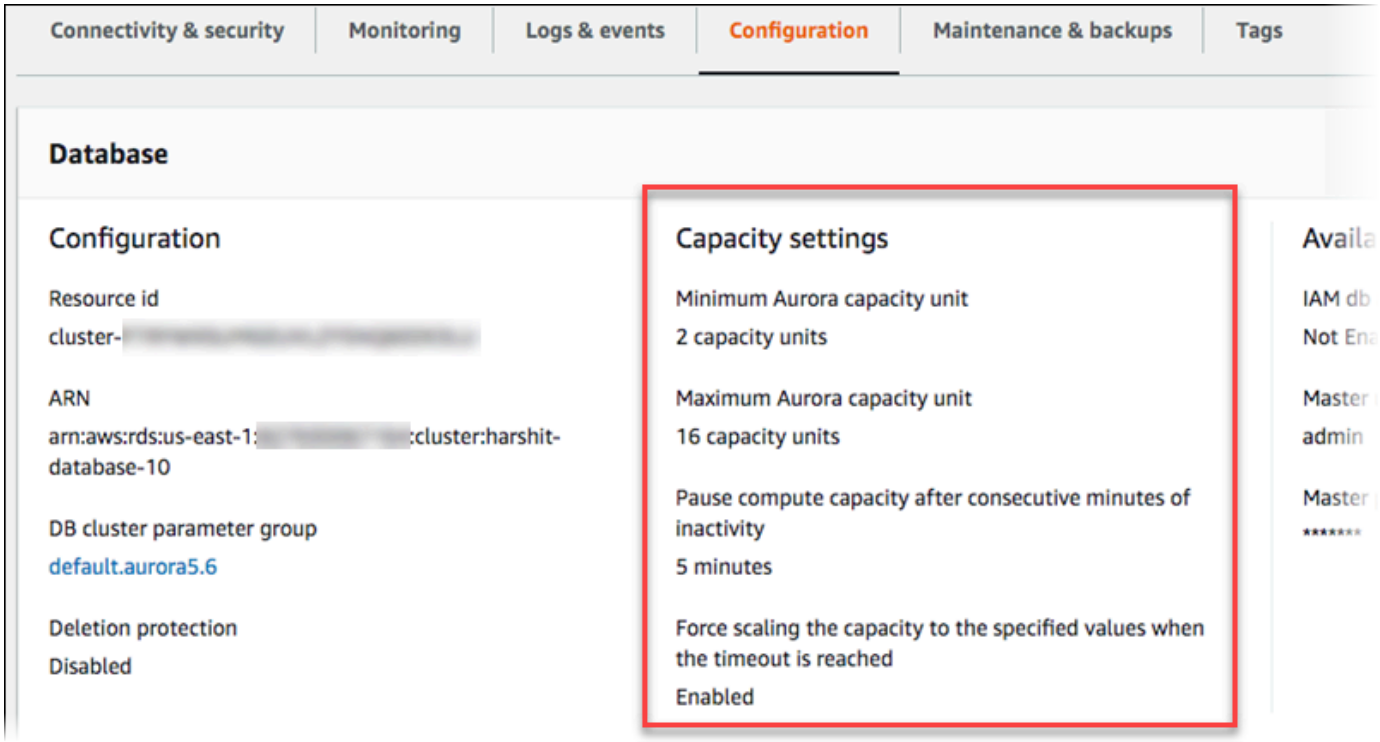


4. Elija el nombre de un clúster de bases de datos de Aurora Serverless v1 para mostrar sus detalles.

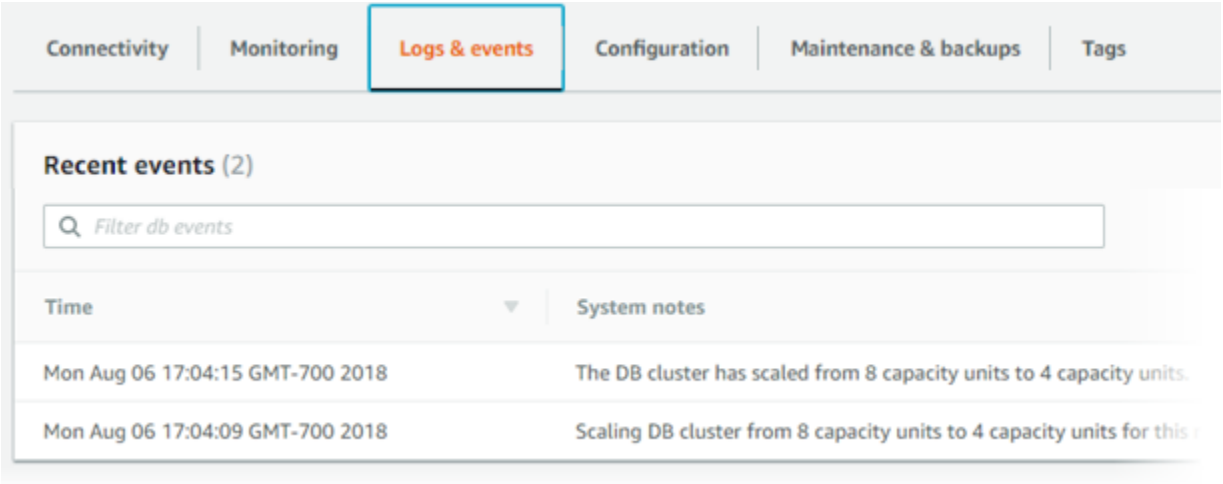
   En la pestaña Connectivity & security (Conectividad y seguridad), tenga en cuenta el punto de enlace de la base de datos. Utilice este punto de enlace para conectarse al clúster de bases de datos de Aurora Serverless v1.



Elija la pestaña Configuration (Configuración) para ver la configuración de la capacidad.



Se genera un evento de escalado cada vez que un clúster de bases de datos se escala para ampliarlo o para reducirlo, se pone en pausa o se reanuda. Elija la pestaña Logs & events (Registros y eventos) para ver los eventos recientes. En la imagen siguiente se muestran ejemplos de estos eventos.



Monitoreo de los eventos de capacidad y escalado del clúster de bases de datos de Aurora Serverless v1

Puede ver el clúster de bases de datos de Aurora Serverless v1 en CloudWatch para monitorear la capacidad asignada al clúster de bases de datos con la métrica `ServerlessDatabaseCapacity`. También puede monitorizar todas las métricas estándar de CloudWatch para Aurora, como `CPUUtilization`, `DatabaseConnections`, `Queries`, etc.

Puede hacer que Aurora publica algunos registros de base de datos o todos en CloudWatch. Seleccione los registros que publicará al habilitar los parámetros de configuración como `general_log` y `slow_query_log` en el grupo de parámetros de clúster de bases de datos asociado al clúster de Aurora Serverless v1. A diferencia de los clústeres aprovisionados, los clústeres de Aurora Serverless v1 no requieren que especifique en la configuración del clúster de bases de datos qué tipos de registro se van a cargar en CloudWatch. Los clústeres de Aurora Serverless v1 cargan automáticamente todos los registros disponibles. Cuando se deshabilita un parámetro de configuración de registro, la publicación del registro en CloudWatch se detiene. También puede eliminar los registros en CloudWatch si ya no son necesarios.

Para obtener una introducción sobre Amazon CloudWatch para su clúster de bases de datos de Aurora Serverless v1, consulte Visualización registros de Aurora Serverless v1 con Amazon CloudWatch. Para obtener más información acerca de cómo monitorear los clústeres de base de datos de Aurora a través de CloudWatch, consulte Monitoreo de eventos de registro en Amazon CloudWatch.

Para conectarse a un clúster de bases de datos de Aurora Serverless v1, utilice el punto de enlace de base de datos. Para obtener más información, consulte Conexión a un clúster de base de datos Amazon Aurora.

> **Note**
>
> No es posible conectarse directamente a instancias de base de datos específicas de los clústeres de base de datos de Aurora Serverless v1.

Eliminación de un clúster de bases de datos de Aurora Serverless v1

> **Important**
>
> AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

En función de cómo cree un clúster de base de datos de Aurora Serverless v1, es posible que se active de forma predeterminada la protección contra la eliminación. No puede eliminar de inmediato un clúster de base de datos de Aurora Serverless v1 que tenga habilitada la Protección contra eliminación. Para eliminar clústeres de base de datos de Aurora Serverless v1 que tengan protección contra eliminación mediante la Consola de administración de AWS, primero modifique el clúster para quitar esta protección. Para obtener información acerca del uso de la AWS CLI para esta tarea, consulte AWS CLI.

Para deshabilitar la protección contra eliminación con la Consola de administración de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija DB Clusters (Clústeres de base de datos).
3. Elija su clúster de bases de datos de Aurora Serverless v1 de la lista.
4. Elija Modify (Modificar) para abrir la configuración del clúster de bases de datos. La página “Modify DB cluster (Modificar clúster de base de datos)” abre los parámetros, la configuración de capacidad y otros detalles de configuración del clúster de base de datos de Aurora Serverless v1. La protección contra eliminación se encuentra en la sección Additional configuration (Configuración adicional).
5. Desmarque la casilla Enable deletion protection (Habilitar la protección contra eliminación) en la tarjeta de propiedades Additional configuration (Configuración adicional).

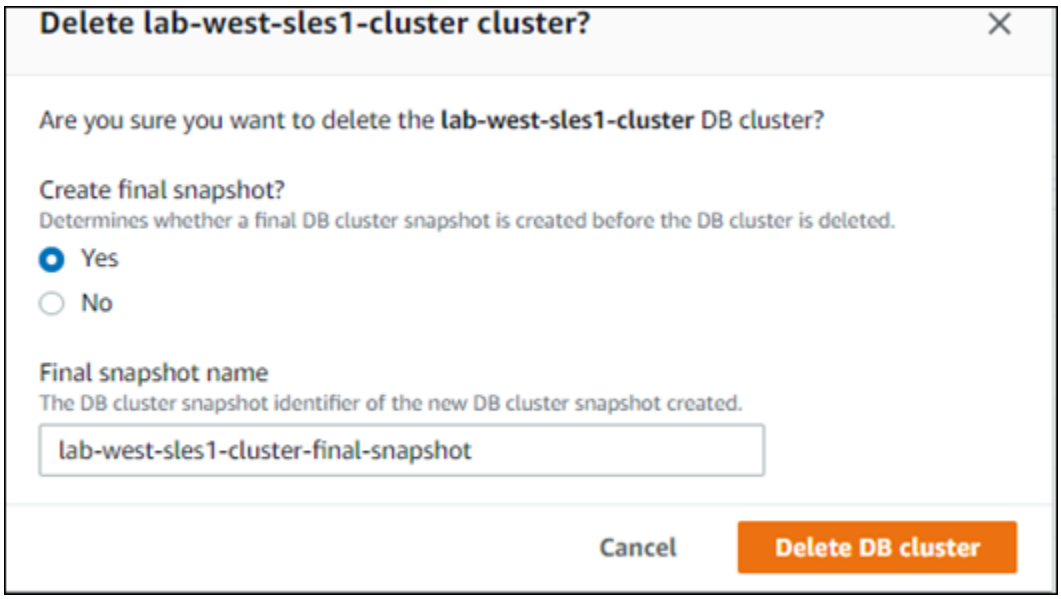
6. Elija Continue (Continuar). Aparecerá el Summary of modifications (Resumen de modificaciones).
7. Elija Modify cluster (Modificar clúster) para aceptar el resumen de las modificaciones. También puede elegir Back (Atrás) para modificar los cambios o Cancel (Cancelar) para descartar los cambios.

Una vez que la protección contra eliminación ya no esté activa, podrá eliminar el clúster de bases de datos de Aurora Serverless v1 mediante la Consola de administración de AWS.

Consola

Para eliminar un clúster de bases de datos de Aurora Serverless v1

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la sección Resources (Recursos), elija DB Clusters (Clústeres de base de datos).
3. Elija el clúster de bases de datos de Aurora Serverless v1 que desea eliminar.
4. En Actions (Acciones), elija Delete (Eliminar). Se le pedirá que confirme que desea eliminar el clúster de bases de datos de Aurora Serverless v1.
5. Le recomendamos que mantenga las opciones preseleccionadas:
   - En Create final snapshot? (¿Crear instantánea final?), elija Yes (Sí)
   - En Final snapshot name (Nombre de la instantánea final), escriba el nombre del clúster de bases de datos de Aurora Serverless v1 más `-final-snapshot`. Sin embargo, puede cambiar el nombre de la instantánea final en este campo.



Si elige No para Create final snapshot? (¿Crear instantánea final?), no podrá restaurar el clúster de bases de datos mediante instantáneas ni restauración puntual en el tiempo.

6. Seleccione Delete DB cluster (Eliminar clúster de bases de datos).

Aurora Serverless v1 elimina el clúster de bases de datos. Si elige tener una instantánea final, verá que el estado de su clúster de bases de datos de Aurora Serverless v1 cambia a “Backing-up (Efectuando copia de seguridad)” antes de eliminarse y no aparecer más en la lista.

AWS CLI

Antes de comenzar, configure su AWS CLI con la ID de clave de acceso de AWS, la clave de acceso secreta de AWS y la Región de AWS donde se ubique su clúster de bases de datos de Aurora Serverless v1. Para obtener más información, consulte Configuración básica en la Guía del usuario de AWS Command Line Interface.

No puede eliminar un clúster de bases de datos de Aurora Serverless v1 hasta después de desactivar la protección contra eliminación para los clústeres configurados con esta opción. Si intenta eliminar un clúster que tiene activada esta opción de protección, verá el siguiente mensaje de error.

```
An error occurred (InvalidParameterCombination) when calling the DeleteDBCluster
  operation: Cannot delete protected Cluster, please disable deletion protection and
 try again.
```

Puede cambiar la configuración de protección contra eliminación de su clúster de bases de datos de Aurora Serverless v1 mediante el comando modify-db-cluster de la AWS CLI, como se muestra a continuación:

```
aws rds modify-db-cluster --db-cluster-identifier your-cluster-name --no-deletion-
protection
```

Este comando devuelve las propiedades revisadas para el clúster de bases de datos especificado. Ahora puede eliminar su clúster de bases de datos de Aurora Serverless v1.

Se recomienda crear siempre una instantánea final cada vez que elimine un clúster de bases de datos de Aurora Serverless v1. El siguiente ejemplo de uso del comando delete-db-cluster de la AWS CLI muestra cómo hacerlo. Proporcione el nombre del clúster de bases de datos y un nombre para la instantánea.

Para Linux, macOS o Unix:

```
aws rds delete-db-cluster --db-cluster-identifier \
  your-cluster-name --no-skip-final-snapshot \
  --final-db-snapshot-identifier name-your-snapshot
```

Para Windows:

```
aws rds delete-db-cluster --db-cluster-identifier ^
  your-cluster-name --no-skip-final-snapshot ^
  --final-db-snapshot-identifier name-your-snapshot
```

Aurora Serverless v1 y versiones del motor de base de datos de Aurora

Important

AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1, que será el 31 de marzo de 2025. Todos los clústeres de Aurora Serverless v1 que no se migren antes del 31 de marzo de 2025 se migrarán a Aurora Serverless v2 durante el periodo de mantenimiento. Si se produce un error en la actualización, Amazon Aurora convierte el clúster sin servidor v1 en un clúster aprovisionado con la versión de motor equivalente

durante el periodo de mantenimiento. Si procede, Amazon Aurora inscribirá el clúster aprovisionado convertido en soporte extendido de Amazon RDS. Para obtener más información, consulte Soporte extendido de RDS.

Aurora Serverless v1 está disponible en determinadas Regiones de AWS y solo para versiones de Aurora MySQL y Aurora PostgreSQL específicas. Para ver la lista actual de Regiones de AWS que admiten Aurora Serverless v1 y las versiones específicas de Aurora MySQL y Aurora PostgreSQL disponibles en cada región, consulte Aurora Serverless v1.

Aurora Serverless v1 utiliza su motor de base de datos de Aurora asociado para identificar las versiones compatibles específicas para cada motor de base de datos admitido, de la siguiente manera:

- Aurora MySQL Serverless
- Aurora PostgreSQL Serverless

Cuando las versiones secundarias de los motores de base de datos se vuelven disponibles para Aurora Serverless v1, se aplican automáticamente en las distintas Regiones de AWS donde Aurora Serverless v1 está disponible. En otras palabras, no necesita actualizar el clúster de bases de datos de Aurora Serverless v1 para obtener una nueva versión secundaria del motor de base de datos del clúster cuando está disponible para Aurora Serverless v1.

Aurora MySQL Serverless

Aurora Serverless v1 está disponible en determinadas Regiones de AWS y solo para versiones de Aurora MySQL específicas. Para ver la lista actual de Regiones de AWS que admiten Aurora Serverless v1 y las versiones específicas de Aurora MySQL disponibles en cada región, consulte Aurora Serverless v1 con Aurora MySQL.

Para obtener información acerca de las mejoras y las correcciones de Aurora MySQL versión 2, consulte Database engine updates for Amazon Aurora MySQL versión 2 (Actualizaciones del motor de base de datos de Amazon Aurora MySQL versión 2) en las Notas de la versión de Aurora MySQL.

Para emplear una versión más reciente de Aurora MySQL, puede usar Aurora Serverless v2. Para ver las Regiones de AWS y las versiones de Aurora MySQL que puede utilizar con Aurora Serverless v2, consulte Aurora Serverless v2 con Aurora MySQL. Para obtener información de uso sobre Aurora Serverless v2, consulte Uso de Aurora Serverless v2.

Aurora PostgreSQL Serverless

Aurora Serverless v1 está disponible en determinadas Regiones de AWS y solo para versiones de Aurora PostgreSQL específicas. Para ver la lista actual de Regiones de AWS que admiten Aurora Serverless v1 y las versiones específicas de Aurora PostgreSQL disponibles en cada región, consulte Aurora Serverless v1 con Aurora PostgreSQL.

Si quiere usar Aurora PostgreSQL para su clúster de base de datos de Aurora Serverless v1, puede elegir las versiones compatibles con Aurora PostgreSQL 13. Las versiones secundarias para Edición compatible con Aurora PostgreSQL incluyen solo los cambios que son compatibles con versiones anteriores. Su clúster de base de datos de Aurora Serverless v1 se actualiza de forma transparente cuando una versión secundaria de Aurora PostgreSQL queda disponible para Aurora Serverless v1 en su Región de AWS.

Para emplear una versión más reciente de Aurora PostgreSQL, puede usar Aurora Serverless v2. Para ver las Regiones de AWS y las versiones de Aurora PostgreSQL que puede utilizar con Aurora Serverless v2, consulte Aurora Serverless v2 con Aurora PostgreSQL. Para obtener información de uso sobre Aurora Serverless v2, consulte Uso de Aurora Serverless v2.

Actualizaciones automáticas de versiones secundarias para Aurora Serverless v1

Cuando las versiones secundarias de los motores de base de datos se vuelven disponibles para Aurora Serverless v1, se aplican automáticamente en las distintas Regiones de AWS donde Aurora Serverless v1 está disponible. En otras palabras, no necesita actualizar el clúster de bases de datos de Aurora Serverless v1 para obtener una nueva versión secundaria del motor de base de datos del clúster cuando está disponible para Aurora Serverless v1.

Uso de la API de datos de Amazon RDS

Al utilizar la API de datos de RDS (API de datos), puede trabajar con una interfaz de servicios web para su clúster de base de datos. La API de datos no requiere una conexión persistente al clúster de base de datos. En su lugar, proporciona un punto de enlace HTTP seguro e integración con los AWS SDK. Puede usar el punto de enlace para ejecutar instrucciones SQL sin administrar conexiones.

Los usuarios no necesitan transferir credenciales con llamadas a la API de datos, porque la API de datos utiliza credenciales de base de datos almacenadas en AWS Secrets Manager. Para almacenar credenciales en Secrets Manager, se debe conceder a los usuarios los permisos adecuados para usar Secrets Manager y también la API de datos. Para obtener más información acerca de la autorización de usuarios, consulte Autorización del acceso a la API de datos de Amazon RDS.

También puede usar la API de datos para integrar Amazon Aurora con otras aplicaciones de AWS como AWS Lambda, AWS AppSync y AWS Cloud9. La API de datos proporciona una forma más segura de usar AWS Lambda. Le habilita para que obtenga acceso a su clúster de base de datos sin tener que configurar una función Lambda para obtener acceso a recursos de una Virtual Private Cloud (VPC). Para obtener más información, consulte AWS Lambda, AWS AppSync y AWS Cloud9.

Puede habilitar la API de datos al crear el clúster de Aurora DB. También puede modificar la configuración más adelante. Para obtener más información, consulte Habilitación de la API de datos de Amazon RDS.

Después de habilitar la API de datos, también puede utilizar el editor de consultas para ejecutar consultas ad hoc sin configurar una herramienta de consulta para acceder a Aurora en una VPC. Para obtener más información, consulte Uso del editor de consultas de Aurora.

Temas

- Disponibilidad de regiones y versiones para la API de datos de Amazon RDS
- Uso de IPv6 con la API de datos de Amazon RDS
- Limitaciones de la API de datos de Amazon RDS
- Comparación de los comportamientos de la API de datos de Amazon RDS para clústeres de Aurora Serverless v2 y aprovisionados con clústeres de Aurora Serverless v1
- Autorización del acceso a la API de datos de Amazon RDS
- Habilitación de la API de datos de Amazon RDS

- Creación de un punto de conexión de VPC de Amazon para la API de datos de Amazon RDS (AWS PrivateLink)
- Llamadas a la API de datos de Amazon RDS
- Usar la biblioteca de cliente de Java para la API de datos de RDS
- Procesamiento de resultados de consultas de API de datos de Amazon RDS en formato JSON
- Solución de problemas de la API de datos de Amazon RDS
- Registro de llamadas a la API de datos de Amazon RDS con AWS CloudTrail
- Supervisión de las consultas de la API de datos de RDS con Información de rendimiento

Disponibilidad de regiones y versiones para la API de datos de Amazon RDS

Para obtener información sobre las regiones y versiones de motores disponibles para la API de datos, consulte las siguientes secciones.

Tipo de clúster	Disponibilidad en regiones y versiones
Aurora PostgreSQL aprovisionada y sin servidor v2	API de datos con Aurora PostgreSQL sin servidor v2 y aprovisionada
Aurora MySQL sin servidor v2 y aprovisionada	API de datos con Aurora MySQL sin servidor v2 y aprovisionada
Aurora PostgreSQL sin servidor v1	API de datos con Aurora PostgreSQL sin servidor v1
Aurora MySQL sin servidor v1	API de datos con Aurora MySQL sin servidor v1

Si necesita módulos criptográficos validados por FIPS 140-2 al acceder a los datos de la API a través de una interfaz de línea de comandos o una API, utilice un punto de enlace de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Uso de IPv6 con la API de datos de Amazon RDS

La API de datos de Amazon RDS es compatible con la conectividad IPv6 a través de puntos de conexión de doble pila. Esto le permite conectarse a la API de datos mediante direcciones IPv6 y, al mismo tiempo, mantener la compatibilidad con versiones anteriores de IPv4.

Compatibilidad de puntos de conexión de IPv6

La API de datos proporciona puntos de conexión de doble pila que admiten conexiones IPv4 e IPv6. Estos puntos de conexión utilizan el dominio `.aws` en lugar del dominio tradicional `.amazonaws.com`.

Tipos de puntos de conexión disponibles

Puntos de conexión de doble pila públicos

- Formato: `rds-data.region.api.aws`
- Ejemplo:: `rds-data.us-east-1.api.aws`

Puntos de conexión de doble pila FIPS

- Formato: `rds-data-fips.region.api.aws`
- Ejemplo:: `rds-data-fips.us-east-1.api.aws`

Puntos de conexión IPv6 de PrivateLink

- Disponible a través de puntos de conexión de VPC compatibles con IPv6
- Permite la conectividad IPv6 privada dentro de la VPC

Puntos de conexión antiguos solo de IPv4

Los puntos de conexión de `.amazonaws.com` existentes siguen admitiendo conexiones solo de IPv4:

- `rds-data.region.amazonaws.com`
- `rds-data-fips.region.amazonaws.com`

> **Note**
>
> Los puntos de conexión antiguos permanecen inalterados para garantizar la compatibilidad con versiones anteriores de las aplicaciones existentes.

Uso de puntos de conexión de IPv6

Para usar IPv6 con la API de datos, actualice la aplicación para usar los nuevos puntos de conexión de doble pila. La aplicación utilizará IPv6 automáticamente, si está disponible, o recurrirá a IPv4.

Para obtener instrucciones generales sobre la configuración de IPv6 en la VPC, consulte Migración a IPv6 en la Guía del usuario de Amazon VPC.

Puede configurar los puntos de conexión de IPv6 de dos maneras:

- Uso de la variable de entorno: configure `AWS_USE_DUALSTACK_ENDPOINT=true` en el entorno de IPv6. La AWS CLI y los AWS SDK crearán automáticamente los puntos de conexión de `api.aws` adecuados sin necesidad de que especifique las URL de los puntos de conexión manualmente.
- Uso de direcciones URL de punto de conexión explícitas: especifique la URL de punto de conexión de doble pila directamente en los comandos de la AWS CLI o en la configuración del SDK, tal y como se muestra en los ejemplos siguientes.

Configuración de la AWS CLI

Configure la AWS CLI para usar puntos de conexión de IPv6 especificando la URL del punto de conexión:

Para Linux, macOS o Unix:

```
aws rds-data execute-statement \
    --endpoint-url https://rds-data.us-east-1.api.aws \
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" \
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" \
    --database "mydb" \
    --sql "SELECT * FROM users LIMIT 10"
```

Para Windows:

```
aws rds-data execute-statement ^
    --endpoint-url https://rds-data.us-east-1.api.aws ^
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" ^
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" ^
    --database "mydb" ^
    --sql "SELECT * FROM users LIMIT 10"
```

Configuración de AWS SDK

Configure los AWS SDK para utilizar puntos de conexión de doble pila:

Python

```
import boto3

# Create RDS Data API client with IPv6 dual-stack endpoint
client = boto3.client(
    'rds-data',
    endpoint_url='https://rds-data.us-east-1.api.aws'
)

# Execute a SQL statement
response = client.execute_statement(
    resourceArn='arn:aws:rds:us-east-1:123456789012:cluster:my-cluster',
    secretArn='arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret',
    database='mydb',
    sql='SELECT * FROM users LIMIT 10'
)

print(response['records'])
```

Java

```
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import java.net.URI;

// Create RDS Data API client with IPv6 dual-stack endpoint
RdsDataClient client = RdsDataClient.builder()
    .endpointOverride(URI.create("https://rds-data.us-east-1.api.aws"))
    .build();
```

```
// Execute a SQL statement
ExecuteStatementRequest request = ExecuteStatementRequest.builder()
    .resourceArn("arn:aws:rds:us-east-1:123456789012:cluster:my-cluster")
    .secretArn("arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret")
    .database("mydb")
    .sql("SELECT * FROM users LIMIT 10")
    .build();

ExecuteStatementResponse response = client.executeStatement(request);
System.out.println(response.records());
```

JavaScript

```
const { RDSDataClient, ExecuteStatementCommand } = require("@aws-sdk/client-rds-
data");

// Create RDS Data API client with IPv6 dual-stack endpoint
const client = new RDSDataClient({
    endpoint: "https://rds-data.us-east-1.api.aws"
});

// Execute a SQL statement
const command = new ExecuteStatementCommand({
    resourceArn: "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster",
    secretArn: "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret",
    database: "mydb",
    sql: "SELECT * FROM users LIMIT 10"
});

const response = await client.send(command);
console.log(response.records);
```

Uso de AWS PrivateLink con IPv6

Puede crear puntos de conexión de VPC para la API de datos que admitan la conectividad de IPv6 dentro de la VPC. Para obtener instrucciones detalladas sobre cómo crear puntos de conexión de VPC para API de datos, consulte Creación de un punto de conexión de VPC de Amazon para la API de datos de Amazon RDS (AWS PrivateLink).

Al crear un punto de conexión de VPC compatible con IPv6, asegúrese de que:

- La VPC y las subredes están configuradas para admitir IPv6
- Los grupos de seguridad permiten el tráfico de IPv6 en los puertos necesarios (normalmente 443 para HTTPS)
- Las ACL de red están configuradas para permitir el tráfico de IPv6

Consideraciones sobre la migración

Cuando migre a puntos de conexión de IPv6, tenga en cuenta lo siguiente:

- Migración gradual: puede migrar las aplicaciones de forma gradual actualizando las URL de los puntos de conexión, una aplicación a la vez.
- Compatibilidad de red: asegúrese de que la infraestructura de red sea compatible con IPv6 antes de realizar la migración.
- Políticas de seguridad: actualice las reglas de los grupos de seguridad y las ACL de red para permitir el tráfico de IPv6 si es necesario.
- Supervisión: actualice las configuraciones de supervisión y registro para gestionar las direcciones IPv6.

> **Note**
>
> Direcciones de conexión de bases de datos: cuando se utilizan puntos de conexión de IPv6 para la API de datos, las conexiones de la base de datos subyacentes y los registros de la base de datos seguirán mostrando las direcciones IPv4. Este es el comportamiento esperado y no afecta a la funcionalidad de las aplicaciones habilitadas para IPv6.

Solución de problemas de conectividad de IPv6

Si tiene problemas con la conectividad de IPv6, verifique lo siguiente:

Configuración de red

Compruebe que la red sea compatible con IPv6 y que el enrutamiento de IPv6 esté configurado correctamente.

Resolución de los DNS

Asegúrese de que el solucionador de DNS pueda resolver los registros AAAA de los puntos de conexión de doble pila.

Grupos de seguridad

Actualice las reglas de los grupos de seguridad para permitir el tráfico de IPv6 en los puertos necesarios (normalmente 443 para HTTPS).

Bibliotecas del cliente

Compruebe que las bibliotecas del cliente de HTTP admiten IPv6 y la conectividad de doble pila.

Limitaciones de la API de datos de Amazon RDS

La API de datos de RDS tiene las siguientes limitaciones:

- Solo puede ejecutar consultas de la API de datos en instancias de escritura de un clúster de base de datos. Sin embargo, las instancias de escritura pueden aceptar consultas de escritura y lectura.
- Con las bases de datos globales de Aurora, puede habilitar la API de datos en los clústeres de bases de datos principales y secundarios. Sin embargo, un clúster secundario no tiene una instancia de escritor hasta que pase a ser el principal. La API de datos requiere acceso a la instancia de escritor para procesar las consultas, incluso para las consultas de lectura. Como resultado, las consultas de lectura y escritura enviadas al clúster secundario producen un error cuando este carece de una instancia de escritor. Cuando un clúster secundario promociona y tiene una instancia de escritor disponible, las consultas de la API de datos en esa instancia de base de datos se realizan correctamente.
- La API de datos no es compatible con las clases de instancias de base de datos T.
- Para clústeres de base de datos de Aurora Serverless v2 y aprovisionados, la API de datos de RDS no admite algunos tipos de datos. Para obtener una lista de los tipos admitidos, consulte the section called “API de datos con Aurora Serverless v2 en comparación con Aurora Serverless v1”.
- Para bases de datos de Aurora PostgreSQL versión 14 y posteriores, la API de datos solo admite `scram-sha-256` para el cifrado de contraseñas.
- El límite de tamaño de respuesta es de 1 MiB. Si la llamada devuelve más de 1 MiB de datos de respuesta, se terminará la llamada.
- Para Aurora Serverless v1, el número máximo de solicitudes por segundo es 1000. Para el resto de bases de datos compatibles, no hay límite.

- El límite de tamaño de la API de datos es 64 KB por fila en el conjunto de resultados devuelto por la base de datos. Asegúrese de que cada fila de un conjunto de resultados sea de 64 KB o menos.

Comparación de los comportamientos de la API de datos de Amazon RDS para clústeres de Aurora Serverless v2 y aprovisionados con clústeres de Aurora Serverless v1

Las mejoras más recientes de las API de datos de Amazon RDS permiten que las API de datos estén disponibles para los clústeres que utilizan versiones recientes de los motores de PostgreSQL o MySQL. Estos clústeres se pueden configurar para el uso de Aurora Serverless v2 o de clases de instancias aprovisionadas, como `db.r6g` o `db.r6i`.

Las secciones siguientes describen las diferencias de la API de datos de Amazon RDS entre clústeres de bases de datos de Aurora Serverless v2 y aprovisionados y los clústeres de bases de datos de Aurora Serverless v1. Aurora Serverless v1 Los clústeres de bases de datos utilizan el modo de motor `serverless`. Los clústeres de bases de datos aprovisionados utilizan el modo de motor `provisioned`. Un clúster de base de datos de Aurora Serverless v2 también utiliza el modo de motor `provisioned` y contiene una o más instancias de base de datos de Aurora Serverless v2 con la clase de instancia `db.serverless`.

Número máximo de solicitudes por segundo

Aurora Serverless v1

Las API de datos pueden realizar hasta 1000 solicitudes por segundo.

Aurora Serverless v2

Las API de datos pueden realizar un número ilimitado de solicitudes por segundo.

Habilitación o desactivación de la API de datos de Amazon RDS en una base de datos existente

Aurora Serverless v1

- Con la API de Amazon RDS: use la operación `ModifyCluster` y especifique `True` o `False`, según proceda, para el parámetro `EnableHttpEndpoint`.

- Con la AWS CLI: use la operación `modify-db-cluster` con la opción `--enable-http-endpoint` o `--no-enable-http-endpoint`, según proceda.

Aurora Serverless v2

- Con la API de Amazon RDS: utilice las operaciones `EnableHttpEndpoint` y `DisableHttpEndpoint`.
- Con la AWS CLI: utilice las operaciones `enable-http-endpoint` y `disable-http-endpoint`.

Eventos de CloudTrail

Aurora Serverless v1

Los eventos de las llamadas a la API de datos son eventos de administración. De forma predeterminada, estos eventos se incluyen automáticamente de un registro. Para obtener más información, consulte the section called “Excluir eventos de la API de datos de un seguimiento de CloudTrail (solo Aurora Serverless v1)”.

Aurora Serverless v2

Los eventos de las llamadas a la API de datos son eventos de datos. De forma predeterminada, estos eventos se excluyen automáticamente de un registro. Para obtener más información, consulte the section called “Inclusión de eventos de la API de datos en un seguimiento de CloudTrail”.

Compatibilidad con instrucciones múltiples

Aurora Serverless v1

- Aurora MySQL no admite las instrucciones múltiples.
- Para Aurora PostgreSQL, las instrucciones múltiples devuelven solo la primera respuesta a la consulta.

Aurora Serverless v2

Las instrucciones múltiples no son compatibles. Intentar ejecutar varias instrucciones en una sola llamada a la API devuelve “`An error occurred (ValidationException) when calling the ExecuteStatement operation: Multistatements aren't supported.`”. Para

ejecutar varias instrucciones, realice llamadas a la API `ExecuteStatement` independientes o utilice `BatchExecuteStatement` para el procesamiento por lotes.

El siguiente ejemplo muestra el mensaje de error resultante de una llamada a la API que intenta ejecutar una instrucción múltiple.

```
 aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table; Select * FROM next_table;

                                "An error occurred (ValidationException) when calling
 the ExecuteStatement operation: Multistatements aren't supported.
```

El siguiente ejemplo ejecuta varias instrucciones con llamadas a la API `ExecuteStatement` independientes.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table;"

aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM next_table;"
```

Solicitudes simultáneas para el mismo ID de transacción

Aurora Serverless v1

Las solicitudes subsiguientes esperan hasta que finalice la solicitud actual. Su aplicación debe gestionar los errores de tiempo de espera si el período de espera es demasiado largo.

Aurora Serverless v2

Cuando la API de datos recibe varias solicitudes con el mismo ID de transacción, devuelve inmediatamente este error:

`DatabaseErrorException: Transaction is still running a query`

Este error se produce en dos situaciones:

- La aplicación realiza solicitudes asíncronas (como las promesas de JavaScript) con el mismo ID de transacción.
- Aún se está procesando una solicitud anterior con ese ID de transacción.

El siguiente ejemplo muestra todas las solicitudes ejecutadas en paralelo con `promise.all()`.

```
const api_calls = [];
for (let i = 0; i < 10; i++) {
api_calls.push(
    client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    )
);
}
await Promise.all(api_calls);
```

Para resolver este error, espere a que finalice la solicitud actual antes de enviar otra solicitud con el mismo ID de transacción o elimine el ID de transacción para permitir las solicitudes en paralelo.

El siguiente ejemplo muestra una llamada a la API que utiliza la ejecución secuencial con el mismo ID de transacción.

```
 for (let i = 0; i < 10; i++) {
    await client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    ).promise()
);
}
```

Comportamiento de BatchExecuteStatement

Para obtener más información acerca de `BatchExecuteStatement`, consulte BatchExecuteStatement.

Aurora Serverless v1

El objeto de campos generado en el resultado de la actualización incluye los valores insertados.

Aurora Serverless v2

- Para Aurora MySQL, el objeto de campos generado en el resultado de la actualización incluye los valores insertados.
- En Aurora PostgreSQL, el objeto de campos generado está vacío.

Comportamiento de ExecuteSQL

Para obtener más información acerca de `ExecuteSQL`, consulte ExecuteSQL.

Aurora Serverless v1

La operación `ExecuteSQL` no está disponible.

Aurora Serverless v2

La operación `ExecuteSQL` no es compatible.

Comportamiento de ExecuteStatement

Para obtener más información acerca de `ExecuteStatement`, consulte ExecuteStatement.

Aurora Serverless v1

El parámetro `ExecuteStatement` admite la recuperación de columnas de matrices multidimensionales y todos los tipos de datos avanzados.

Aurora Serverless v2

El parámetro `ExecuteStatement` no admite columnas de matrices multidimensionales. Tampoco admite determinados tipos de datos de PostgreSQL, incluidos los tipos geométricos y monetarios. Cuando una API de datos encuentra un tipo de datos no compatible, devuelve este error:

`UnsupportedResultException: The result contains the unsupported data type data_type`.

Para evitar este problema, convierta el tipo de datos no compatible a `TEXT`. El siguiente ejemplo convierte un tipo de datos no compatible en `TEXT`.

```
SELECT custom_type::TEXT FROM my_table;--
ORSELECT CAST(custom_type AS TEXT) FROM my_table;
```

Para obtener una lista de los tipos de datos admitidos para cada motor de base de datos de Aurora, consulte Referencia de operaciones de la API de datos.

Comportamiento del parámetro de esquema

Aurora Serverless v1

El parámetro `Schema` no es compatible. Cuando incluye el parámetro `Schema` en una llamada a la API, la API de datos lo ignora.

Aurora Serverless v2

El parámetro `Schema` no está disponible. Cuando incluye el parámetro `Schema` en una llamada a la API, la API de datos devuelve este error: `ValidationException: The schema parameter isn't supported`. El siguiente ejemplo muestra una llamada a la API de datos que devuelve el error `ValidationException`.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--schema "your_schema" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Para solucionar este problema, elimine el parámetro `Schema` de la llamada a la API.

El siguiente ejemplo muestra una llamada a la API de datos con el parámetro `Schema` eliminado.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
```

```
--sql "SELECT * FROM your_table LIMIT 10"
```

Autorización del acceso a la API de datos de Amazon RDS

Los usuarios solo pueden invocar operaciones de la API de datos de Amazon RDS (API de datos) si están autorizados a hacerlo. Puede conceder a un usuario permiso para utilizar la API de datos asociando una política de AWS Identity and Access Management (IAM) que defina sus privilegios. También puede asociar la política a un rol si utiliza roles de IAM. Una política administrada por AWS, `AmazonRDSDataFullAccess`, incluye permisos para la API de datos.

La política `AmazonRDSDataFullAccess` también incluye permisos para que el usuario obtenga el valor de un secreto de AWS Secrets Manager. Los usuarios deben usar Secrets Manager para almacenar secretos que pueden usar en sus llamadas a la API de datos. El uso de secretos significa que los usuarios no tienen que incluir credenciales de base de datos para los recursos a los que se dirigen en sus llamadas a la API de datos. La API de datos llama de forma transparente a Secrets Manager, lo que permite (o deniega) la solicitud del secreto del usuario. Para obtener información acerca de cómo configurar secretos para utilizarlos con la API de datos, consulte Almacenamiento de credenciales de base de datos en AWS Secrets Manager.

La política `AmazonRDSDataFullAccess` proporciona acceso completo (a través de la API de datos) a los recursos. Puede restringir el ámbito definiendo sus propias políticas que especifiquen el nombre de recurso de Amazon (ARN) de un recurso.

Por ejemplo, la siguiente política muestra un ejemplo de los permisos mínimos necesarios para que un usuario acceda a la API de datos para el clúster de base de datos identificado por su ARN. La política incluye los permisos necesarios para acceder a Secrets Manager y obtener autorización a la instancia de base de datos para el usuario.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
```

```
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:prod"
        }
    ]
}
```

Le recomendamos que utilice un ARN específico para el elemento «Recursos» en sus declaraciones de política (como se muestra en el ejemplo) en lugar de un comodín (*).

Trabajo con autorización basada en etiquetas

La API de datos de RDS (API de datos) y Secrets Manager admiten autorización basada en etiquetas. Las etiquetas son pares clave-valor que etiquetan un recurso, como un clúster RDS, con un valor de cadena adicional, por ejemplo:

- `environment:production`
- `environment:development`

Puede aplicar etiquetas a los recursos para la asignación de costos, soporte de operaciones, control de acceso y muchas otras razones. (Si aún no tiene etiquetas en sus recursos y desea aplicarlas, puede obtener más información en Etiquetado de recursos de Amazon RDS). Puede utilizar las etiquetas en las instrucciones de política para limitar el acceso a los clústeres de RDS que están etiquetados con estas etiquetas. Por ejemplo, un clúster de base de datos de Aurora puede tener etiquetas que identifican su entorno como producción o desarrollo.

En el ejemplo siguiente se muestra cómo se pueden utilizar etiquetas en las instrucciones de política. Esta instrucción requiere que tanto el clúster como el secreto transferido en la solicitud de API de datos tengan una etiqueta `environment:production`.

Así es como se aplica la política: cuando un usuario realiza una llamada utilizando la API de datos, la solicitud se envía al servicio. La API de datos comprueba primero que el ARN del clúster transferido en la solicitud esté etiquetado con `environment:production`. A continuación, llama a Secrets Manager para recuperar el valor del secreto del usuario en la solicitud. Secrets Manager también verifica que el secreto del usuario esté etiquetado con `environment:production`. Si es así, la API de datos utiliza el valor recuperado para la contraseña de base de datos del usuario. Finalmente, si eso también es correcto, la solicitud de la API de datos se invoca correctamente para el usuario.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                 "secretsmanager:GetSecretValue"
               ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
             }
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                  "rds-data:*"
               ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
             }
```

```
        }
    ]
}
```

En el ejemplo se muestran acciones independientes para `rds-data` y `secretsmanager` para la API de datos y Secrets Manager. Sin embargo, puede combinar acciones y definir condiciones de etiqueta de muchas maneras distintas para admitir casos de uso específicos. Para obtener más información, consulte Uso de políticas basadas en identidad (políticas de IAM) para Secrets Manager.

En el elemento «Condición» de la política, puede elegir claves de etiqueta entre las siguientes:

- `aws:TagKeys`
- `aws:ResourceTag/${TagKey}`

Para obtener más información sobre las etiquetas de recursos y cómo utilizar `aws:TagKeys`, consulte Control del acceso a los recursos de AWS mediante etiquetas de recursos.

> **Note**
>
> Tanto la API de datos como AWS Secrets Manager autorizan usuarios. Si no tiene permisos para todas las acciones definidas en una política, obtendrá un error `AccessDeniedException`.

Almacenamiento de credenciales de base de datos en AWS Secrets Manager

Al llamar a la API de datos de Amazon RDS (API de datos), puede transferir las credenciales del clúster de base de datos de Aurora mediante un secreto en Secrets Manager. Para pasar credenciales mediante este método, especifique el nombre del secreto o el Nombre de recurso de Amazon (ARN) del secreto.

Para almacenar las credenciales de clúster de base de datos en un secreto

1. Utilice Secrets Manager para crear un secreto que contenga credenciales para el clúster de base de datos de Aurora.

Para obtener instrucciones, consulte Creación de un secreto básico en la Guía del usuario de AWS Secrets Manager.

2. Utilice la consola de Secrets Manager para ver los detalles del secreto que ha creado, o ejecute el comando `aws secretsmanager describe-secret` de la AWS CLI.

   Anote el nombre y el ARN del secreto. Puede utilizarlos en llamadas a la API de datos.

Para obtener más información acerca de cómo utilizar Secrets Manager, consulte la Guía del usuario de Secrets Manager de AWS.

Para comprender cómo administra Amazon Aurora Identity and Access Management, consulte Cómo funciona Amazon Aurora con IAM.

Para obtener más información acerca de cómo crear una política de IAM, consulte Creación de políticas de IAM en la Guía del usuario de IAM. Para obtener información sobre cómo añadir una política de IAM a un usuario, consulte Adición y eliminación de permisos de identidad de IAM en la Guía del usuario de IAM.

Habilitación de la API de datos de Amazon RDS

Para utilizar la API de datos de Amazon RDS (API de datos), habilítela para el clúster de base de datos de Aurora. Puede habilitar la API de datos cuando cree o modifique el clúster de base de datos.

> **Note**
>
> La disponibilidad de la API de datos para el clúster depende de la versión de Aurora, del motor de base de datos y de la región de AWS. En las versiones anteriores de Aurora, la API de datos solo funciona con los clústeres de Aurora Serverless v1. En las versiones más recientes de Aurora, la API de datos funciona con clústeres que utilizan tanto instancias de Aurora Serverless v2 como aprovisionadas. Para comprobar si el clúster puede utilizar la API de datos, consulte Regiones y motores de base de datos Aurora admitidos para API de datos de RDS.

Temas

- Habilitación de la API de datos de RDS al crear una base de datos

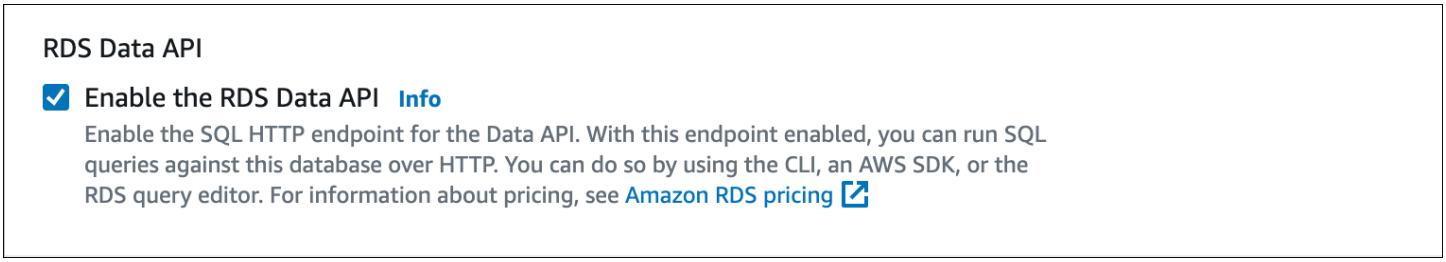
- Habilitación de la API de datos de RDS en una base de datos existente

Habilitación de la API de datos de RDS al crear una base de datos

Al crear una base de datos compatible con la API de datos de RDS (API de datos), puede habilitar esta característica. Los siguientes procedimientos describen el proceso al utilizar la Consola de administración de AWS, la AWS CLI o la API de RDS.

Consola

Para habilitar la API de datos al crear un clúster de base de datos, seleccione la casilla Habilitar la API de datos de RDS en la sección Conectividad de la página Crear base de datos, como se muestra en la siguiente captura de pantalla.



Para obtener instrucciones sobre cómo crear un clúster de base de datos de Aurora que pueda usar la API de datos de RDS, consulte lo siguiente:

- En clústeres de Aurora Serverless v2 y aprovisionados: Creación de un clúster de base de datos de Amazon Aurora
- Para:Aurora Serverless v1 Creación de un clúster de bases de datos de Aurora Serverless v1

AWS CLI

Para habilitar la API de datos mientras crea un clúster de base de datos de Aurora, ejecute el comando de la AWS CLI create-db-clúster con la opción `--enable-http-endpoint`.

En el ejemplo siguiente se crea un clúster de base de datos de Aurora PostgreSQL con la API de datos habilitada.

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
```

```
    --db-cluster-identifier my_pg_cluster \
    --engine aurora-postgresql \
    --enable-http-endpoint
```

Para Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my_pg_cluster ^
    --engine aurora-postgresql ^
    --enable-http-endpoint
```

API de RDS

Para habilitar la API de datos al crear un clúster de base de datos de Aurora, utilice la operación CreateDBclúster con el valor del parámetro `EnableHttpEndpoint` establecido en `true`.

Habilitación de la API de datos de RDS en una base de datos existente

Puede modificar un clúster de base de datos que admita la API de datos de RDS (API de datos) para activar o desactivar esta característica.

Temas

- Habilitación o deshabilitación de la API de datos (Aurora Serverless v2 y aprovisionada)
- Habilitación o deshabilitación de la API de datos (solo Aurora Serverless v1)

Habilitación o deshabilitación de la API de datos (Aurora Serverless v2 y aprovisionada)

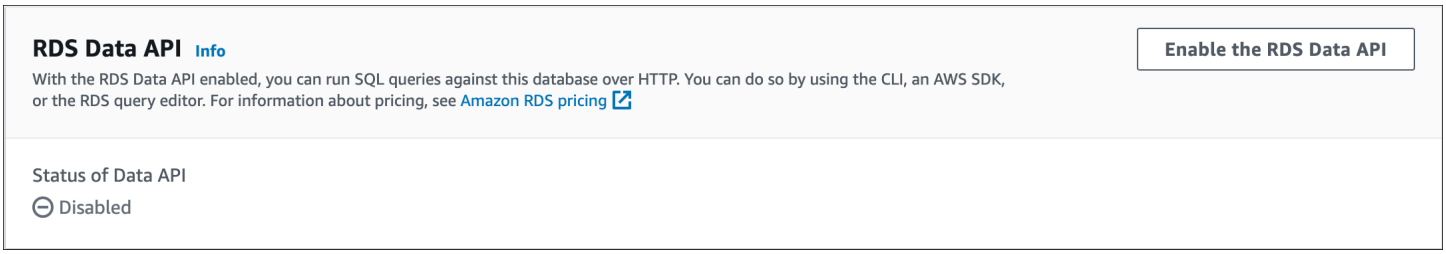
Utilice los siguientes procedimientos para habilitar o deshabilitar la API de datos en las bases de datos de Aurora Serverless v2 y aprovisionadas. Para activar o desactivar la API de datos en las bases de datos de Aurora Serverless v1, utilice los procedimientos descritos en the section called "Habilitación o deshabilitación de la API de datos (solo Aurora Serverless v1)".

Consola

Puede habilitar o deshabilitar la API de datos con la consola de RDS para un clúster de base de datos que sea compatible con esta característica. Para ello, abra la página de detalles del clúster de la base de datos en la que desee habilitar o deshabilitar la API de datos y, en la pestaña

Conectividad y seguridad, vaya a la sección API de datos de RDS. Esta sección muestra el estado de la API de datos y le permite habilitarla o deshabilitarla.

La siguiente captura de pantalla muestra que la API de datos de RDS no está habilitada.



AWS CLI

Para habilitar o deshabilitar la API de datos en una base de datos existente, ejecute el comando de la AWS CLI enable-http-endpoint o disable-http-endpoint y especifique el ARN de su clúster de base de datos.

En el ejemplo siguiente se habilita la API de datos.

Para Linux, macOS o Unix:

```
aws rds enable-http-endpoint \
    --resource-arn cluster_arn
```

Para Windows:

```
aws rds enable-http-endpoint ^
    --resource-arn cluster_arn
```

API de RDS

Para habilitar o deshabilitar la API de datos en una base de datos existente, utilice las operaciones EnableHttpEndpoint y DisableHttpEndpoint.

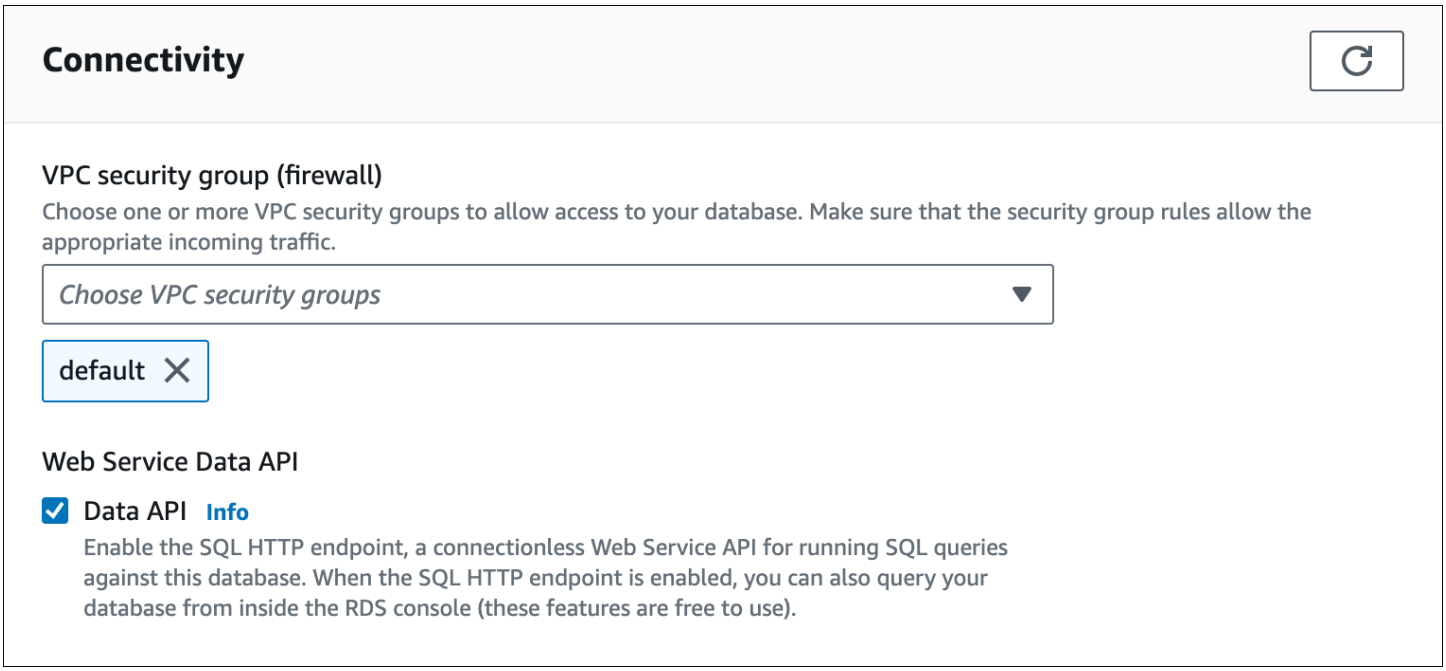
Habilitación o deshabilitación de la API de datos (solo Aurora Serverless v1)

Para habilitar o deshabilitar la API de datos en las bases de datos de Aurora Serverless v1 existentes. Para habilitar o deshabilitar la API de datos en las bases de datos de Aurora Serverless v2 y aprovisionadas, utilice los procedimientos indicados en the section called “Habilitación o deshabilitación de la API de datos”.

Consola

Cuando cree o modifique un clúster de base de datos de Aurora Serverless v1, debe habilitar la API de datos en la sección Conectividad de la consola de RDS.

En la siguiente captura de pantalla se muestra la API de datos cuando se modifica un clúster de base de datos de Aurora.



Para obtener instrucciones sobre cómo modificar un clúster de base de datos de Aurora Serverless v1, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1.

AWS CLI

Para habilitar o deshabilitar la API de datos, ejecute el comando de la AWS CLI modify-db-clúster con `--enable-http-endpoint` o `--no-enable-http-endpoint`, según corresponda.

En el ejemplo siguiente se habilita la API de datos en `sample-cluster`.

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --enable-http-endpoint
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --enable-http-endpoint
```

API de RDS

Para habilitar la API de datos, utilice la operación ModifyDBclúster y establezca el valor de `EnableHttpEndpoint` en `true` o `false`, según corresponda.

Creación de un punto de conexión de VPC de Amazon para la API de datos de Amazon RDS (AWS PrivateLink)

La Amazon VPC le permite lanzar recursos de AWS, como clústeres de bases de datos y aplicaciones de Aurora, en una Virtual Private Cloud (VPC). AWS PrivateLink proporciona conectividad privada entre las VPC y los servicios de AWS con alta seguridad en la red de Amazon. Con AWS PrivateLink, puede crear puntos de enlace de la Amazon VPC que le permiten conectarse a servicios a través de diferentes cuentas y VPC basados en Amazon VPC. Para obtener más información acerca de AWS PrivateLink, consulte Servicios de punto de enlace de la VPC (AWS PrivateLink) en la guía del usuario de Amazon Virtual Private Cloud.

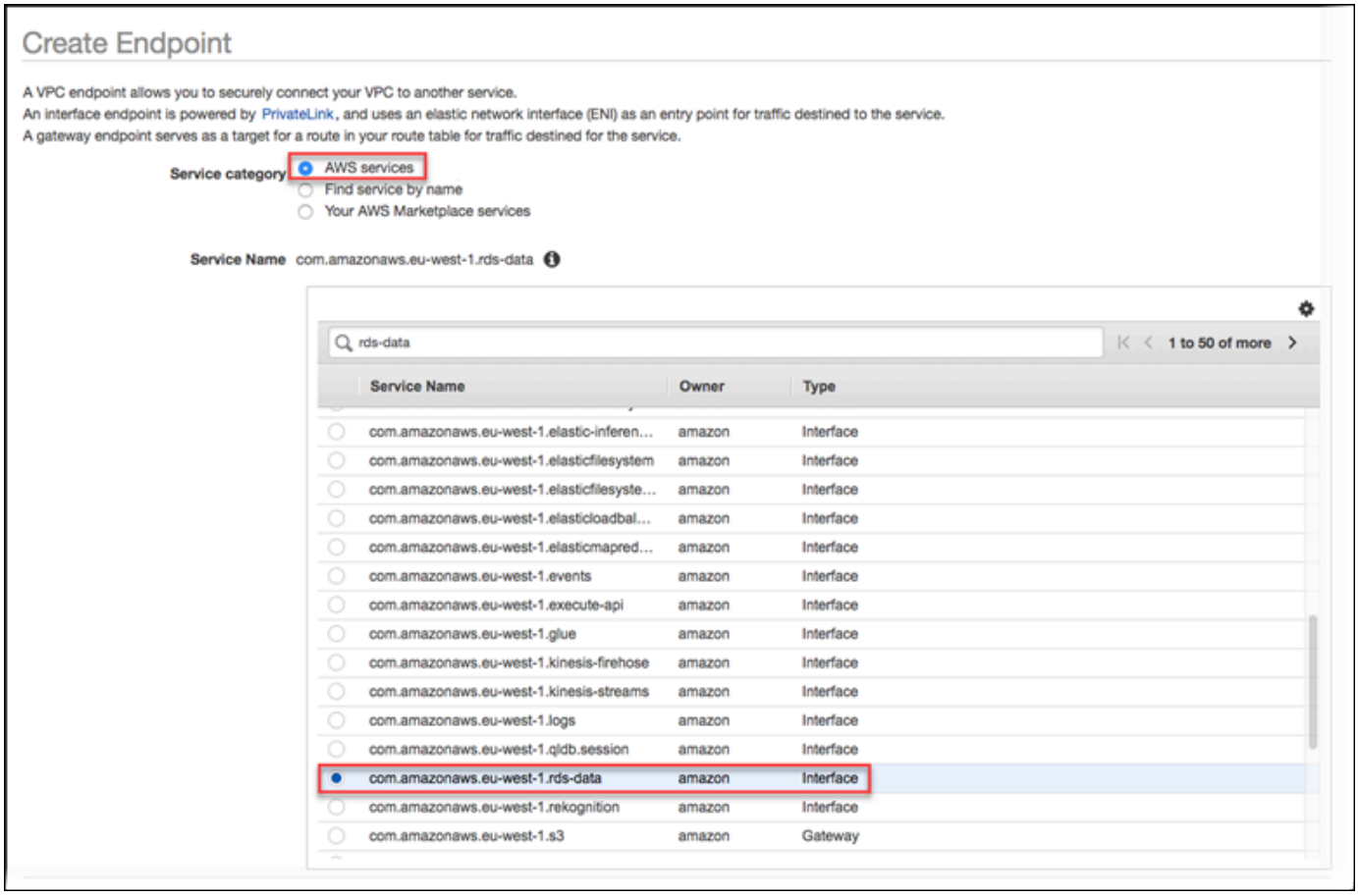
Puede llamar a la API de datos de RDS (API de datos) con los puntos de conexión de la Amazon VPC. El uso de un punto de conexión de VPC de Amazon mantiene el tráfico entre las aplicaciones de su Amazon VPC y la API de datos en la red de AWS, sin usar direcciones IP públicas. Los puntos de enlace de la Amazon VPC pueden ayudarle a cumplir los requisitos reglamentarios y de conformidad relacionados con la limitación de la conectividad a internet público. Por ejemplo, si utiliza un punto de conexión de VPC de Amazon, puede mantener el tráfico entre una aplicación que se ejecuta en una instancia Amazon EC2 y la API de datos en las VPC donde se contienen.

Después de crear el punto de enlace de la Amazon VPC, puede comenzar a usarlo sin realizar ningún cambio de código o configuración en la aplicación.

Para crear un punto de conexión de VPC de Amazon para la API de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Elija Endpoints (Puntos de enlace) y, a continuación, elija Create Endpoint (Crear punto de enlace).

3. En la página Create Endpoint (Crear punto de conexión), en Service category (Categoría de servicio), elija AWS services (Servicios de AWS). En Service Name (Nombre del servicio), elija rds-data.



4. Para VPC, elija la VPC en la que crear el punto de enlace.

   Elija la VPC que contiene la aplicación que realiza llamadas a la API de datos.

5. En Subnets (Subredes), elija la subred para cada zona de disponibilidad (AZ) utilizada por el servicio de AWS que ejecuta la aplicación.

Para crear un punto de enlace de la Amazon VPC, especifique el rango de direcciones IP privadas en el que se podrá acceder al punto de enlace. Para ello, elija la subred para cada zona de disponibilidad. Al hacerlo, se restringe el punto de enlace de la VPC al rango de direcciones IP privadas específico de cada zona de disponibilidad y también se crea un punto de enlace de la Amazon VPC en cada zona de disponibilidad.

6. En Enable Private DNS Name (Habilitar nombre de DNS privado), seleccione Enable for this endpoint (Habilitar para este punto de enlace).

El DNS privado resuelve el nombre de host de DNS de la API de datos estándar (`https://rds-data.`*`region`*`.amazonaws.com`) en las direcciones IP privadas asociadas con el nombre de host de DNS específico del punto de enlace de la Amazon VPC. Como resultado, puede acceder al punto de conexión de la VPC de la API de datos utilizando los SDK de AWS CLI o la AWS sin realizar ningún cambio de código o configuración para actualizar la URL del punto de conexión de la API de datos.

7. En Security group (Grupo de seguridad), elija los grupos de seguridad que deban asociarse al punto de enlace de la Amazon VPC.

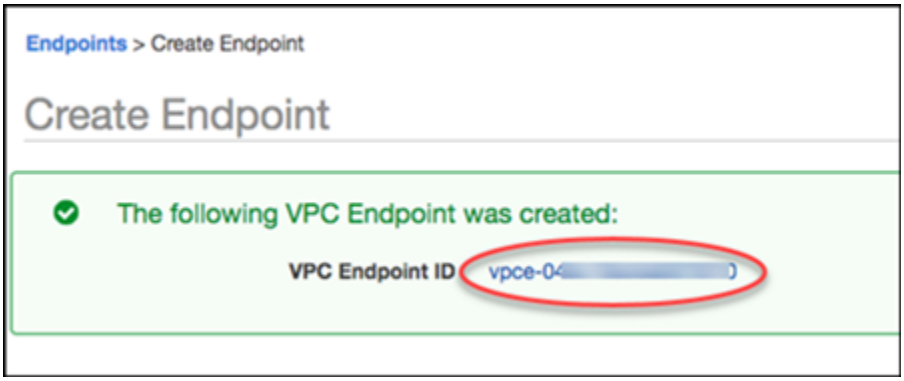
   Elija el grupo de seguridad que permita el acceso al servicio de AWS que ejecuta la aplicación. Por ejemplo, si una instancia Amazon EC2 está ejecutando la aplicación, elija el grupo de seguridad que permita el acceso a la instancia Amazon EC2. El grupo de seguridad le permite controlar el tráfico al punto de enlace de la Amazon VPC desde los recursos de la VPC.

8. En Policy (Política), elija Full Access (Acceso total) para permitir que cualquier persona dentro de la Amazon VPC acceda a la API de datos a través de este punto de enlace. O bien, elija Custom (Personalizado) para especificar una política que limite el acceso.

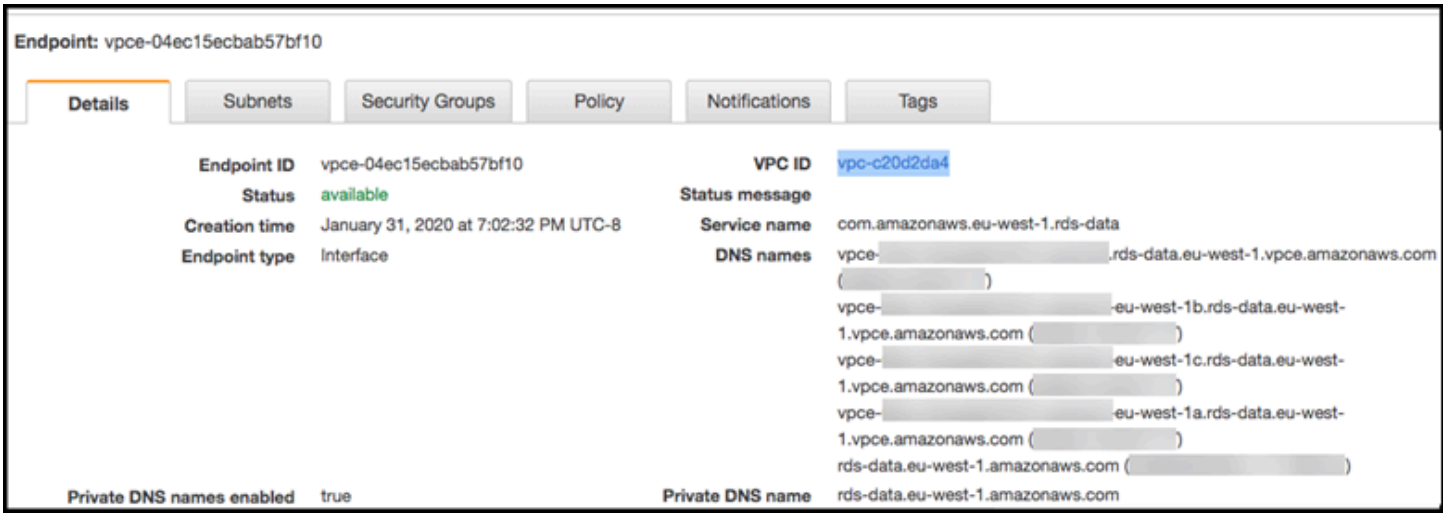
   Si eligeCustom (Personalizado), introduzca la política en la herramienta de creación de políticas.

9. Elija Create endpoint.

Una vez creado el punto de enlace, elija el vínculo en la Consola de administración de AWS para ver los detalles del punto de enlace.



La ficha Details (Detalles) del punto de enlace muestra los nombres de host de DNS que se generaron al crear el punto de enlace de la Amazon VPC.



Puede utilizar el punto de enlace estándar (`rds-data.`*`region`*`.amazonaws.com`) o uno de los puntos de enlace específicos de la VPC para llamar a la API de datos dentro de la Amazon VPC. El punto de enlace de la API de datos estándar se dirige automáticamente al punto de enlace de la Amazon VPC. Este enrutamiento se produce porque cuando se creó el punto de enlace de la Amazon VPC se habilitó el nombre de host de DNS privado.

Cuando utiliza un punto de conexión de VPC de Amazon en una llamada a la API de datos, todo el tráfico entre la aplicación y la API de datos permanece en las Amazon VPC donde se contienen. Puede usar un punto de enlace de la Amazon VPC para cualquier tipo de llamada a la API de datos. Para obtener más información sobre la llamada a la API de datos, consulte Llamadas a la API de datos de Amazon RDS.

Llamadas a la API de datos de Amazon RDS

Con la API de datos de Amazon RDS (API de datos) habilitada en el clúster de base de datos de Aurora, puede ejecutar instrucciones SQL en el clúster de base de datos de Aurora mediante la API de datos o la AWS CLI. La API de los datos es compatible con los idiomas de programación compatibles con AWS SDK. Para obtener más información, consulte Herramientas para crear en AWS.

Temas

- Referencia de las operaciones de la API de datos de Amazon RDS
- Llamadas a la API de datos de Amazon RDS con la AWS CLI
- Llamadas a la API de datos de Amazon RDS desde una aplicación Python
- Llamadas a la API de datos de Amazon RDS desde una aplicación Java
- Control del comportamiento del tiempo de espera de la API de datos

Referencia de las operaciones de la API de datos de Amazon RDS

La API de datos de Amazon RDS proporciona las operaciones siguientes para realizar instrucciones SQL.

Operación de la API	AWS CLI command	Descripción
ExecuteStatement	aws rds-data execute-statement	Ejecuta una instrucción SQL en una base de datos.
BatchExecuteStatement	aws rds-data batch-execute-statement	Ejecuta una instrucción SQL por lotes en una matriz de datos para operaciones de inserción y actualización masivas. Puede ejecutar una instrucción de lenguaje de manipulación de datos (DML) con una matriz de conjuntos de parámetros. Una instrucción SQL por lotes puede proporcionar una mejora significativa del rendimiento en comparación con las instrucciones de actualización e inserción individuales.

Puede utilizar cualquiera de las operaciones para ejecutar sentencias SQL individuales o para ejecutar transacciones. La API de datos proporciona las operaciones siguientes para las transacciones.

Operación de la API	AWS CLI command	Descripción
BeginTransaction	aws rds-data begin-transaction	Inicia una transacción SQL.
CommitTransaction	aws rds-data commit-transaction	Finaliza una transacción SQL y confirma los cambios.
RollbackTransaction	aws rds-data rollback-transaction	Ejecuta una restauración de una transacción.

Las operaciones para realizar instrucciones SQL y darle soporte a transacciones tienen los siguientes parámetros de la API de datos y opciones de AWS CLI comunes. Algunas operaciones dan soporte a otros parámetros u opciones.

Parámetro de operación de la API de datos	AWS CLIOpción de comando de la	Obligatorio	Descripción
resourceArn	--resource-arn	Sí	El nombre de recurso de Amazon (ARN) del clúster de base de datos de Aurora. El clúster debe estar en la misma Cuenta de AWS que el rol o el usuario de IAM que invoca la API de datos. Para acceder a un clúster en una cuenta diferente, asuma un rol en esa cuenta.

Parámetro de operación de la API de datos	AWS CLIOpción de comando de la	Obligatorio	Descripción
`secretArn`	`--secret-arn`	Sí	Nombre o ARN del secreto que permite el acceso al clúster de base de datos.

La API de datos de RDS admite los tipos de datos siguientes para Aurora MySQL:

- `TINYINT(1)`, `BOOLEAN`, `BOOL`
- `TINYINT`
- `SMALLINT` [`SIGNED` | `UNSIGNED`]
- `MEDIUMINT` [`SIGNED` | `UNSIGNED`]
- `INT` [`SIGNED` | `UNSIGNED`]
- `BIGINT` [`SIGNED` | `UNSIGNED`]
- `FLOAT`
- `DOUBLE`
- `VARCHAR`, `CHAR`, `TEXT`, `ENUM`
- `VARBINARY`, `BINARY`, `BLOB`
- `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`
- `DECIMAL`
- `JSON`
- `BIT`, `BIT(N)`

La API de datos de RDS admite los siguientes tipos escalares de Aurora PostgreSQL:

- `BOOL`
- `BYTEA`
- `DATE`
- `CIDR`

- DECIMAL, NUMERIC
- ENUM
- FLOAT8, DOUBLE PRECISION
- INET
- INT, INT4, SERIAL
- INT2, SMALLINT, SMALLSERIAL
- INT8, BIGINT, BIGSERIAL
- JSONB, JSON
- REAL, FLOAT
- TEXT, CHAR(N), VARCHAR, NAME
- TIME
- TIMESTAMP
- UUID
- VECTOR

La API de datos de RDS admite los siguientes tipos de matriz de Aurora PostgreSQL:

- BOOL[], BIT[]
- DATE[]
- DECIMAL[], NUMERIC[]
- FLOAT8[], DOUBLE PRECISION[]
- INT[], INT4[]
- INT2[]
- INT8[], BIGINT[]
- JSON[]
- REAL[], FLOAT[]
- TEXT[], CHAR(N)[], VARCHAR[], NAME[]
- TIME[]
- TIMESTAMP[]
- UUID[]

Puede usar parámetros en las llamadas a la API de datos para `ExecuteStatement` y `BatchExecuteStatement`, y cuando ejecuta los comandos de la AWS CLI `execute-statement` y `batch-execute-statement`. Para utilizar un parámetro, especifique un par de nombre-valor en el tipo de datos `SqlParameter`. Especifique el valor con el tipo de datos `Field`. En la tabla siguiente se mapean los tipos de datos de Java Database Connectivity (JDBC) con los tipos de datos que especifica en las llamadas a la API de datos.

Tipo de datos JDBC	Tipo de datos de la API de datos
`INTEGER, TINYINT, SMALLINT, BIGINT`	`LONG` (o `STRING`)
`FLOAT, REAL, DOUBLE`	`DOUBLE`
`DECIMAL`	`STRING`
`BOOLEAN, BIT`	`BOOLEAN`
`BLOB, BINARY, LONGVARBINARY, VARBINARY`	`BLOB`
`CLOB`	`STRING`
Otros tipos (incluidos los tipos relacionados con la fecha y hora)	`STRING`

> **Note**
>
> Puede especificar el tipo de datos `LONG` o `STRING` en la llamada de API de datos para `LONG` los valores devueltos por la base de datos. Le recomendamos que lo haga para evitar perder precisión para números extremadamente grandes, lo que puede suceder cuando trabaja con JavaScript.

Ciertos tipos, como `DECIMAL` y `TIME`, requieren una sugerencia para que la API de datos pase `String` valores a la base de datos como el tipo correcto. Para utilizar una sugerencia, incluya valores para `typeHint` en los tipos de datos `SqlParameter`. Los posibles valores de `typeHint` son:

- `DATE` –: el valor del parámetro `String` correspondiente se envía como un objeto de tipo `DATE` a la base de datos. El formato aceptado es `YYYY-MM-DD`.
- `DECIMAL` –: el valor del parámetro `String` correspondiente se envía como un objeto de tipo `DECIMAL` a la base de datos.
- `JSON` –: el valor del parámetro `String` correspondiente se envía como un objeto de tipo `JSON` a la base de datos.
- `TIME` –: el valor del parámetro `String` correspondiente se envía como un objeto de tipo `TIME` a la base de datos. El formato aceptado es `HH:MM:SS[.FFF]`.
- `TIMESTAMP` –: el valor del parámetro `String` correspondiente se envía como un objeto de tipo `TIMESTAMP` a la base de datos. El formato aceptado es `YYYY-MM-DD HH:MM:SS[.FFF]`.
- `UUID` –: el valor del parámetro `String` correspondiente se envía como un objeto de tipo `UUID` a la base de datos.

> **Note**
>
> Actualmente, la API de datos no admite matrices de identificadores únicos universales (UUID).

> **Note**
>
> Para Amazon Aurora PostgreSQL, la API de datos siempre devuelve el tipo de datos de Aurora PostgreSQL `TIMESTAMPTZ` en la zona horaria UTC.

Llamadas a la API de datos de Amazon RDS con la AWS CLI

Puede llamar a la API de datos de RDS (API de datos) mediante la AWS CLI.

En los siguientes ejemplos se utiliza la AWS CLI para la API de datos. Para obtener más información, consulte la referencia de AWS CLI de la API de datos.

En cada ejemplo, sustituya el nombre de recurso de Amazon (ARN) del clúster de base de datos por el ARN de su clúster de base de datos de Aurora. Reemplace también el ARN del secreto por el ARN del secreto de Secrets Manager que permite obtener acceso al clúster de base de datos.

> **Note**
>
> La AWS CLI puede dar formato a las respuestas de JSON.

Temas

- Inicio de una transacción SQL
- Ejecución de una instrucción SQL
- Ejecución de una instrucción SQL por lotes en una matriz de datos
- Confirmación de una transacción SQL
- Restauración de una transacción SQL

Inicio de una transacción SQL

Puede iniciar una transacción SQL ejecutando el comando de la CLI `aws rds-data begin-transaction`. La llamada devuelve un identificador de transacción.

> **Important**
>
> Dentro de la API de datos, el tiempo de la transacción se agota si no hay llamadas que usen su ID de transacción en un periodo de tres minutos. Si una transacción agota su tiempo antes de que se confirme, la API de datos se revertirá automáticamente.
> Las instrucciones de lenguaje de definición de datos (DDL) de MySQL dentro de una transacción causan una confirmación implícita. Recomendamos que ejecute cada instrucción DDL de MySQL en un comando `execute-statement` independiente con la opción `--continue-after-timeout`.

Además de las opciones comunes, especifique la opción `--database`, que proporciona el nombre de la base de datos.

Por ejemplo, el comando de la CLI siguiente inicia una transacción SQL.

Para Linux, macOS o Unix:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Para Windows:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "transactionId": "ABC1234567890xyz"
}
```

Ejecución de una instrucción SQL

Puede ejecutar una instrucción SQL usando el comando de la CLI `aws rds-data execute-statement`.

Puede ejecutar la instrucción SQL en una transacción especificando el identificador de transacción con la opción `--transaction-id`. Puede iniciar una transacción ejecutando el comando de la CLI `aws rds-data begin-transaction`. Puede finalizar y confirmar una transacción ejecutando el comando de la CLI `aws rds-data commit-transaction`.

> **Important**
>
> Si no especifica la opción `--transaction-id`, los cambios que se generan a partir de la llamada se confirman automáticamente.

Además de las opciones habituales, especifique las opciones siguientes:

- `--sql` (obligatorio): instrucción SQL que debe ejecutarse en el clúster de base de datos.
- `--transaction-id` (opcional): identificador de una transacción que se inició mediante el comando `begin-transaction` de la CLI. Especifique el ID de la transacción en la que desea incluir la instrucción SQL.
- `--parameters` (opcional): parámetros de la instrucción SQL.

- `--include-result-metadata | --no-include-result-metadata` (opcional): valor que indica si deben incluirse o no metadatos en los resultados. El valor predeterminado es `--no-include-result-metadata`.
- `--database` (opcional): el nombre de la base de datos.

  Es posible que la opción `--database` no funcione al ejecutar una instrucción SQL después de ejecutar `--sql "use database_name;"` en la solicitud anterior. Le recomendamos que utilice la opción `--database` en lugar de ejecutar instrucciones `--sql "use database_name;"`.
- `--continue-after-timeout | --no-continue-after-timeout` (opcional): un valor que indica si se seguirá o no ejecutando la instrucción después de que la llamada supere el intervalo de tiempo de espera de la API de datos de 45 segundos. El valor predeterminado es `--no-continue-after-timeout`.

  Para las instrucciones en lenguaje de definición de datos (DDL), recomendamos seguir ejecutando la instrucción después de que se agote el tiempo de la llamada, a fin de evitar errores y la posibilidad de que las estructuras de datos se dañen.
- `--format-records-as "JSON"|"NONE"`: valor opcional que especifica si se debe dar formato al conjunto de resultados como cadena JSON. El valor predeterminado es `"NONE"`. Para obtener información sobre el procesamiento de conjuntos de resultados JSON, consulte Procesamiento de resultados de consultas de API de datos de Amazon RDS en formato JSON.

El clúster de base de datos devuelve una respuesta para la llamada.

> **Note**
>
> El límite de tamaño de respuesta es de 1 MiB. Si la llamada devuelve más de 1 MiB de datos de respuesta, se terminará la llamada.
> Para Aurora Serverless v1, el número máximo de solicitudes por segundo es 1000. Para el resto de bases de datos compatibles, no hay límite.

Por ejemplo, el siguiente comando de la CLI ejecuta una única instrucción SQL y omite los metadatos en los resultados (valor predeterminado).

Para Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "select * from mytable"
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "select * from mytable"
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "numberOfRecordsUpdated": 0,
    "records": [
        [
            {
                "longValue": 1
            },
            {
                "stringValue": "ValueOne"
            }
        ],
        [
            {
                "longValue": 2
            },
            {
                "stringValue": "ValueTwo"
            }
        ],
        [
            {
                "longValue": 3
            },
            {
                "stringValue": "ValueThree"
            }
        ]
    ]
```

```
}
```

El siguiente comando de la CLI ejecuta una única instrucción SQL en una transacción mediante la opción `--transaction-id`.

Para Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "numberOfRecordsUpdated": 1
}
```

El siguiente comando de la CLI ejecuta una única instrucción SQL con parámetros.

Para Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "numberOfRecordsUpdated": 1
}
```

El siguiente comando de la CLI ejecuta una instrucción SQL de lenguaje de definición de datos (DDL). La instrucción DDL cambia el nombre de la columna `job` por la columna `role`.

> **Important**
>
> Para las instrucciones DDL, recomendamos seguir ejecutando la instrucción después de que se agote el tiempo de la llamada. Cuando se termina una instrucción DDL antes de que acabe de ejecutarse, pueden generarse errores y es posible que las estructuras de datos se dañen. Para seguir ejecutando una instrucción después de que una llamada supere el intervalo de tiempo de espera de la API de datos de RDS de 45 segundos, especifique la opción `--continue-after-timeout`.

Para Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Para Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "generatedFields": [],
    "numberOfRecordsUpdated": 0
}
```

> **Note**
>
> Los datos `generatedFields` no se admiten en Aurora PostgreSQL. Para obtener los valores de los campos generados, utilice la cláusula `RETURNING`. Para obtener más información, consulte Returning Data From Modified Rows en la documentación de PostgreSQL.

Ejecución de una instrucción SQL por lotes en una matriz de datos

Puede ejecutar una instrucción SQL por lotes en una matriz de datos ejecutando el comando `aws rds-data batch-execute-statement` de la CLI. Puede utilizar este comando para realizar una operación de actualización o importación masiva.

Puede ejecutar la instrucción SQL en una transacción especificando el identificador de transacción con la opción `--transaction-id`. Puede iniciar una transacción ejecutando el comando de la CLI `aws rds-data begin-transaction`. Puede finalizar y confirmar una transacción mediante el comando `aws rds-data commit-transaction` de la CLI.

> **Important**
>
> Si no especifica la opción `--transaction-id`, los cambios que se generan a partir de la llamada se confirman automáticamente.

Además de las opciones habituales, especifique las opciones siguientes:

- `--sql` (obligatorio): instrucción SQL que debe ejecutarse en el clúster de base de datos.

> **Tip**
>
> Para las sentencias compatibles con MySQL, no incluya un punto y coma al final del parámetro `--sql`. Un punto y coma final puede causar un error de sintaxis.

- `--transaction-id` (opcional): identificador de una transacción que se inició mediante el comando `begin-transaction` de la CLI. Especifique el ID de la transacción en la que desea incluir la instrucción SQL.
- `--parameter-set` (opcional): los conjuntos de parámetros para la operación por lotes.
- `--database` (opcional): el nombre de la base de datos.

El clúster de base de datos devuelve una respuesta a la llamada.

> **Note**
>
> No existe un límite máximo fijo en el número de conjuntos de parámetros. Sin embargo, el tamaño máximo de la solicitud HTTP enviada a través de la API de datos es de 4 MiB. Si la solicitud supera este límite, la API de datos devuelve un error y no procesa la solicitud. Este límite de 4 MiB incluye el tamaño de los encabezados HTTP y la notación JSON en la solicitud. Por lo tanto, el número de conjuntos de parámetros que puede incluir depende de una combinación de factores, como el tamaño de la sentencia SQL y el tamaño de cada conjunto de parámetros.
>
> El límite de tamaño de respuesta es de 1 MiB. Si la llamada devuelve más de 1 MiB de datos de respuesta, se terminará la llamada.
>
> Para Aurora Serverless v1, el número máximo de solicitudes por segundo es 1000. Para el resto de bases de datos compatibles, no hay límite.

Por ejemplo, el siguiente comando de la CLI ejecuta una instrucción SQL por lotes en una matriz de datos con un conjunto de parámetros.

Para Linux, macOS o Unix:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" \
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

Para Windows:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" ^
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

> **Note**
>
> No incluya saltos de línea en la opción `--parameter-sets`.

Confirmación de una transacción SQL

Con el comando `aws rds-data commit-transaction` de la CLI, puede finalizar una transacción SQL que inició con `aws rds-data begin-transaction` y confirmar los cambios.

Además de las opciones habituales, especifique la opción siguiente:

- `--transaction-id` (obligatorio): identificador de una transacción que se inició ejecutando el comando `begin-transaction` de la CLI. Especifique el ID de la transacción que desea finalizar y confirmar.

Por ejemplo, el siguiente comando de la CLI finaliza una transacción SQL y confirma los cambios.

Para Linux, macOS o Unix:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Para Windows:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "transactionStatus": "Transaction Committed"
}
```

Restauración de una transacción SQL

Con el comando `aws rds-data rollback-transaction` de la CLI, puede revertir una transacción SQL que inició con `aws rds-data begin-transaction`. Si revierte una transacción, cancelará sus cambios.

> **Important**
>
> Si el ID de la transacción ha vencido, la transacción se revertirá automáticamente. En este caso, un comando `aws rds-data rollback-transaction` que especifique el ID de transacción que ha vencido devolverá un error.

Además de las opciones habituales, especifique la opción siguiente:

- `--transaction-id` (obligatorio): identificador de una transacción que se inició ejecutando el comando `begin-transaction` de la CLI. Especifique el ID de la transacción que desea revertir.

Por ejemplo, el comando de la AWS CLI siguiente revierte una transacción SQL.

Para Linux, macOS o Unix:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Para Windows:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

A continuación se muestra un ejemplo de respuesta.

```
{
    "transactionStatus": "Rollback Complete"
    }
```

Llamadas a la API de datos de Amazon RDS desde una aplicación Python

Puede llamar a la API de datos de Amazon RDS (API de datos) desde una aplicación Python.

En los ejemplos siguientes se usa el AWS SDK para Python (Boto). Para obtener más información acerca de Boto, consulte la documentación de AWS SDK para Python (Boto 3).

En cada ejemplo, sustituya el nombre de recurso de Amazon (ARN) del clúster de base de datos por el ARN de su clúster de base de datos de Aurora. Reemplace también el ARN del secreto por el ARN del secreto de Secrets Manager que permite obtener acceso al clúster de base de datos.

Temas

- Ejecución de una consulta SQL
- Ejecución de una instrucción SQL DML
- Ejecución de una transacción SQL

Ejecución de una consulta SQL

Puede ejecutar una instrucción `SELECT` y recopilar los resultados con una aplicación Python.

En el ejemplo siguiente, se ejecuta una consulta SQL.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

response1 = rdsData.execute_statement(
            resourceArn = cluster_arn,
            secretArn = secret_arn,
            database = 'mydb',
            sql = 'select * from employees limit 3')

print (response1['records'])
[
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'ROSALEZ'
        },
        {
            'stringValue': 'ALEJANDRO'
        },
        {
            'stringValue': '2016-02-15 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'DOE'
        },
        {
            'stringValue': 'JANE'
        },
        {
            'stringValue': '2014-05-09 04:34:33.0'
        }
```

```
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'STILES'
        },
        {
            'stringValue': 'JOHN'
        },
        {
            'stringValue': '2017-09-20 04:34:33.0'
        }
    ]
]
```

Ejecución de una instrucción SQL DML

Puede ejecutar una instrucción de lenguaje de manipulación de datos (DML) para insertar, actualizar o eliminar datos en su base de datos. También puede utilizar parámetros en instrucciones DML.

> **Important**
>
> Si una llamada no forma parte de una transacción porque no incluye el parámetro `transactionID`, los cambios que se generen a partir de la llamada se confirmarán automáticamente.

En el ejemplo siguiente se ejecuta una instrucción SQL de inserción y se utilizan parámetros.

```
import boto3

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

rdsData = boto3.client('rds-data')


param1 = {'name':'firstname', 'value':{'stringValue': 'JACKSON'}}
param2 = {'name':'lastname', 'value':{'stringValue': 'MATEO'}}
paramSet = [param1, param2]
```

```
response2 = rdsData.execute_statement(resourceArn=cluster_arn,
                                      secretArn=secret_arn,
                                      database='mydb',
                                      sql='insert into employees(first_name, last_name)
 VALUES(:firstname, :lastname)',
                                      parameters = paramSet)

print (response2["numberOfRecordsUpdated"])
```

Ejecución de una transacción SQL

Puede iniciar una transacción SQL, ejecutar una o varias instrucciones SQL y luego confirmar los cambios con una aplicación Python.

> **Important**
>
> El tiempo de la transacción se agota si no hay llamadas que usen su ID de transacción en un período de tres minutos. Si una transacción agota su tiempo antes de que se confirme, se revertirá automáticamente.
> Si no especifica un ID de transacción, los cambios que se generen a partir de la llamada se confirmarán automáticamente.

En el ejemplo siguiente se ejecuta una transacción SQL que inserta una fila en una tabla.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

tr = rdsData.begin_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb')

response3 = rdsData.execute_statement(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
```

```
     database = 'mydb',
     sql = 'insert into employees(first_name, last_name) values('XIULAN', 'WANG')',
     transactionId = tr['transactionId'])

cr = rdsData.commit_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     transactionId = tr['transactionId'])

cr['transactionStatus']
'Transaction Committed'

response3['numberOfRecordsUpdated']
1
```

> **Note**
>
> Si ejecuta una instrucción de lenguaje de definición de datos (DDL), recomendamos que siga ejecutando la instrucción después de que se agote el tiempo de la llamada. Cuando se termina una instrucción DDL antes de que acabe de ejecutarse, pueden generarse errores y es posible que las estructuras de datos se dañen. Para seguir ejecutando una instrucción después de que una llamada supere el intervalo de tiempo de espera de la API de datos de RDS de 45 segundos, ajuste el parámetro `continueAfterTimeout` en `true`.

Llamadas a la API de datos de Amazon RDS desde una aplicación Java

Puede llamar a la API de datos de Amazon RDS (API de datos) desde una aplicación Java.

En los ejemplos siguientes se usa el AWS SDK para Java. Para obtener más información, consulte AWS SDK para Java Developer Guide.

En cada ejemplo, sustituya el nombre de recurso de Amazon (ARN) del clúster de base de datos por el ARN de su clúster de base de datos de Aurora. Reemplace también el ARN del secreto por el ARN del secreto de Secrets Manager que permite obtener acceso al clúster de base de datos.

Temas

- Ejecución de una consulta SQL
- Ejecución de una transacción SQL

- Ejecución de una operación SQL por lotes

Ejecución de una consulta SQL

Puede ejecutar una instrucción SELECT y recopilar los resultados con una aplicación Java.

En el ejemplo siguiente, se ejecuta una consulta SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementResult;
import com.amazonaws.services.rdsdata.model.Field;

import java.util.List;

public class FetchResultsExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    ExecuteStatementRequest request = new ExecuteStatementRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb")
            .withSql("select * from mytable");

    ExecuteStatementResult result = rdsData.executeStatement(request);

    for (List<Field> fields: result.getRecords()) {
      String stringValue = fields.get(0).getStringValue();
      long numberValue = fields.get(1).getLongValue();

      System.out.println(String.format("Fetched row: string = %s, number = %d",
 stringValue, numberValue));
    }
  }
```

```
}
```

Ejecución de una transacción SQL

Puede iniciar una transacción SQL, ejecutar una o varias instrucciones SQL y luego confirmar los cambios con una aplicación Java.

> **Important**
>
> El tiempo de la transacción se agota si no hay llamadas que usen su ID de transacción en un período de tres minutos. Si una transacción agota su tiempo antes de que se confirme, se revertirá automáticamente.
>
> Si no especifica un ID de transacción, los cambios que se generen a partir de la llamada se confirmarán automáticamente.

En el ejemplo siguiente, se ejecuta una transacción SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BeginTransactionRequest;
import com.amazonaws.services.rdsdata.model.BeginTransactionResult;
import com.amazonaws.services.rdsdata.model.CommitTransactionRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;

public class TransactionExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BeginTransactionRequest beginTransactionRequest = new BeginTransactionRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb");
    BeginTransactionResult beginTransactionResult =
 rdsData.beginTransaction(beginTransactionRequest);
```

```
    String transactionId = beginTransactionResult.getTransactionId();

    ExecuteStatementRequest executeStatementRequest = new ExecuteStatementRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table VALUES ('hello world!')");
    rdsData.executeStatement(executeStatementRequest);

    CommitTransactionRequest commitTransactionRequest = new CommitTransactionRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN);
    rdsData.commitTransaction(commitTransactionRequest);
  }
}
```

> **Note**
>
> Si ejecuta una instrucción de lenguaje de definición de datos (DDL), recomendamos que siga ejecutando la instrucción después de que se agote el tiempo de la llamada. Cuando se termina una instrucción DDL antes de que acabe de ejecutarse, pueden generarse errores y es posible que las estructuras de datos se dañen. Para seguir ejecutando una instrucción después de que una llamada supere el intervalo de tiempo de espera de la API de datos de RDS de 45 segundos, ajuste el parámetro `continueAfterTimeout` en `true`.

Ejecución de una operación SQL por lotes

Puede ejecutar operaciones de inserción y actualización masivas en una matriz de datos, con una aplicación Java. Puede ejecutar una instrucción DML con una matriz de conjuntos de parámetros.

> **Important**
>
> Si no especifica un ID de transacción, los cambios que se generen a partir de la llamada se confirmarán automáticamente.

En el siguiente ejemplo se ejecuta una operación de inserción por lotes.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BatchExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.Field;
import com.amazonaws.services.rdsdata.model.SqlParameter;

import java.util.Arrays;

public class BatchExecuteExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
      AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BatchExecuteStatementRequest request = new BatchExecuteStatementRequest()
            .withDatabase("test")
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table2 VALUES (:string, :number)")
            .withParameterSets(Arrays.asList(
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("Hello")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(1L))
                    ),
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("World")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(2L))
                    )
            ));

    rdsData.batchExecuteStatement(request);
  }
}
```

Control del comportamiento del tiempo de espera de la API de datos

Todas las llamadas a la API de datos son síncronas. Imaginemos que realiza una operación de API de datos que ejecuta una instrucción de SQL como `INSERT` o`CREATE TABLE`. Si la llamada a la API de datos se devuelve correctamente, el procesamiento de SQL finaliza cuando se devuelve la llamada.

De forma predeterminada, la API de datos cancela una operación y devuelve un error de tiempo de espera si la operación no termina de procesarse en 45 segundos. En ese caso, los datos no se insertan, la tabla no se crea, etc.

Puede usar la API de datos para realizar operaciones de larga duración que no se puedan completar en 45 segundos. Si espera que una operación, como una operación masiva de `INSERT` o DDL en una tabla grande, tarde más de 45 segundos, puede especificar el parámetro `continueAfterTimeout` para la operación `ExecuteStatement`. La aplicación sigue recibiendo el error de tiempo de espera. Sin embargo, la operación continúa ejecutándose y no se cancela. Para ver un ejemplo, consulta Ejecución de una transacción SQL.

Si el AWS SDK de su lenguaje de programación tiene su propio tiempo de espera para las llamadas a la API o las conexiones de sockets HTTP, asegúrese de que todos esos periodos de tiempo de espera sean superiores a 45 segundos. En algunos SDK, el tiempo de espera es inferior a 45 segundos de forma predeterminada. Recomendamos ajustar cualquier periodo de tiempo de espera específico del SDK o específico del cliente en al menos un minuto. De este modo, se evita la posibilidad de que la aplicación reciba un error de tiempo de espera mientras la operación de la API de datos se complete correctamente. De esta forma, puede estar seguro de si desea volver a intentar la operación o no.

Por ejemplo, supongamos que el SDK devuelve un error de tiempo de espera a su aplicación, pero la operación de la API de datos sigue completándose dentro del intervalo de tiempo de espera de la API de datos. En ese caso, volver a intentar la operación podría insertar datos duplicados o producir resultados incorrectos. Es posible que el SDK vuelva a intentar la operación automáticamente, lo que provoca datos incorrectos sin que la aplicación realice ninguna acción.

El intervalo de tiempo de espera es especialmente importante para el SDK de Java 2. En ese SDK, el tiempo de espera de la llamada a la API y el tiempo de espera del socket HTTP es de 30 segundos de forma predeterminada. En este ejemplo se muestra cómo ajustar esos tiempos de espera en un valor superior:

```
public RdsDataClient createRdsDataClient() {
```

```
    return RdsDataClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .httpClientBuilder(createHttpClientBuilder())
        .credentialsProvider(defaultCredentialsProvider()) // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return ApacheHttpClient.builder() // Change this to your desired HttpClient
        .socketTimeout(Duration.ofSeconds(60));
}
```

A continuación, se muestra un ejemplo equivalente con el cliente de datos asíncrono:

```
public static RdsDataAsyncClient createRdsDataAsyncClient() {
    return RdsDataAsyncClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .credentialsProvider(defaultCredentialsProvider())  // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return NettyNioAsyncHttpClient.builder() // Change this to your desired
 AsyncHttpClient
        .readTimeout(Duration.ofSeconds(60));
}
```

Usar la biblioteca de cliente de Java para la API de datos de RDS

Puede descargar y utilizar una biblioteca de cliente de Java para la API de datos de RDS (API de datos). La biblioteca de cliente de Java ofrece un método alternativo para utilizar la API de datos. Al utilizar esta biblioteca, puede asignar las clases del lado del cliente a las solicitudes y respuestas de la API de datos. Esta compatibilidad con la asignación puede facilitar la integración con algunos tipos de Java específicos, como `Date`, `Time` y `BigDecimal`.

Descarga de la biblioteca de cliente de Java para la API de datos

La biblioteca de cliente Java para la API de datos es de código abierto en GitHub en la siguiente ubicación:

https://github.com/awslabs/rds-data-api-client-library-java

Puede crear la biblioteca de forma manual a partir de los archivos de origen, pero la práctica recomendada es utilizar la biblioteca mediante la administración de dependencias de Apache Maven. Agregue la siguiente dependencia a su archivo Maven POM.

Para la versión 2.x, compatible con AWS SDK 2.x, utilice lo siguiente:

```
<dependency>
   <groupId>software.amazon.rdsdata</groupId>
   <artifactId>rds-data-api-client-library-java</artifactId>
   <version>2.0.0</version>
</dependency>
```

Para la versión 1.x, compatible con AWS SDK 1.x, utilice lo siguiente:

```
<dependency>
    <groupId>software.amazon.rdsdata</groupId>
    <artifactId>rds-data-api-client-library-java</artifactId>
    <version>1.0.8</version>
</dependency>
```

Ejemplos de la biblioteca de cliente de Java

A continuación, puede encontrar algunos ejemplos habituales del uso de la biblioteca de cliente de Java para la API de datos. En estos ejemplos se asume que tiene una tabla `accounts` con dos columnas: `accountId` y `name`. También tiene el siguiente objeto de transferencia de datos (DTO).

```
public class Account {
    int accountId;
    String name;
    // getters and setters omitted
}
```

La biblioteca de cliente le permite transferir los DTO como parámetros de entrada. En el siguiente ejemplo se muestra cómo los DTO del cliente se asignan a los conjuntos de parámetros de entrada.

```
var account1 = new Account(1, "John");
var account2 = new Account(2, "Mary");
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParamSets(account1, account2)
         .execute();
```

En algunos casos, es más fácil trabajar con valores simples como parámetros de entrada. Puede hacerlo mediante la siguiente sintaxis:

```
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParameter("accountId", 3)
         .withParameter("name", "Zhang")
         .execute();
```

A continuación se muestra otro ejemplo que funciona con valores simples como parámetros de entrada.

```

client.forSql("INSERT INTO accounts(accountId, name) VALUES(?, ?)", 4, "Carlos")
         .execute();

```

La biblioteca de cliente proporciona un mapeo automático a los DTO cuando se devuelve un resultado. En los siguientes ejemplos se muestra cómo se asigna el resultado a los DTO.

```
List<Account> result = client.forSql("SELECT * FROM accounts")
          .execute()
          .mapToList(Account.class);

Account result = client.forSql("SELECT * FROM accounts WHERE account_id = 1")
          .execute()
```

```
          .mapToSingle(Account.class);
```

En muchos casos, el conjunto de resultados de la base de datos contiene un solo valor. Para simplificar la recuperación de estos resultados, la biblioteca cliente ofrece la siguiente API:

```
int numberOfAccounts = client.forSql("SELECT COUNT(*) FROM accounts")
          .execute()
          .singleValue(Integer.class);
```

> **Note**
>
> La función `mapToList` convierte un conjunto de resultados SQL en una lista de objetos definida por el usuario. No se admite el uso de la instrucción `.withFormatRecordsAs(RecordsFormatType.JSON)` en una llamada de `ExecuteStatement` a la biblioteca cliente Java, porque tiene el mismo propósito. Para obtener más información, consulte Procesamiento de resultados de consultas de API de datos de Amazon RDS en formato JSON.

Procesamiento de resultados de consultas de API de datos de Amazon RDS en formato JSON

Cuando se llama a la operación `ExecuteStatement`, se puede elegir que los resultados de la consulta se devuelvan como cadena en formato JSON. Esto permite utilizar las capacidades de análisis JSON de su lenguaje de programación para interpretar el conjunto de resultados y volver a darle formato. Hacerlo puede ayudar a evitar escribir código adicional para pasar por el conjunto de resultados e interpretar el valor de cada columna.

Para solicitar el conjunto de resultados en formato JSON, es preciso pasar el parámetro opcional `formatRecordsAs` con un valor `JSON`. El conjunto de resultados con formato JSON se devuelve en el campo `formattedRecords` de la estructura `ExecuteStatementResponse`.

La acción `BatchExecuteStatement` no devuelve un conjunto de resultados. Por lo tanto, la opción JSON no se aplica a esa acción.

Para personalizar las claves de la estructura hash JSON, defina alias de columna en el conjunto de resultados. Puede hacerlo mediante la cláusula `AS` de la lista de columnas de la consulta SQL.

Puede hacer uso de la capacidad JSON para facilitar la lectura del conjunto de resultados y asignar su contenido a marcos específicos de lenguaje. Dado que el volumen del conjunto de resultados codificado en ASCII es mayor que la representación predeterminada, puede elegir la representación predeterminada para consultas que devuelvan un gran número de filas o valores de columna grandes que consuman más memoria de la que está disponible para la aplicación.

Temas

- Recuperación de resultados de consultas en formato JSON
- Asignación de tipos de datos
- Solución de problemas
- Ejemplos

Recuperación de resultados de consultas en formato JSON

Para recibir el conjunto de resultados como una cadena JSON, incluya `.withFormatRecordsAs(RecordsFormatType.JSON)` en la llamada a `ExecuteStatement`. El valor devuelto vuelve como cadena JSON en el campo `formattedRecords`. En este caso, `columnMetadata` es `null`. Las etiquetas de columna son las claves del objeto que representa cada fila. Estos nombres de columna se repiten para cada fila del conjunto de resultados. Los valores de columna son cadenas entre comillas, valores numéricos o valores especiales que representan `true`, `false` o `null`. Los metadatos de columna, como las restricciones de longitud y el tipo preciso de números y cadenas, no se conservan en la respuesta JSON.

Si omite la llamada `.withFormatRecordsAs()` o especifica un parámetro de `NONE`, el conjunto de resultados se devuelve en formato binario mediante los campos `Records` y `columnMetadata`.

Asignación de tipos de datos

Los valores SQL del conjunto de resultados se asignan a un conjunto más pequeño de tipos JSON. Los valores se representan en JSON como cadenas, números y ciertas constantes especiales, como `true`, `false` y `null`. Puede convertir estos valores en variables en su aplicación mediante tipado fuerte o débil según corresponda en el lenguaje de programación.

Tipo de datos JDBC	Tipos de datos de JSON
INTEGER, TINYINT, SMALLINT, BIGINT	Número de forma predeterminada. Cadena si la opción LongReturnType está configurada en STRING.
FLOAT, REAL, DOUBLE	Número
DECIMAL	Cadena de forma predeterminada. Número si la opción DecimalReturnType está configurada en DOUBLE_OR_LONG .
STRING	Cadena
BOOLEAN, BIT	Booleano
BLOB, BINARY, VARBINARY , LONGVARBINARY	Cadena en codificación base64.
CLOB	Cadena
ARRAY	Matriz
NULL	null
Otros tipos (incluidos los tipos relacionados con la fecha y hora)	Cadena

Solución de problemas

La respuesta JSON se limita a 10 megabytes. Si la respuesta supera este límite, el programa recibe un error BadRequestException. En este caso, puede resolver el error mediante una de las siguientes técnicas:

- Reducir el número de filas en el conjunto de resultados. Para ello, añada una cláusula LIMIT. Puede dividir un conjunto de resultados grande en varios más pequeños enviando varias consultas con cláusulas LIMIT y OFFSET.

Si el conjunto de resultados incluye filas filtradas por lógica de aplicación, puede eliminarlas del conjunto de resultados añadiendo más condiciones en la cláusula `WHERE`.

- Reducir el número de columnas en el conjunto de resultados. Para ello, retire elementos de la lista de selección de la consulta.
- Acortar las etiquetas de columna utilizando alias de columna en la consulta. El nombre de cada columna se repite en la cadena JSON para cada fila del conjunto de resultados. Así, un resultado de consulta con nombres de columna largos y muchas filas podría superar el límite de tamaño. En particular, utilice alias de columna para expresiones complicadas para evitar que se repita toda la expresión en la cadena JSON.
- Aunque con SQL puede utilizar alias de columna para producir un conjunto de resultados que tenga más de una columna con el mismo nombre, no puede haber nombres de claves duplicados en JSON. La API de datos de RDS devuelve un error si solicita el conjunto de resultados en formato JSON y hay más de una columna con el mismo nombre. Así pues, asegúrese de que todas las etiquetas de columna tengan nombres únicos.

Ejemplos

Los siguientes ejemplos de Java muestran cómo llamar a `ExecuteStatement` con la respuesta como cadena con formato JSON y, a continuación, interpretar el conjunto de resultados. Sustituya los valores apropiados por los parámetros *`databaseName`*, *`secretStoreArn`* y *`clústerArn`*.

En el siguiente ejemplo de Java se muestra una consulta que devuelve un valor numérico decimal en el conjunto de resultados. Las llamadas `assertThat` prueban que los campos de la respuesta tengan las propiedades esperadas según las reglas de los conjuntos de resultados JSON.

Este ejemplo funciona con el siguiente esquema y datos de ejemplo:

```
create table test_simplified_json (a float);
insert into test_simplified_json values(10.0);
```

```
public void JSON_result_set_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
```

```
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request);
}
```

El valor del campo `formattedRecords` del programa anterior es:

```
[{"a":10.0}]
```

Los campos `Records` y `ColumnMetadata` de la respuesta son nulos debido a la presencia del conjunto de resultados JSON.

En el siguiente ejemplo de Java se muestra una consulta que devuelve un valor numérico entero en el conjunto de resultados. En el ejemplo se llama a `getFormattedRecords` para devolver solo la cadena con formato JSON e ignorar los demás campos de respuesta en blanco o nulos. En el ejemplo se deserializa el resultado en una estructura que representa una lista de registros. Cada registro tiene campos cuyos nombres corresponden a los alias de columna del conjunto de resultados. Esta técnica simplifica el código que analiza el conjunto de resultados. La aplicación no tiene que recorrer las filas y las columnas del conjunto de resultados y convertir cada valor al tipo adecuado.

Este ejemplo funciona con el siguiente esquema y datos de ejemplo:

```
create table test_simplified_json (a int);
insert into test_simplified_json values(17);
```

```
public void JSON_deserialization_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request)
      .getFormattedRecords();

/* Turn the result set into a Java object, a list of records.
   Each record has a field 'a' corresponding to the column
   labelled 'a' in the result set. */
    private static class Record { public int a; }
    var recordsList = new ObjectMapper().readValue(
```

```
        response, new TypeReference<List<Record>>() {
        });
}
```

El valor del campo `formattedRecords` del programa anterior es:

```
[{"a":17}]
```

Para recuperar la columna a de la fila de resultados 0, la aplicación haría referencia a `recordsList.get(0).a`.

En cambio, en el siguiente ejemplo de Java se muestra el tipo de código necesario para construir una estructura de datos que contenga el conjunto de resultados cuando no se utilice el formato JSON. En este caso, cada fila del conjunto de resultados contiene campos con información sobre un solo usuario. La creación de una estructura de datos para representar el conjunto de resultados requiere recorrer las filas en bucle. Para cada fila, el código recupera el valor de cada campo, hace la conversión de tipo adecuada y asigna el resultado al campo correspondiente del objeto que representa la fila. A continuación, el código añade el objeto que representa a cada usuario a la estructura de datos que representa todo el conjunto de resultados. Si la consulta se modificó para reordenar, agregar o quitar campos del conjunto de resultados, el código de la aplicación tendría que cambiar también.

```
/* Verbose result-parsing code that doesn't use the JSON result set format */
for (var row: response.getRecords()) {
    var user = User.builder()
      .userId(row.get(0).getLongValue())
      .firstName(row.get(1).getStringValue())
      .lastName(row.get(2).getStringValue())
      .dob(Instant.parse(row.get(3).getStringValue()))
      .build();
    result.add(user);
  }
```

Los siguientes valores de ejemplo muestran los valores del campo `formattedRecords` para conjuntos de resultados con diferentes números de columnas, alias de columna y tipos de datos de columnas.

Si el conjunto de resultados incluye varias filas, cada fila se representa como un objeto que es un elemento de matriz. Cada columna del conjunto de resultados se convierte en una clave dentro del objeto. Las claves se repiten para cada fila del conjunto de resultados. Por lo tanto, para los

conjuntos de resultados que constan de varias filas y columnas, es posible que deba definir alias de columna cortos para evitar superar el límite de longitud de toda la respuesta.

Este ejemplo funciona con el siguiente esquema y datos de ejemplo:

```
create table sample_names (id int, name varchar(128));
insert into sample_names values (0, "Jane"), (1, "Mohan"), (2, "Maria"), (3, "Bruce"),
 (4, "Jasmine");
```

```
[{"id":0,"name":"Jane"},{"id":1,"name":"Mohan"},
{"id":2,"name":"Maria"},{"id":3,"name":"Bruce"},{"id":4,"name":"Jasmine"}]
```

Si una columna del conjunto de resultados se define como expresión, el texto de la expresión se convierte en la clave JSON. Así pues, suele ser conveniente definir un alias de columna descriptivo para cada expresión de la lista de selección de la consulta. Por ejemplo, la siguiente consulta incluye expresiones como llamadas a funciones y operaciones aritméticas en su lista de selección.

```
select count(*), max(id), 4+7 from sample_names;
```

Esas expresiones se pasan al conjunto de resultados JSON como claves.

```
[{"count(*)":5,"max(id)":4,"4+7":11}]
```

Añadir columnas AS con etiquetas descriptivas simplifica la interpretación de las claves en el conjunto de resultados JSON.

```
select count(*) as rows, max(id) as largest_id, 4+7 as addition_result from
 sample_names;
```

Con la consulta SQL revisada, se utilizan como nombres de clave las etiquetas de columna definidas por las cláusulas AS.

```
[{"rows":5,"largest_id":4,"addition_result":11}]
```

El valor de cada par clave-valor de la cadena JSON puede ser una cadena entre comillas. La cadena podría contener caracteres unicode. Si la cadena contiene secuencias de escape o caracteres '' o \, esos personajes van precedidos de caracteres de escape de barra invertida. Los siguientes ejemplos de cadenas JSON muestran estas posibilidades. Por ejemplo, el resultado

`string_with_escape_sequences` contiene el valor de retroceso para caracteres especiales, nueva línea, retorno de carro, sangría, salto de página y \.

```
[{"quoted_string":"hello"}]
[{"unicode_string":"####"}]
[{"string_with_escape_sequences":"\b \n \r \t \f \\ '"}]
```

El valor de cada par clave-valor de la cadena JSON puede también representar un número. El número puede ser un entero, un valor de coma flotante, un valor negativo o un valor representado como notación exponencial. Los siguientes ejemplos de cadenas JSON muestran estas posibilidades.

```
[{"integer_value":17}]
[{"float_value":10.0}]
[{"negative_value":-9223372036854775808,"positive_value":9223372036854775807}]
[{"very_small_floating_point_value":4.9E-324,"very_large_floating_point_value":1.79769313486231
```

Los valores booleanos y nulos se representan con las palabras clave especiales sin comillas `true`, `false` y `null`. Los siguientes ejemplos de cadenas JSON muestran estas posibilidades.

```
[{"boolean_value_1":true,"boolean_value_2":false}]
[{"unknown_value":null}]
```

Si selecciona un valor de tipo BLOB, el resultado se representa en la cadena JSON como un valor codificado en base64. Para convertir el valor a su representación original, puede utilizar la función de descodificación adecuada en el lenguaje de la aplicación. Por ejemplo, en Java se llama a la función `Base64.getDecoder().decode()`. El siguiente ejemplo muestra el resultado de seleccionar un valor BLOB de `hello world` y devuelve el conjunto de resultados como una cadena JSON.

```
[{"blob_column":"aGVsbG8gd29ybGQ="}]
```

El siguiente ejemplo de Python muestra cómo acceder a los valores del resultado de una llamada a la función de Python `execute_statement`. El conjunto de resultados es un valor de cadena en el campo `response['formattedRecords']`. El código convierte la cadena JSON en una estructura de datos llamando a la función `json.loads`. A continuación, cada fila del conjunto de resultados es un elemento de lista dentro de la estructura de datos y, dentro de cada fila, puede hacer referencia a cada campo del conjunto de resultados por nombre.

```
import json
```

```
result = json.loads(response['formattedRecords'])
print (result[0]["id"])
```

En el siguiente ejemplo de JavaScript se muestra cómo acceder a los valores del resultado de una llamada a la función `executeStatement` de JavaScript. El conjunto de resultados es un valor de cadena en el campo `response.formattedRecords`. El código convierte la cadena JSON en una estructura de datos llamando a la función `JSON.parse`. A continuación, cada fila del conjunto de resultados es un elemento de matriz dentro de la estructura de datos y, dentro de cada fila, puede hacer referencia a cada campo del conjunto de resultados por nombre.

```
<script>
    const result = JSON.parse(response.formattedRecords);
    document.getElementById("display").innerHTML = result[0].id;
</script>
```

Solución de problemas de la API de datos de Amazon RDS

Use las siguientes secciones, tituladas con mensajes de error comunes, para ayudar a solucionar problemas que tenga con la API de datos de Amazon RDS (API de datos).

Temas

- No se ha encontrado la transacción <transaction_ID>
- El paquete de la consulta es demasiado grande
- Límite de tamaño superado de respuesta de base de datos
- HttpEndpoint no está habilitado para el clúster <clúster_ID>
- DatabaseErrorException: Transaction is still running a query
- Excepción de resultado no compatible
- Las instrucciones múltiples no son compatibles
- El parámetro de esquema no es compatible
- Problemas de conectividad de IPv6

No se ha encontrado la transacción <transaction_ID>

En este caso, el ID de transacción especificado en la API de datos no se ha podido encontrar. La causa de este problema se anexa al mensaje de error y es uno de los siguientes:

- La transacción puede haber caducado.

  Asegúrese también de que cada llamada transaccional se realice antes de que transcurran tres minutos con respecto al uso anterior.

  También es posible que el ID de transacción especificado no se creara con una llamada a BeginTransaction. Asegúrese de que su llamada tenga un ID de transacción válido.
- Una llamada anterior dio lugar a la finalización de la transacción.

  La transacción ya la ha finalizado su `CommitTransaction` o la llamada a `RollbackTransaction`.
- La transacción se ha anulado debido a un error de una llamada anterior.

  Comprueba si sus llamadas anteriores han arrojado excepciones.

Para obtener más información acerca de la ejecución de transacciones, consulte Llamadas a la API de datos de Amazon RDS.

El paquete de la consulta es demasiado grande

En este caso, el conjunto de resultados devuelto para una fila era demasiado grande. El límite de tamaño de la API de datos es 64 KB por fila en el conjunto de resultados devuelto por la base de datos.

Para solventar este problema, asegúrese de que cada fila de un conjunto de resultados sea de 64 KB o menos.

Límite de tamaño superado de respuesta de base de datos

En este caso, el tamaño del conjunto de resultados devuelto por la base de datos era demasiado grande. El límite de la API de datos es 1 MB en el conjunto de resultados devuelto por la base de datos.

Para resolver este problema, asegúrese de que las llamada a la API de datos devuelvan 1 MiB de datos o menos. Si necesita devolver más de 1 MiB, puede usar varias llamadas `ExecuteStatement` con la cláusula `LIMIT` en la consulta.

Para obtener más información acerca de la cláusula `LIMIT`, consulte SELECT Syntax en la documentación de MySQL.

HttpEndpoint no está habilitado para el clúster <clúster_ID>

Compruebe las siguientes posibles causas de este problema:

- El clúster de base de datos de Aurora no admite la API de datos. Para obtener información sobre los tipos de clústeres de bases de datos compatibles con la API de datos de RDS, consulte the section called “Disponibilidad de regiones y versiones para la API de datos de Amazon RDS”.
- La API de datos no está habilitada para el clúster de base de datos de Aurora. Para utilizar la API de datos con un clúster de base de datos de Aurora, la API de datos debe estar habilitada para el clúster de base de datos. Para obtener más información sobre la habilitación de la API de datos, consulte Habilitación de la API de datos de Amazon RDS.
- Se cambió el nombre del clúster de base de datos después de que la API de datos se habilitara para él. En ese caso, desactive la API de datos de ese clúster y vuelva a habilitarla.
- El ARN que ha especificado no coincide exactamente con el ARN del clúster. Compruebe que el ARN devuelto de otro origen o creado por lógica de programa coincida exactamente con el ARN del clúster. Por ejemplo, asegúrese de que el ARN que utiliza tenga la mayúscula o minúscula correcta para todos los caracteres alfabéticos.

DatabaseErrorException: Transaction is still running a query

Si su aplicación envía una solicitud con un ID de transacción y esa transacción está procesando otra solicitud en ese momento, la API de datos devuelve este error a la aplicación inmediatamente. Esta condición puede producirse si la aplicación realiza solicitudes asíncronas con un mecanismo como el de las “promesas” en Javascript.

Para solucionar este problema, espere a que finalice la solicitud anterior y, a continuación, vuelva a intentarlo. Puede volver a intentarlo hasta que el error deje de producirse o hasta que la aplicación reciba algún tipo de error diferente.

Esta condición puede ocurrir con la API de datos para las instancias de Aurora Serverless v2 y las aprovisionadas. En la API de datos de Aurora Serverless v1, las solicitudes subsiguientes para el mismo ID de transacción esperan automáticamente a que finalice la solicitud anterior. Sin embargo, este comportamiento antiguo podría provocar que se agoten los tiempos de espera debido a que la solicitud anterior ha tardado demasiado. Si va a migrar una aplicación de la API de datos antigua que realiza solicitudes simultáneas, modifique la lógica de gestión de excepciones para tener en cuenta este nuevo tipo de error.

Excepción de resultado no compatible

La API de datos no admite todos los tipos de datos. Este error se produce al ejecutar una consulta que devuelve un tipo de datos no compatible.

Para evitar este problema, convierta el tipo de datos no compatible a `TEXT`. Por ejemplo:

```
SELECT custom_type::TEXT FROM my_table;
-- OR
SELECT CAST(custom_type AS TEXT) FROM my_table;
```

Las instrucciones múltiples no son compatibles

Las instrucciones múltiples no son compatibles con la API de datos para Aurora Serverless v2 y los clústeres aprovisionados. Intentar ejecutar varias instrucciones en una sola llamada a la API produce este error.

Para ejecutar varias instrucciones, utilice llamadas a la API independientes `ExecuteStatement` o utilice la API `BatchExecuteStatement` para el procesamiento por lotes.

El parámetro de esquema no es compatible

Aurora Serverless v1 ignora en modo silencioso el parámetro de esquema. Sin embargo, Aurora Serverless v2 y los clústeres aprovisionados rechazan explícitamente las llamadas a la API que incluyen el parámetro de esquema.

Para resolver este problema, elimine el parámetro schema de todas las llamadas a la API de datos cuando utilice Aurora Serverless v2 o clústeres aprovisionados.

Problemas de conectividad de IPv6

Si tiene problemas al conectarse a la API de datos mediante puntos de conexión de IPv6, compruebe las siguientes causas posibles:

- La red no admite IPv6: compruebe que la infraestructura de red sea compatible con IPv6 y que el enrutamiento de IPv6 esté configurado correctamente.
- Problemas de resolución de DNS: asegúrese de que el solucionador de DNS pueda resolver los registros AAAA de los puntos de conexión de doble pila (por ejemplo, `rds-data.us-east-1.api.aws`).

- Configuración del grupo de seguridad: actualice las reglas del grupo de seguridad para permitir el tráfico de IPv6 en el puerto 443 (HTTPS). Agregue reglas para los bloques de CIDR de IPv6 (por ejemplo, `::/0` para todas las direcciones IPv6).
- Configuración de las ACL de red: asegúrese de que las ACL de red permitan el tráfico de IPv6 en los puertos necesarios.
- Compatibilidad de biblioteca de cliente: compruebe que las bibliotecas de cliente de HTTP y los AWS SDK admiten la conectividad de IPv6 y de doble pila.
- Configuración de punto de conexión de VPC: si utiliza PrivateLink, asegúrese de que el punto de conexión de VPC esté configurado para admitir IPv6 y de que las subredes asociadas tengan bloques de CIDR de IPv6 asignados.

Para solucionar problemas de conectividad de IPv6:

1. Pruebe la conectividad mediante los puntos de conexión exclusivos de IPv4 (`.amazonaws.com`) para comprobar que el problema es específico de IPv6.
2. Utilice herramientas de diagnóstico de red para verificar la conectividad de IPv6 con los puntos de conexión de doble pila.
3. Compruebe los registros de CloudTrail para ver si hay errores de autenticación o autorización cuando utilice puntos de conexión de IPv6.
4. Compruebe que la aplicación esté configurada correctamente para usar las nuevas URL de punto de conexión de doble pila.

Registro de llamadas a la API de datos de Amazon RDS con AWS CloudTrail

La API de datos de RDS (API de datos) está integrada con AWS CloudTrail, un servicio que proporciona un registro de las acciones del usuario, el rol o un servicio de AWS en la API de datos. CloudTrail captura todas las llamadas a la API para la API de datos como eventos, incluidas las llamadas procedentes de la consola de Amazon RDS y de las llamadas de código a operaciones de la API de datos. Si crea un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, incluidos los eventos de la API de datos. Los datos recopilados por CloudTrail le permitirán determinar mucha información. Esta información incluye la solicitud que se hizo a la API de datos, la dirección IP desde la que se realizó la solicitud, quién hizo la solicitud, cuándo se realizó y otros detalles.

Para obtener más información sobre CloudTrail, consulte la Guía del usuario de AWS CloudTrail.

Trabajar con información de API de datos en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce una actividad compatible (eventos de administración) en la API de datos, dicha actividad se registra en un evento de CloudTrail junto con otros eventos de servicios de AWS en Historial de eventos. Puede ver, buscar y descargar los últimos eventos de administración de la cuenta de AWS. Para obtener más información, consulte Trabajar con el historial de eventos de CloudTrail en la Guía del usuario de AWS CloudTrail.

Para mantener un registro continuo de los eventos de su cuenta de AWS, incluidos los eventos de la API de datos, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. De manera predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. El seguimiento registra los eventos de todas las regiones de AWS en la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte los siguientes temas en la Guía del usuario de AWS CloudTrail:

- Introducción a la creación de registros de seguimiento
- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de registro de CloudTrail de varias regiones y Recepción de archivos de registro de CloudTrail de varias cuentas

CloudTrail registra todas las operaciones de la API de datos, las cuales quedan documentadas en la referencia de la API de servicio de datos de Amazon RDS. Por ejemplo, las llamadas a las operaciones `BatchExecuteStatement`, `BeginTransaction`, `CommitTransaction` y `ExecuteStatement` generan entradas en los archivos de registro de CloudTrail.

Cada entrada de registro o evento contiene información acerca de quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.

- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte Elemento userIdentity de CloudTrail.

Inclusión y exclusión de eventos de la API de datos de un seguimiento de AWS CloudTrail

La mayoría de los usuarios de la API de datos confían en los eventos de un seguimiento de AWS CloudTrail para proporcionar un registro de las operaciones de la API de datos. Los datos de eventos no revelan el nombre de la base de datos, el nombre del esquema ni las instrucciones SQL en las solicitudes a la API de datos. Sin embargo, saber qué usuario realizó un tipo de llamada a un clúster de base de datos específico en un momento dado puede ayudar a detectar patrones de acceso anómalos.

Inclusión de eventos de la API de datos en un seguimiento de AWS CloudTrail

En el caso de Aurora PostgreSQL sin servidor v2 y las bases de datos aprovisionadas, las siguientes operaciones de la API de datos se registran en AWS CloudTrail como eventos de datos. Los eventos de datos son operaciones de API del plano de datos de gran volumen que CloudTrail no registra de forma predeterminada. Se aplican cargos adicionales a los eventos de datos. Para obtener información sobre los precios de CloudTrail, consulte Precios de AWS CloudTrail.

- BatchExecuteStatement
- BeginTransaction
- CommitTransaction
- ExecuteStatement
- RollbackTransaction

Puede utilizar la consola de CloudTrail, AWS CLI o las operaciones de la API de CloudTrail para registrar estas operaciones de la API de datos. En la consola de CloudTrail, elija API de datos de RDS - clúster de base de datos para el tipo de evento de datos. Para obtener más información, consulte Logging data events with the Consola de administración de AWS en la Guía del usuario de AWS CloudTrail.

Con la AWS CLI, ejecute el comando `aws cloudtrail put-event-selectors` para registrar estas operaciones de la API de datos para su seguimiento. Para registrar todos los eventos de la

API de datos en los clústeres de bases de datos, especifique `AWS::RDS::DBCluster` como tipo de recurso. El siguiente ejemplo registra todos los eventos de la API de datos en los clústeres de bases de datos. Para obtener más información, consulte Logging data events with the AWS Command Line Interface en la Guía del usuario de AWS CloudTrail.

```
aws cloudtrail put-event-selectors --trail-name trail_name --advanced-event-selectors \
'{
   "Name": "RDS Data API Selector",
   "FieldSelectors": [
      {
         "Field": "eventCategory",
         "Equals": [
            "Data"
         ]
      },
      {
         "Field": "resources.type",
         "Equals": [
            "AWS::RDS::DBCluster"
         ]
      }
   ]
}'
```

Puede configurar selectores de eventos avanzados para filtrar adicionalmente por los campos `readOnly`, `eventName`, y `resources.ARN`. Para obtener más información sobre estos campos, consulte AdvancedFieldSelector.

Excluir eventos de la API de datos de un seguimiento de AWS CloudTrail (solo Aurora Serverless v1)

Para Aurora Serverless v1, los eventos de la API de datos son eventos de administración. De forma predeterminada, todos los eventos de la API de datos se incluyen en un registro AWS CloudTrail. Sin embargo, dado que la API de datos puede generar un gran número de eventos, es posible que quiera excluir estos eventos de un seguimiento de CloudTrail. La configuración Excluir eventos de la API de datos de Amazon RDS excluye todos los eventos de la API de datos del seguimiento. No puede excluir eventos específicos de la API de datos.

Para excluir eventos de la API de datos de un seguimiento, haga lo siguiente:

- En la consola de CloudTrail, elija la opción de la configuración Exclude Amazon RDS Data API events (Excluir eventos de la API de datos de Amazon RDS) cuando cree un seguimiento o actualice un seguimiento.
- En la API de CloudTrail, utilice la operación PutEventSelectors. Si utiliza selectores de eventos avanzados, puede excluir los eventos de la API de datos configurando el campo `eventSource` de forma distinta a `rdsdata.amazonaws.com`. Si utiliza selectores de eventos básicos, puede excluir los eventos de la API de datos configurando el valor del atributo `ExcludeManagementEventSources` en `rdsdata.amazonaws.com`. Para obtener más información, consulte Registro de eventos con la AWS Command Line Interface, en la Guía del usuario de AWS CloudTrail.

> **Warning**
>
> La exclusión de eventos de la API de datos de un registro de CloudTrail puede ocultar las acciones de la API de datos. Actúe con precaución al conceder a las entidades principales el permiso `cloudtrail:PutEventSelectors` necesario para realizar esta operación.

Puede deshabilitar esta exclusión en cualquier momento cambiando la configuración de la consola o los selectores de eventos de un seguimiento. A continuación, el seguimiento comenzará a registrar eventos de la API de datos. Sin embargo, no puede recuperar los eventos de la API de datos que se produjeran mientras la exclusión estaba en vigor.

Al excluir eventos de la API de datos con la consola o la API, la operación `PutEventSelectors` resultante de la API CloudTrail también se registra en los registros de CloudTrail. Si los eventos de la API de datos no aparecen en los registros de CloudTrail, busque un evento `PutEventSelectors` con el atributo `ExcludeManagementEventSources` establecido en `rdsdata.amazonaws.com`.

Para obtener más información, consulte Registro de eventos de administración para seguimiento en la Guía del usuario de AWS CloudTrail.

Descripción de las entradas del archivo de registro de la API de datos

Un seguimiento es una configuración que permite la entrega de eventos como archivos de registro en un bucket de Amazon S3 que especifique. Los archivos de registro de CloudTrail pueden contener una o varias entradas de registro. Un evento representa una solicitud específica realizada desde un origen y contiene información sobre la acción solicitada, la fecha y la hora de la acción, los

parámetros de la solicitud, etc. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas públicas a la API, por lo que estas no aparecen en ningún orden específico.

Aurora PostgreSQL sin servidor v2 y aprovisionada

En el siguiente ejemplo se muestra una entrada de registro de CloudTrail que ilustra el funcionamiento de `ExecuteStatement` para bases de datos de Aurora PostgreSQL sin servidor v2 y aprovisionadas. En estas bases de datos, todos los eventos de la API de datos son eventos de datos en los que el origen del evento es rdsdataapi.amazonaws.com y el tipo de evento es Rds Data Service.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdataapi.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "Rds Data Service",
```

```
    "recipientAccountId": "123456789012"
}
```

Aurora Serverless v1

El siguiente ejemplo muestra cómo aparece la entrada de registro de CloudTrail del ejemplo anterior para Aurora Serverless v1. Para Aurora Serverless v1, todos los eventos son eventos de administración en los que el origen del evento es rdsdata.amazonaws.com y el tipo de evento es AwsApiCall.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdata.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
```

```
}
```

Supervisión de las consultas de la API de datos de RDS con Información de rendimiento

Si el clúster de Aurora está ejecutando instancias de Aurora Serverless v2 o aprovisionadas, puede usar Información de rendimiento con la API de datos de RDS.

Para obtener más información sobre cómo utilizar Información de rendimiento con Aurora, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

Cómo se representan las consultas de la API de datos de RDS en Información de rendimiento

Con la API de datos, el clúster de Aurora procesa las consultas en función de las llamadas a la API de datos que envía desde la aplicación. La API de datos también ejecuta algunas instrucciones de SQL como parte de su propio funcionamiento interno, como, por ejemplo, la cancelación de las consultas que superan el umbral de tiempo de espera. Ambos tipos de operaciones de SQL se muestran en las estadísticas y los gráficos de Información de rendimiento.

- En el caso de las consultas de API de datos que envíe a un clúster de Aurora, el campo Host del panel de Información de rendimiento se marca como API de datos de RDS. En Aurora PostgreSQL, el campo application_name tiene el valor `rds-data-api`. Busque estas etiquetas cuando analice la carga de la base de datos mediante Hosts principales o Aplicaciones principales como dimensión.
- Todas las consultas internas que la API de datos ejecuta para administrar aspectos de la base de datos, como el grupo de conexiones y los tiempos de espera de las consultas, se anotan con el prefijo API de datos de RDS. Ejemplo: `/* RDS Data API */ select * from my_table;` busca estos prefijos al analizar la carga de la base de datos mediante SQL principal como dimensión. Las instrucciones se anotan con el comentario de SQL `/* RDS Data API */`.

Uso del editor de consultas de Aurora

El editor de consultas de Aurora le permite ejecutar instrucciones SQL en su clúster de base de datos de Aurora a través de la Consola de administración de AWS. Puede ejecutar consultas SQL, instrucciones de manipulación de datos (DML) e instrucciones de definición de datos (DDL). El uso de la interfaz de la consola le permite realizar el mantenimiento de la base de datos, generar informes y realizar experimentos de SQL. Puede evitar configurar la red para conectarse al clúster de base de datos desde un sistema cliente independiente, como una instancia EC2 o un ordenador portátil.

El editor de consultas requiere un clúster de base de datos de Aurora con la API de datos de RDS (API de datos) habilitada. Para obtener información sobre los clústeres de bases de datos que admiten la API de datos y cómo habilitarla, consulte Uso de la API de datos de Amazon RDS. El SQL que puede ejecutar está sujeto a las limitaciones de la API de datos. Para obtener más información, consulte the section called “Limitaciones”.

Disponibilidad del editor de consultas

El editor de consultas está disponible para clústeres de bases de datos de Aurora en los que se utilicen versiones del motor de Aurora MySQL y Aurora PostgreSQL compatibles con la API de datos, y en las Regiones de AWS en las que la API de datos esté disponible. Para obtener más información, consulte Regiones y motores de base de datos Aurora admitidos para API de datos de RDS.

Autorizar el acceso al editor de consultas

Para ejecutar consultas en el editor de consultas, el usuario debe estar autorizado. Puede autorizar a un usuario para que ejecute consultas en el editor de consultas agregando la política `AmazonRDSDataFullAccess`, una política de AWS Identity and Access Management (IAM) predefinida, a dicho usuario.

> **Note**
>
> Asegúrese de utilizar el mismo nombre de usuario y la misma contraseña al crear el usuario de IAM que para el usuario de la base de datos, como el nombre de usuario y la contraseña administrativos. Para obtener más información, consulte Creación de un usuario de IAM en su Cuenta de AWS en la AWS Identity and Access ManagementGuía del usuario de .

También puede crear una política de IAM que conceda acceso al editor de consultas. Tras crear la política, añádala a cada usuario que necesite acceder al editor de consultas.

La siguiente política proporciona los permisos mínimos necesarios para que un usuario acceda al editor de consultas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "QueryEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutResourcePolicy",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:TagResource"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "QueryEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "tag:GetResources",
                "secretsmanager:CreateSecret",
                "secretsmanager:ListSecrets",
                "dbqms:CreateFavoriteQuery",
                "dbqms:DescribeFavoriteQueries",
                "dbqms:UpdateFavoriteQuery",
                "dbqms:DeleteFavoriteQueries",
                "dbqms:GetQueryString",
                "dbqms:CreateQueryHistory",
                "dbqms:UpdateQueryHistory",
                "dbqms:DeleteQueryHistory",
                "dbqms:DescribeQueryHistory",
                "rds-data:BatchExecuteStatement",
```

```
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "*"
        }
    ]
}
```

Para obtener más información acerca de cómo crear una política de IAM, consulte Creación de políticas de IAM en la Guía del usuario de AWS Identity and Access Management.

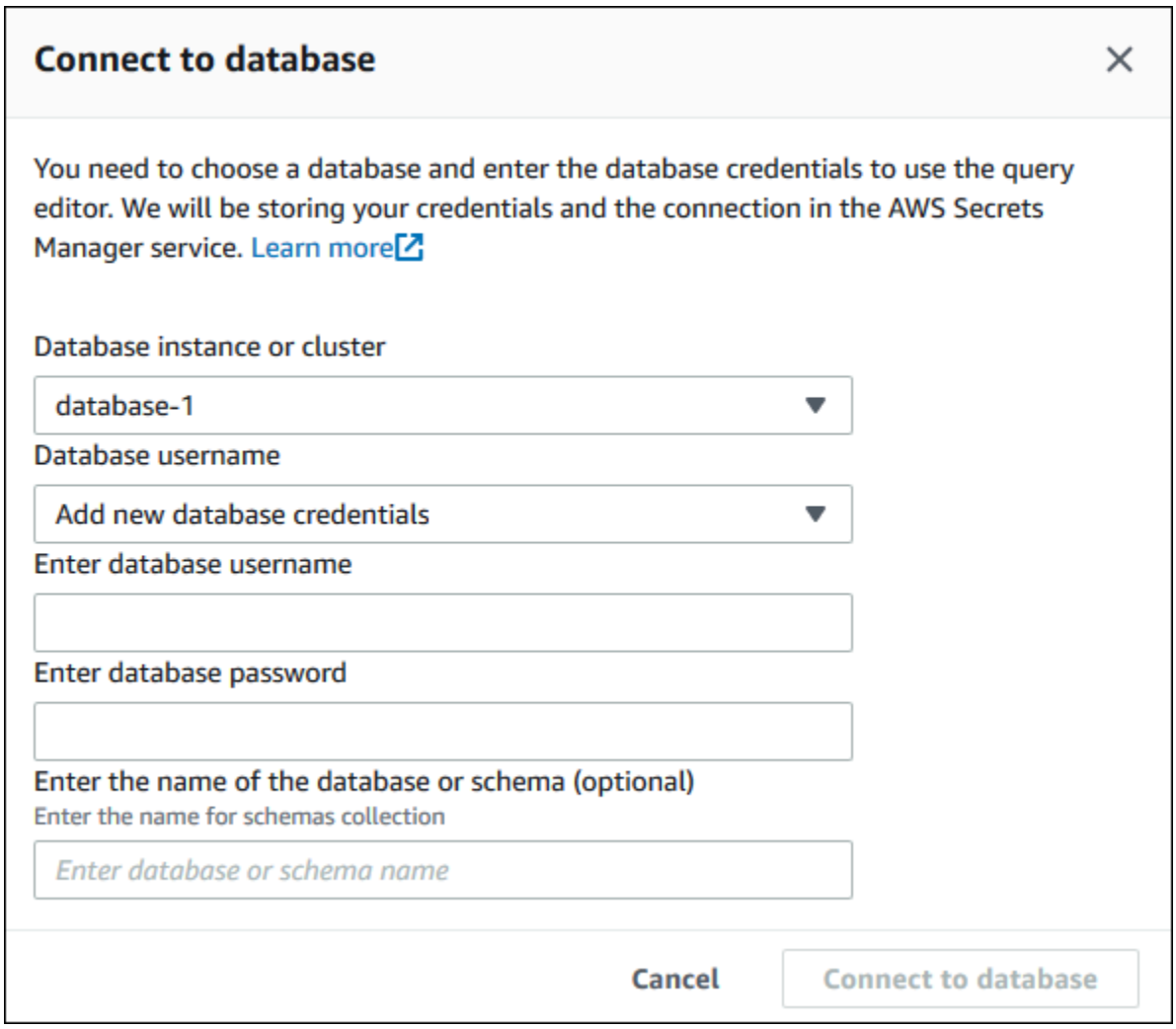
Para obtener información sobre cómo agregar una política de IAM a un usuario, consulte Adición y eliminación de permisos de identidad de IAM en la Guía del usuario de AWS Identity and Access Management.

Ejecución de consultas en el editor de consultas

Puede ejecutar instrucciones SQL en un clúster de base de datos de Aurora en el editor de consultas. El SQL que puede ejecutar está sujeto a las limitaciones de la API de datos. Para obtener más información, consulte the section called “Limitaciones”.

Para ejecutar una consulta en el editor de consultas

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la Consola de administración de AWS, seleccione la Región de AWS en la que ha creado los clústeres de base de datos de Aurora en los que desea realizar la consulta.
3. En el panel de navegación, elija Databases (Bases de datos).
4. Elija el clúster de base de datos de Aurora en el que desea ejecutar consultas SQL.
5. En Actions (Acciones), seleccione Query (Consultar). Si no se ha conectado aún a la base de datos, se abrirá la página Connect to database (Conectar a la base de datos).



6. Introduzca la información siguiente:

   a. En Clúster o instancia de base de datos, elija el clúster de base de datos de Aurora en el que desea ejecutar consultas SQL.

   b. En Database username (Nombre de usuario de base de datos), seleccione el nombre de usuario del usuario de la base de datos al que conectarse o elija Add new database credentials (Añadir nuevas credenciales de base de datos). Si elige Add new database credentials (Añadir nuevas credenciales de base de datos), especifique el nombre de usuario de las nuevas credenciales de base de datos en Enter database username (Introducir nombre de usuario de base de datos).

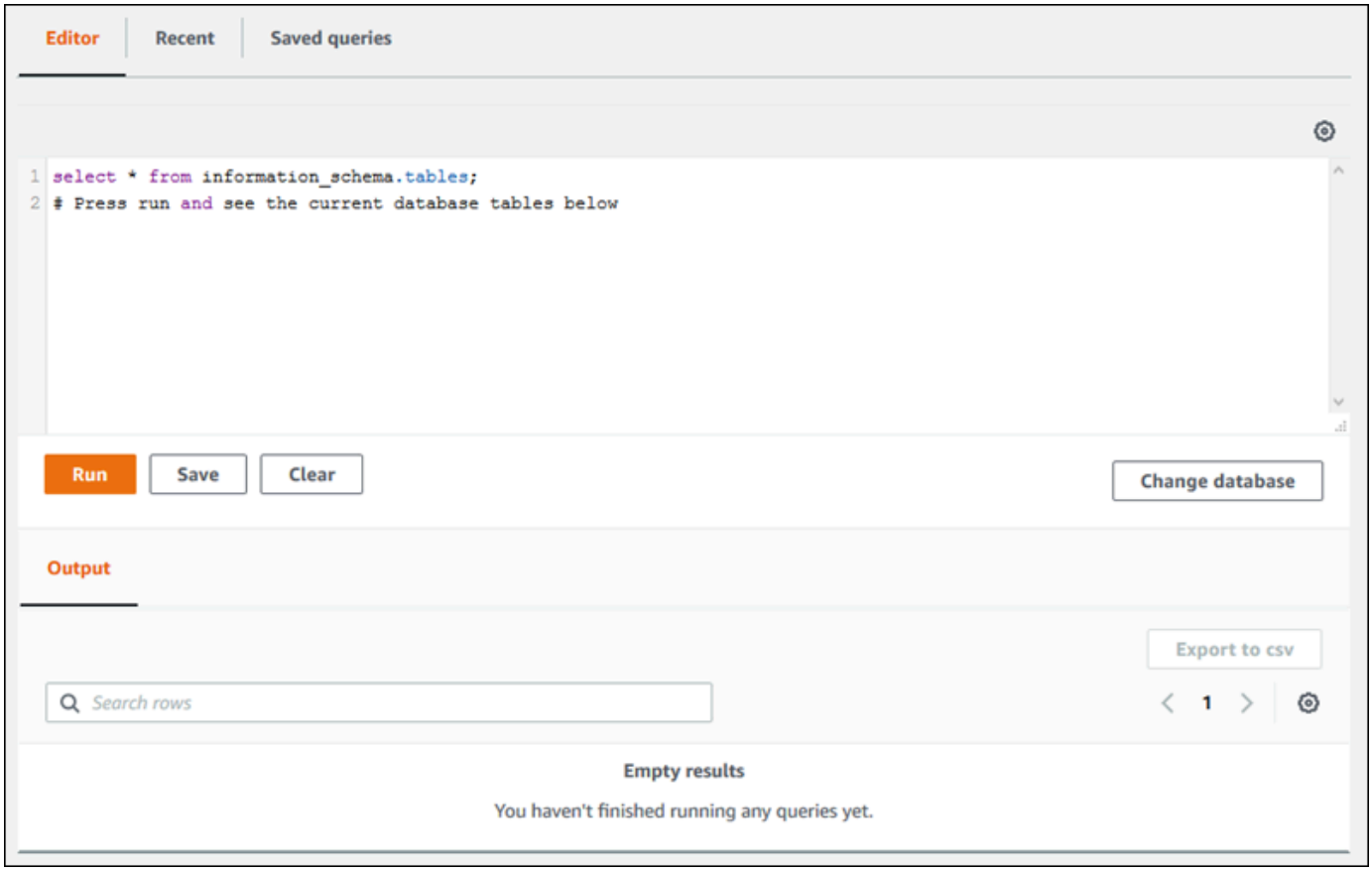
   c. En Enter database password (Introducir contraseña de base de datos), escriba la contraseña para el usuario de la base de datos que ha elegido.

   d. En la última casilla, introduzca el nombre de la base de datos o esquema que quiera usar para el clúster de base de datos de Aurora.

e. Seleccione Connect to database (Conectar a base de datos).

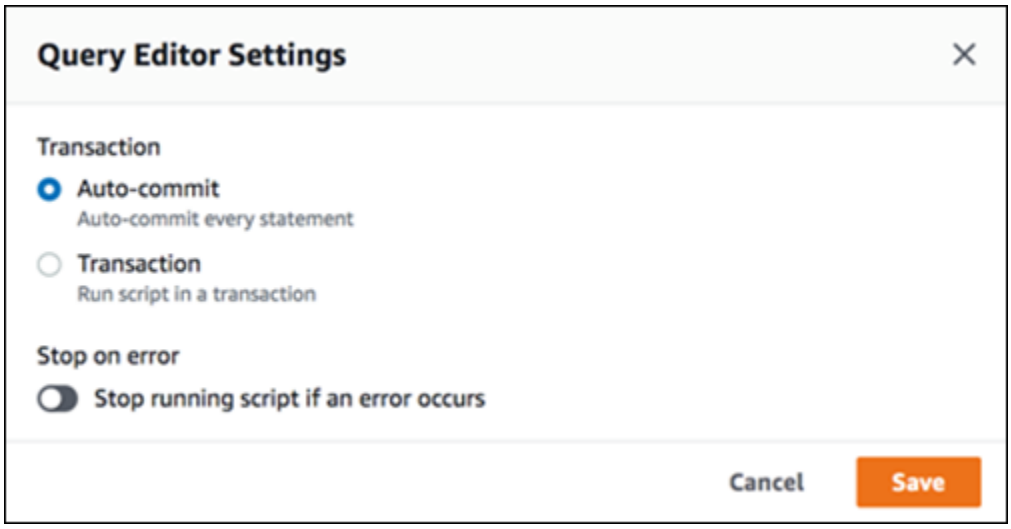
> **Note**
>
> Si su conexión se establece correctamente, la información de conexión y autenticación se almacenará en AWS Secrets Manager. No tendrá que volver a introducir la información de conexión de nuevo.

7. En el editor de consultas, introduzca la consulta SQL que desea ejecutar en la base de datos.



Cada instrucción SQL puede confirmarse automáticamente, o bien puede ejecutar instrucciones SQL en un script, como parte de una transacción. Para controlar este comportamiento, seleccione el icono de engranaje que se encuentra en la ventana de consulta.

Se visualizará la ventana Query Editor Settings (Configuración del editor de consultas).



Si elige Auto-commit (Confirmación automática), todas las instrucciones SQL se confirmarán automáticamente. Si elige Transacción, puede ejecutar un grupo de sentencias en un script. Las instrucciones se confirman automáticamente al final del script a menos que se hayan confirmado explícitamente o se hayan revertido con anterioridad. Por otra parte, puede elegir parar un script que se está ejecutando si se produce un error, habilitando Stop on error (Parar al producirse un error).

> **Note**
>
> En un grupo de instrucciones, las instrucciones de lenguaje de definición de datos (DDL) pueden hacer que las instrucciones de lenguaje de manipulación de datos (DML) anteriores se confirmen. También puede incluir instrucciones `COMMIT` y `ROLLBACK` en un grupo de instrucciones de un script.

Después de elegir las opciones en la ventana Query Editor Settings (Configuración del editor de consultas), elija Save (Guardar).

8. Elija Run (Ejecutar) o pulse Ctrl+Intro, y el editor de consultas mostrará los resultados de su consulta.

   Tras ejecutar la consulta, guárdela en Saved queries (Consultas guardadas) seleccionando Save (Guardar).

   Exporte los resultados de la consulta al formato de hoja de cálculo seleccionando Export to csv (Exportar a csv).

Puede encontrar, editar y volver a ejecutar consultas anteriores. Para ello, seleccione la pestaña Recent (Reciente) o la pestaña Saved queries (Consultas guardadas), seleccione el texto de la consulta y después elija Run (Ejecutar).

Para cambiar la base de datos, elija Change database (Cambiar base de datos).

Referencia de la API de Database Query Metadata Service (DBQMS)

El Database Query Metadata Service (`dbqms`) es un servicio únicamente interno. Proporciona sus consultas recientes y guardadas para el editor de consultas en la Consola de administración de AWS para varios servicios de la AWS, incluido Amazon RDS.

Acciones DBQMS compatibles

- CreateAvoriteQuery
- CreateQueryHistory
- CreateTab

- DeleteFavoritequeries
- DeletEqueryHistory
- Deletetab
- DescribeFavoritequeries
- DescribeQueryHistory
- DescribeABS
- GetQueryString
- UpdateFavoriteQuery
- UpdateQueryHistory
- UpdateTab

CreateAvoriteQuery

Guarde una nueva consulta favorita. Cada usuario puede crear hasta 1000 consultas guardadas. Este límite está sujeto a cambios en el futuro.

CreateQueryHistory

Guarde una nueva entrada del historial de consultas.

CreateTab

Guarde una nueva pestaña de consulta. Cada usuario puede crear hasta 10 pestañas de consulta.

DeleteFavoritequeries

Elimine una o más consultas guardadas.

DeletEqueryHistory

Elimine las entradas del historial de consultas.

Deletetab

Elimine las entradas de la ficha de consulta.

DescribeFavoritequeries

Lista de las consultas guardadas creadas por un usuario en una cuenta determinada.

DescribeQueryHistory

Lista de las entradas del historial de consultas.

DescribeABS

Lista de las pestañas de consulta creadas por un usuario en una cuenta determinada.

GetQueryString

Recupere el texto completo de la consulta de un ID de consulta.

UpdateFavoriteQuery

Actualice la cadena de consulta, la descripción, el nombre o la fecha de caducidad.

UpdateQueryHistory

Actualice el estado del historial de consultas.

UpdateTab

Actualice el nombre de la ficha de consulta y la cadena de consulta.

Uso de machine learning de Amazon Aurora

Mediante el machine learning de Amazon Aurora, puede integrar su clúster de base de datos de Aurora con uno de los siguientes servicios de machine learning de AWS, según sus necesidades. Cada uno admite casos de uso de machine learning específicos.

Amazon Bedrock

Amazon Bedrock es un servicio totalmente administrado que ofrece los principales modelos básicos de las empresas de IA a través de una API, junto con herramientas para desarrolladores que permiten crear y escalar aplicaciones de IA generativa. Con Amazon Bedrock, usted paga por realizar inferencias sobre cualquiera de los modelos fundacionales de terceros. El precio se basa en el volumen de los tokens de entrada y de salida, y en función de si ha adquirido el rendimiento aprovisionado para el modelo. Para obtener más información, consulte ¿Qué es Amazon Bedrock? en la Guía del usuario de Amazon Bedrock.

Amazon Comprehend

Amazon Comprehend es un servicio de procesamiento de lenguaje natural (NLP) que se utiliza para extraer información de los documentos. Con Amazon Comprehend, puede deducir el sentimiento en función del contenido de los documentos mediante el análisis de entidades, frases clave, lenguaje y otras funciones. Para obtener más información, consulte ¿Qué es Amazon Comprehend? en la Guía para desarrolladores de Amazon Comprehend.

IA de SageMaker

IA de Amazon SageMaker es un servicio de machine learning completamente administrado. Los científicos y desarrolladores de datos utilizan IA de Amazon SageMaker para crear, entrenar y probar modelos de machine learning para distintas tareas de inferencia, como la detección de fraudes y la recomendación de productos. Cuando un modelo de machine learning esté listo para usarse en producción, se puede implementar en el entorno alojado de IA de Amazon SageMaker. Para obtener más información, consulte What Is Amazon SageMaker AI? en la Guía para desarrolladores de IA de Amazon SageMaker.

Para usar Amazon Comprehend con su clúster de base de datos de Aurora se requiere menos configuración preliminar que para usar IA de SageMaker. Si es la primera vez que utiliza el machine learning de AWS, le recomendamos que comience por explorar Amazon Comprehend.

Temas

- Uso de machine learning de Amazon Aurora con Aurora MySQL
- Uso de machine learning de Amazon Aurora con Aurora PostgreSQL

Uso de machine learning de Amazon Aurora con Aurora MySQL

Al utilizar el machine learning de Amazon Aurora con su clúster de base de datos de Aurora MySQL, puede utilizar Amazon Bedrock, Amazon Comprehend o IA de Amazon SageMaker, en función de sus necesidades. Cada uno admite casos de uso de machine learning diferentes.

Contenido

- Requisitos para usar machine learning de Aurora con Aurora MySQL
- Disponibilidad en regiones y versiones
- Funciones y limitaciones compatibles del machine learning de Aurora con Aurora MySQL
- Configuración del clúster de base de datos Aurora MySQL para utilizar el machine learning de Aurora
  - Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon Bedrock
  - Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon Comprehend
  - Configuración del clúster de base de datos de Aurora MySQL para utilizar IA de SageMaker
    - Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon S3 para IA de SageMaker (opcional)
  - Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora
    - Concesión de acceso a las funciones de Amazon Bedrock
    - Concesión de acceso a las funciones de Amazon Comprehend
    - Concesión de acceso a las funciones de IA de SageMaker
- Uso de Amazon Bedrock con el clúster de base de datos de Aurora MySQL
- Uso de Amazon Comprehend con el clúster de base de datos de Aurora MySQL
- Uso de IA de SageMaker con el clúster de base de datos de Aurora MySQL
  - Requisito del conjunto de caracteres en las funciones de IA de SageMaker que devuelven cadenas
  - Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado)
- Consideraciones sobre el rendimiento para utilizar el machine learning de Aurora con Aurora MySQL

  - Modelo y petición
  - Caché de consultas
  - Optimización de lotes para las llamadas a las funciones de machine learning de Aurora
- Monitorización del machine learning de Aurora

Requisitos para usar machine learning de Aurora con Aurora MySQL

Los servicios de machine learning de AWS son servicios administrados que se configuran y ejecutan en sus propios entornos de producción. El machine learning de Aurora admite la integración con Amazon Bedrock, Amazon Comprehend e IA de SageMaker. Antes de intentar configurar el clúster de base de datos de Aurora MySQL para usar el machine learning de Aurora, asegúrese de comprender los siguientes requisitos y requisitos previos.

- Los servicios de machine learning deben ejecutarse en la misma Región de AWS que su clúster de base de datos de Aurora MySQL. No puede utilizar los servicios de machine learning de un clúster de base de datos de Aurora MySQL en una región diferente.
- Si su clúster de base de datos de Aurora MySQL se encuentra en una nube pública virtual (VPC) diferente a la de su servicio de Amazon Bedrock, Amazon Comprehend o IA de SageMaker, el grupo de seguridad de la VPC debe permitir las conexiones salientes al servicio de machine learning de Aurora de destino. Para obtener más información, consulte Controlar el tráfico hacia los recursos de AWS mediante grupos de seguridad en la Guía del usuario de Amazon VPC.
- Puede actualizar un clúster de Aurora que ejecute una versión anterior de Aurora MySQL a una versión posterior compatible si desea utilizar el machine learning de Aurora con ese clúster. Para obtener más información, consulte Actualizaciones del motor de base de datos de Amazon Aurora MySQL.
- Su clúster de base de datos de Aurora MySQL debe utilizar un grupo de parámetros de clúster de base de datos personalizado. Al final del proceso de configuración de cada servicio de machine learning de Aurora que desee utilizar, añada el nombre de recurso de Amazon (ARN) del rol de IAM asociado que se creó para el servicio. Le recomendamos que cree un grupo de parámetros de clúster de base de datos personalizado para su Aurora MySQL con antelación y que configure su clúster de base de datos de Aurora MySQL para usarlo de modo que esté listo para modificarlo al final del proceso de configuración.
- Para IA de SageMaker:
  - Los componentes de machine learning que desee usar para las inferencias deben estar configurados y listos para usarse. Durante el proceso de configuración de su clúster de base

de datos de Aurora MySQL, asegúrese de tener disponible el ARN del punto de conexión de IA de SageMaker. Es probable que los científicos de datos de su equipo sean los más capacitados para trabajar con IA de SageMaker para preparar los modelos y gestionar otras tareas similares. Para empezar a utilizar IA de Amazon SageMaker, consulte Get Started with Amazon SageMaker AI. Para obtener más información sobre inferencias y puntos de conexión, consulte Inferencia en tiempo real.

- Para utilizar IA de SageMaker con sus propios datos de entrenamiento, debe configurar un bucket de Amazon S3 como parte de la configuración de Aurora MySQL para el machine learning de Aurora. Para ello, siga el mismo proceso general utilizado para configurar la integración de IA de SageMaker. Para obtener un resumen de este proceso de configuración opcional, consulte Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon S3 para IA de SageMaker (opcional).

- Para las bases de datos globales de Aurora, debe configurar los servicios de machine learning de Aurora que desea utilizar en todas las Regiones de AWS que conforman la base de datos global de Aurora. Por ejemplo, si desea utilizar el machine learning de Aurora con IA de SageMaker para su base de datos global de Aurora, haga lo siguiente para cada clúster de base de datos de Aurora MySQL en cada Región de AWS:
  - Configure los servicios de IA de Amazon SageMaker con los mismos modelos de entrenamiento y puntos de conexión de IA de SageMaker. También deben usar los mismos nombres.
  - Cree los roles de IAM tal y como se detalla en Configuración del clúster de base de datos Aurora MySQL para utilizar el machine learning de Aurora.
  - Agregue el ARN del rol de IAM al grupo de parámetros del clúster de base de datos personalizado para cada clúster de base de datos de Aurora MySQL en cada Región de AWS.

  Estas tareas requieren que el machine learning de Aurora esté disponible para su versión de Aurora MySQL en todas las Regiones de AWS que conforman la base de datos global de Aurora.

Disponibilidad en regiones y versiones

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos de Aurora y entre Regiones de AWS.

- Para obtener información sobre la disponibilidad en versiones y regiones de Amazon Comprehend e IA de Amazon SageMaker con Aurora MySQL, consulte Machine Learning de Aurora con Aurora MySQL.
- Amazon Bedrock solo es compatible con Aurora MySQL versión 3.06 y posteriores.

Para obtener información sobre la disponibilidad en regiones de Amazon Bedrock, consulte Model support by Región de AWS en la Guía del usuario de Amazon Bedrock.

Funciones y limitaciones compatibles del machine learning de Aurora con Aurora MySQL

Al usar Aurora MySQL con el machine learning de Aurora, se aplican las siguientes limitaciones:

- La extensión de machine learning de Aurora no admite interfaces vectoriales.
- Las integraciones de machine learning de Aurora no se admiten cuando se utilizan en un disparador.
- Las funciones de machine learning de Aurora no son compatibles con la replicación de registro binario (binlog).
  - La opción `--binlog-format=STATEMENT` genera una excepción en las llamadas a las funciones de Machine Learning de Aurora.
  - Las funciones de machine learning de Aurora son no deterministas, y las funciones almacenadas no deterministas no son compatibles con el formato binlog.

  Para obtener más información, consulte Binary Logging Formats en la documentación de MySQL.
- No se admiten las funciones almacenadas que llaman a tablas con columnas generadas siempre. Esto se aplica a cualquier función almacenada de Aurora MySQL. Para obtener más información sobre este tipo de columna, consulte CREAR TABLA y columnas generadas en la documentación de MySQL.
- Las funciones de Amazon Bedrock no admiten `RETURNS JSON`. Puede usar `CONVERT` o `CAST` para convertir `TEXT` en `JSON` si es necesario.
- Amazon Bedrock no admite solicitudes por lotes.
- Aurora MySQL es compatible con cualquier punto de conexión de IA de SageMaker que lea y escriba el formato de valores separados por comas (CSV), a través de un `ContentType` de `text/csv`. Los siguientes algoritmos integrados de IA de SageMaker aceptan este formato:
  - Linear Learner
  - Random Cut Forest
  - XGBoost

Para obtener más información sobre estos algoritmos, consulte Choose an Algorithm en la Guía para desarrolladores de IA de Amazon SageMaker.

Configuración del clúster de base de datos Aurora MySQL para utilizar el machine learning de Aurora

En los siguientes temas, puede encontrar procedimientos de configuración independientes para cada uno de estos servicios de machine learning de Aurora.

Temas

- Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon Bedrock
- Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon Comprehend
- Configuración del clúster de base de datos de Aurora MySQL para utilizar IA de SageMaker
  - Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon S3 para IA de SageMaker (opcional)
- Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora
  - Concesión de acceso a las funciones de Amazon Bedrock
  - Concesión de acceso a las funciones de Amazon Comprehend
  - Concesión de acceso a las funciones de IA de SageMaker

Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon Bedrock

El machine learning de Aurora se basa en roles de AWS Identity and Access Management (IAM) y políticas para permitir que su clúster de base de datos de Aurora MySQL acceda a los servicios de Amazon Bedrock y los utilice. Los siguientes procedimientos crean una política de permisos y un rol de IAM para que el clúster de base de datos se pueda integrar con Amazon Bedrock.

Creación de la política de IAM

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Políticas.
3. Elija Crear una política.

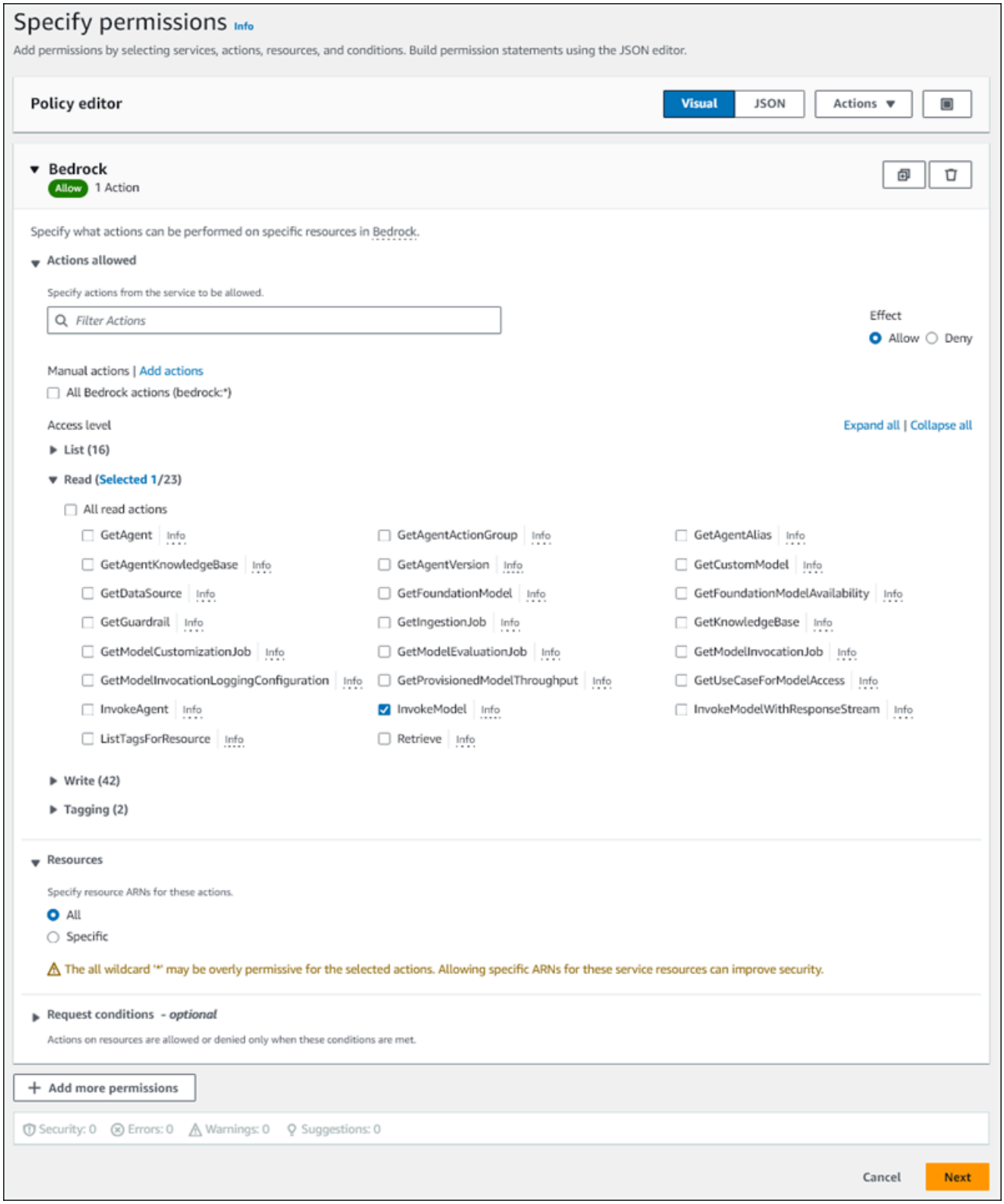
4. En la página Especificar permisos, en Seleccionar un servicio, elija Bedrock.

   Aparecen los permisos de Amazon Bedrock.

5. Expanda Leer y, a continuación, seleccione InvokeModel.

6. En Recursos, seleccione Todos.

   La página Especificar permisos debería parecerse a la de la siguiente figura.



7. Elija Siguiente.
8. En la página Revisar y crear, en introduzca un nombre para la política como, por ejemplo, **`BedrockInvokeModel`**.

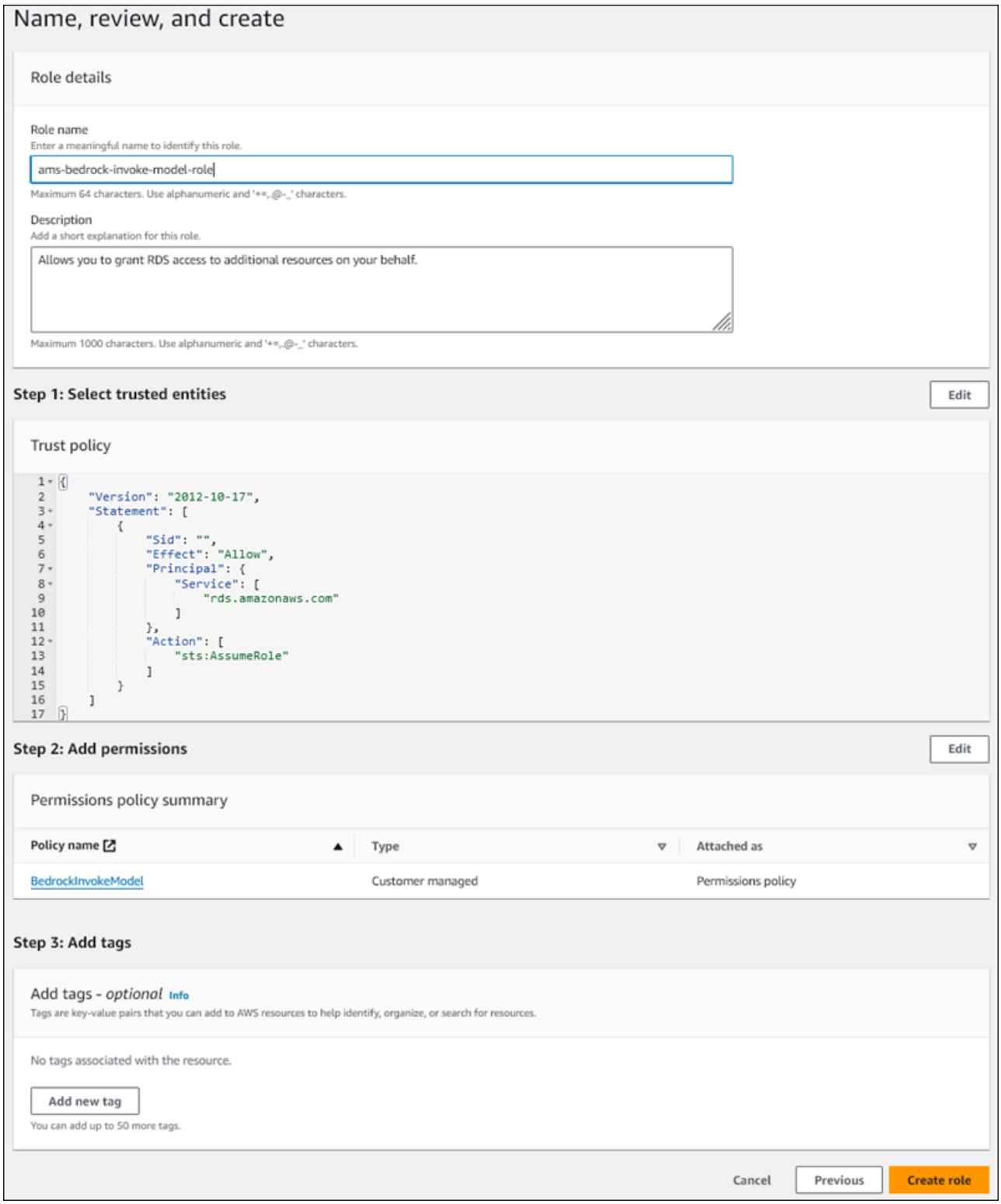
9. Revise la política y luego seleccione Crear política.

A continuación, debe crear el rol de IAM que usa la política de permisos de Amazon Bedrock.

Creación del rol de IAM

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles en el panel de navegación.
3. Elija Creación de rol.
4. En la página Seleccionar entidad de confianza, en Caso de uso, elija RDS.
5. Seleccione RDS - Agregar rol a la base de datos y, a continuación, elija Siguiente.
6. En la página Agregar permisos, en Políticas de permisos, seleccione la política de IAM que creó y, a continuación, seleccione Siguiente.
7. En la página Asignar nombre, revisar y crear, introduzca un nombre para su rol como, por ejemplo, **`ams-bedrock-invoke-model-role`**.

   El rol debería parecerse al de la siguiente figura.



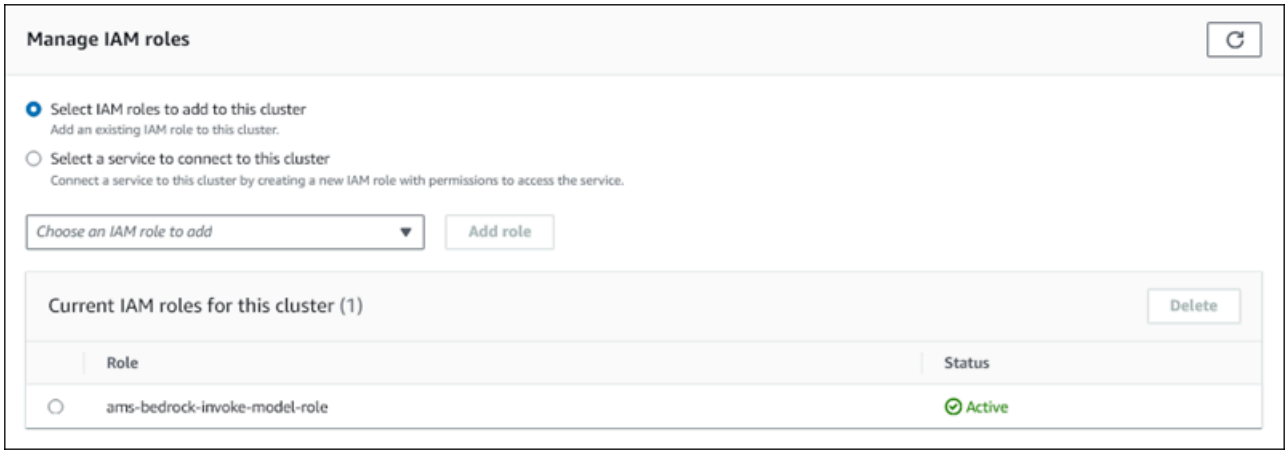
8. Revise el rol y, a continuación, elija Crear rol.

A continuación, asocie el rol de IAM de Amazon Bedrock con su clúster de base de datos.

Asociación del rol de IAM a su clúster de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Bases de datos en el panel de navegación.
3. Elija el clúster de base de datos de Aurora MySQL que desea conectar a los servicios de Amazon Bedrock.
4. Elija la pestaña Conectividad y seguridad.
5. En la sección Administrar los roles de IAM, elija Seleccionar roles de IAM para agregarlos a este clúster.
6. Elija el rol de IAM que creó y, a continuación, seleccione Agregar rol.

   El rol de IAM está asociado a su clúster de base de datos, primero con el estado Pendiente y, después, Activo. Cuando se complete el proceso, puede encontrar el rol en la lista Current IAM roles for this clúster (Roles de IAM actuales para este clúster).



Debe agregar el ARN de este rol de IAM al parámetro `aws_default_bedrock_role` del grupo de parámetros del clúster de base de datos personalizado asociado a su clúster de base de datos de Aurora MySQL. Si el clúster de base de datos de Aurora MySQL no usa un grupo de parámetros de clúster de base de datos personalizado, debe crear uno para usarlo con el clúster de base de datos de Aurora MySQL para completar la integración. Para obtener más información, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

Configuración del parámetro del clúster de base de datos

1. En la consola de Amazon RDS, abra la pestaña Configuración del clúster de base de datos de Aurora MySQL.

2. Localice el grupo de parámetros del clúster de base de datos configurado para su clúster. Elija el enlace para abrir el grupo de parámetros del clúster de base de datos personalizado y, a continuación elija Editar.
3. Busque el parámetro `aws_default_bedrock_role` en el grupo de parámetros del clúster de base de datos personalizado.
4. En el campo Valor, escriba el ARN del rol de IAM.
5. Elija Guardar cambios, para guardar la configuración.
6. Reinicie la instancia principal del clúster de base de datos de Aurora MySQL para que se aplique esta configuración de parámetros.

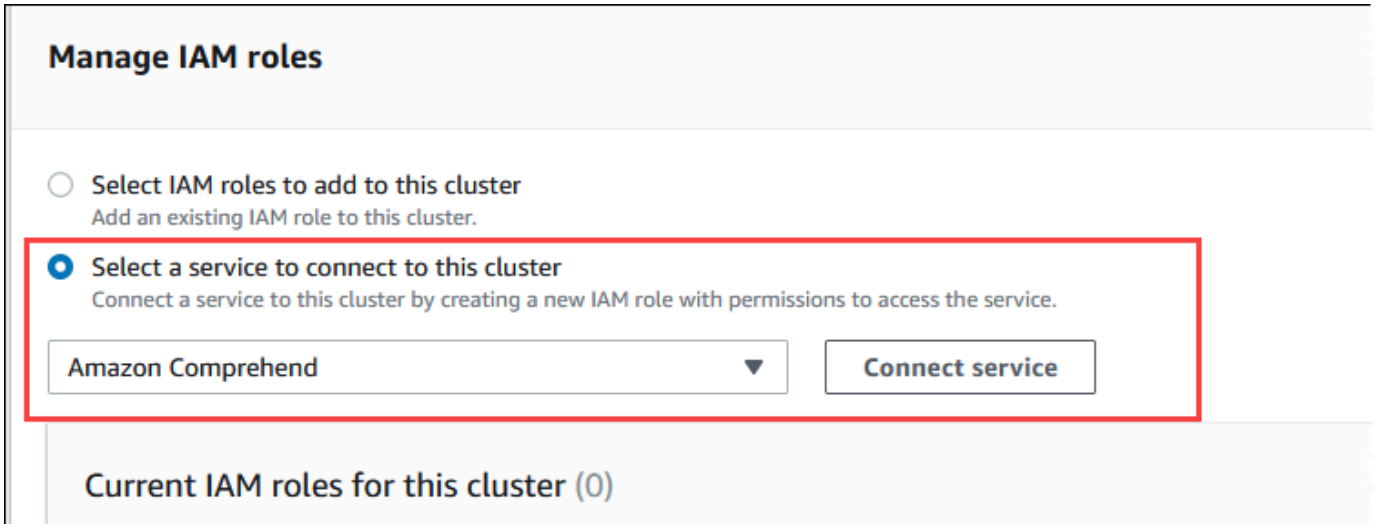
La integración de IAM para Amazon Bedrock está completa. Continúe configurando su clúster de base de datos de Aurora MySQL para que funcione con Amazon Bedrock mediante Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora.

Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon Comprehend

El machine learning de Aurora se basa en roles y políticas de AWS Identity and Access Management para permitir que su clúster de base de datos de Aurora MySQL acceda a los servicios de Amazon Comprehend y los utilice. El siguiente procedimiento crea automáticamente una política y un rol de IAM para su clúster, de modo que pueda usar Amazon Comprehend.

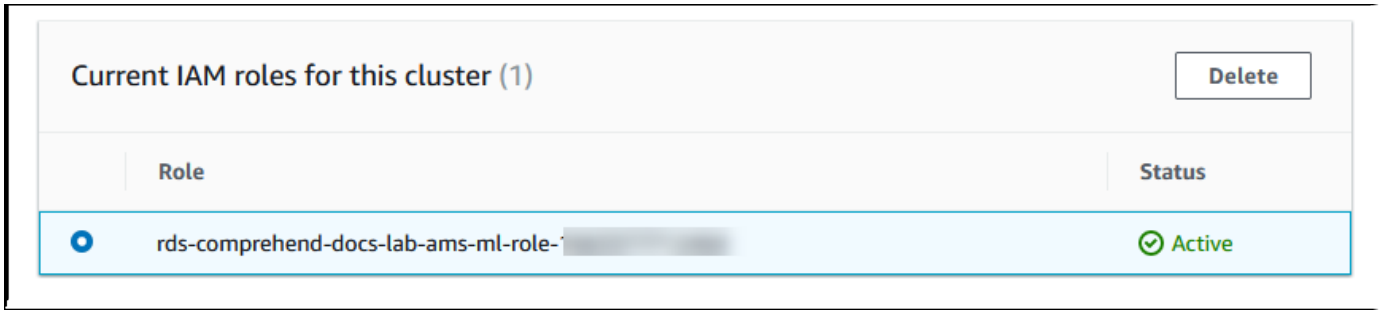
Para configurar el clúster de base de datos de Aurora MySQL para utilizar Amazon Comprehend

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Bases de datos en el panel de navegación.
3. Elija el clúster de base de datos de Aurora MySQL que desea conectar a los servicios de Amazon Comprehend.
4. Elija la pestaña Conectividad y seguridad.
5. En la sección Administrar los roles de IAM, elija Seleccionar un servicio para conectarse a este clúster.
6. Elija Amazon Comprehend en el menú y, a continuación, seleccione Conectar servicio.



7. El cuadro de diálogo Connect clúster to Amazon Comprehend (Conectar el clúster a Amazon Comprehend) no requiere información adicional. Sin embargo, es posible que vea un mensaje que le notifique que la integración entre Aurora y Amazon Comprehend se encuentra actualmente en versión preliminar. Asegúrese de leer el mensaje antes de continuar. Puede elegir Cancelar si prefiere no continuar.
8. Elija Connect service (Conectar servicio) para completar el proceso de integración.

   Aurora crea el rol de IAM. También crea la política que permite al clúster de base de datos de Aurora MySQL utilizar los servicios de Amazon Comprehend y asocia la política al rol. Cuando se complete el proceso, puede encontrar el rol en la lista Current IAM roles for this clúster (Roles de IAM actuales para este clúster), como se muestra en la siguiente imagen.



   Debe agregar el ARN de este rol de IAM al parámetro `aws_default_comprehend_role` del grupo de parámetros del clúster de base de datos personalizado asociado a su clúster de base de datos de Aurora MySQL. Si el clúster de base de datos de Aurora MySQL no usa un grupo de parámetros de clúster de base de datos personalizado, debe crear uno para usarlo con el clúster de base de datos de Aurora MySQL para completar la integración. Para obtener más información, consulte Grupos de parámetros de clústeres de base de datos para clústeres de base de datos en Amazon Aurora.

Tras crear el grupo de parámetros del clúster de base de datos personalizado y asociarlo al clúster de base de datos de Aurora MySQL, puede seguir estos pasos.

Si el clúster utiliza un grupo de parámetros del clúster de base de datos personalizado, haga lo siguiente.

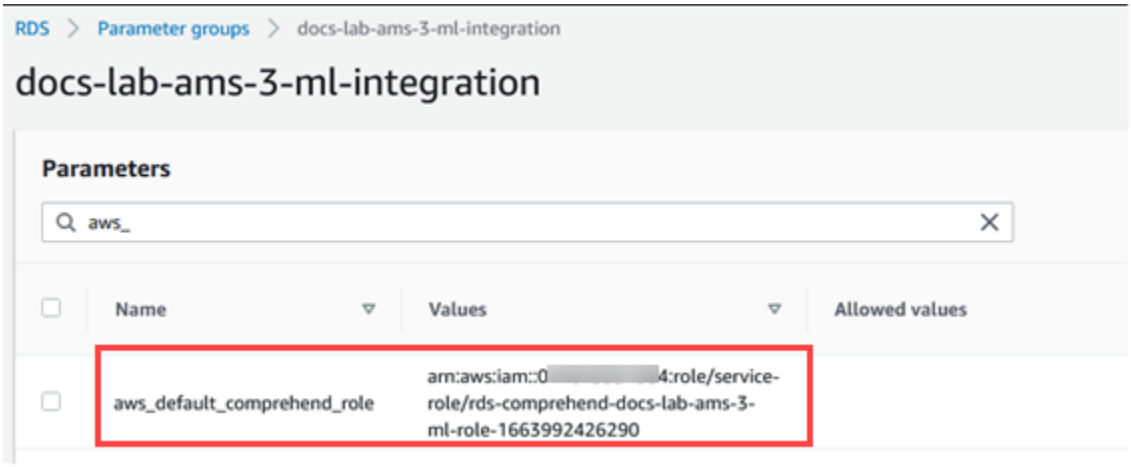
a. En la consola de Amazon RDS, abra la pestaña Configuración del clúster de base de datos de Aurora MySQL.

b. Localice el grupo de parámetros del clúster de base de datos configurado para su clúster. Elija el enlace para abrir el grupo de parámetros del clúster de base de datos personalizado y, a continuación elija Editar.

c. Busque el parámetro `aws_default_comprehend_role` en el grupo de parámetros del clúster de base de datos personalizado.

d. En el campo Valor, escriba el ARN del rol de IAM.

e. Elija Save Changes (Guardar cambios), para guardar la configuración. En la siguiente imagen, puede ver un ejemplo.

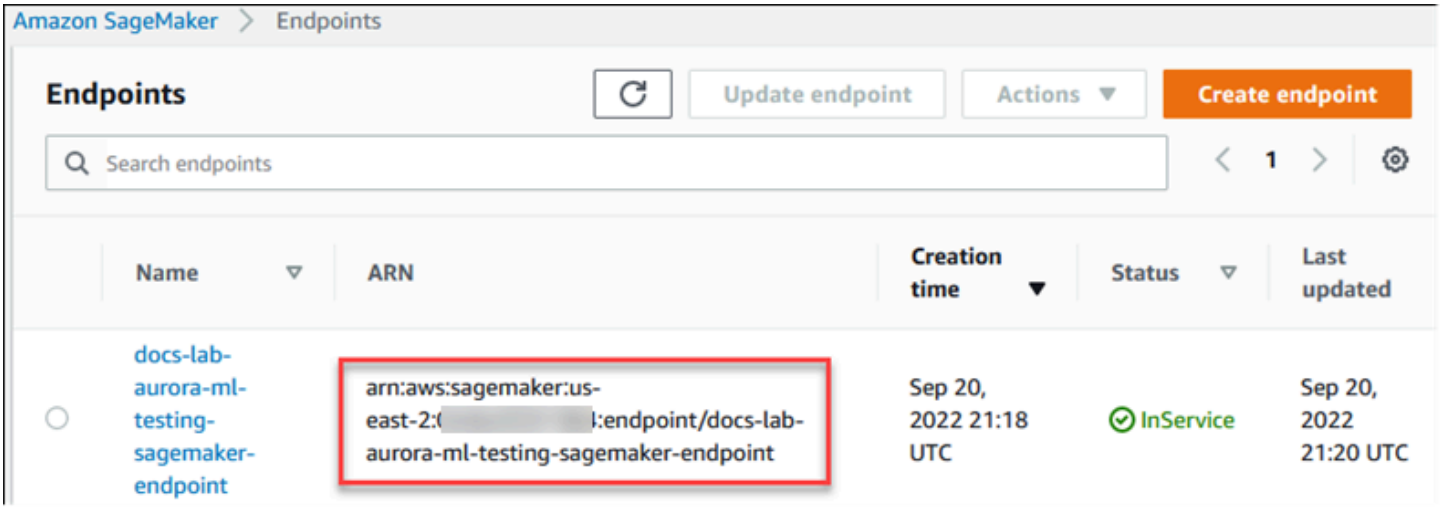


Reinicie la instancia principal del clúster de base de datos de Aurora MySQL para que se aplique esta configuración de parámetros.

La integración de IAM para Amazon Comprehend está completa. Siga configurando su clúster de base de datos Aurora MySQL para que funcione con Amazon Comprehend concediendo acceso a los usuarios de la base de datos adecuados.

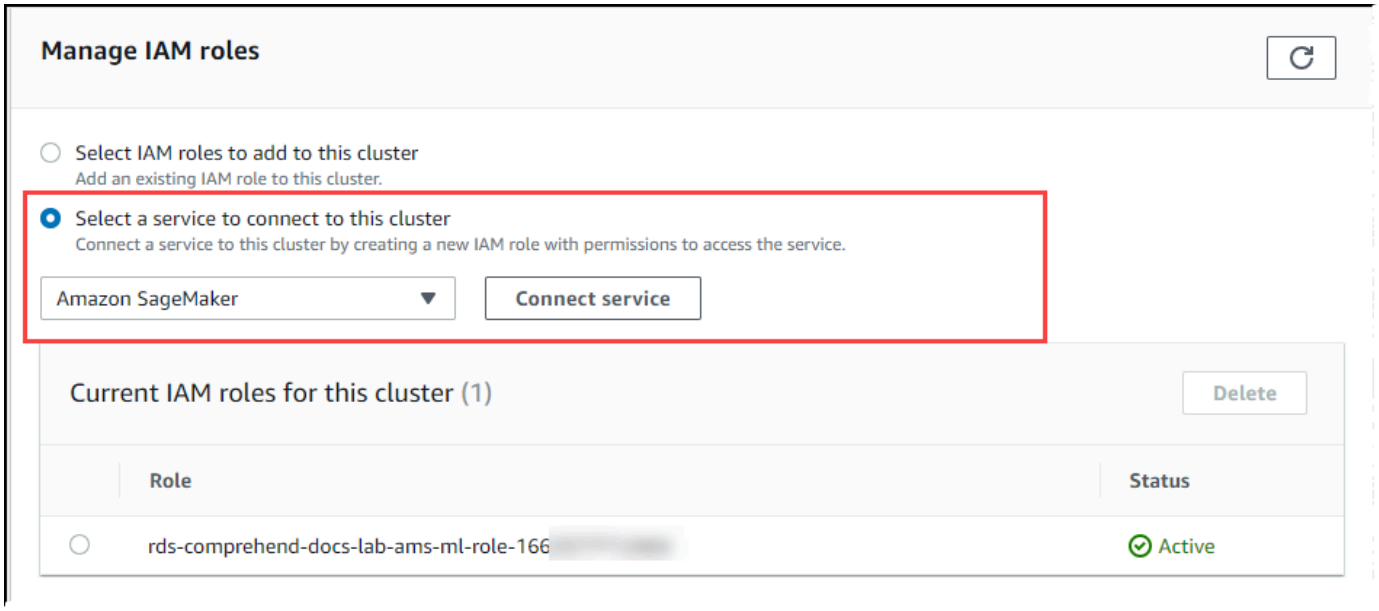
Configuración del clúster de base de datos de Aurora MySQL para utilizar IA de SageMaker

El siguiente procedimiento crea automáticamente una política y un rol de IAM para su clúster de base de datos de Aurora MySQL, de modo que pueda usar IA de SageMaker. Antes de intentar seguir este procedimiento, asegúrese de tener disponible el punto de conexión de IA de SageMaker para poder introducirlo cuando sea necesario. Por lo general, los científicos de datos de su equipo se encargarían de crear un punto de conexión que pueda utilizar desde su clúster de base de datos de Aurora MySQL. Puede encontrar estos puntos de conexión en la consola de IA de SageMaker. En el panel de navegación, abra el menú Inferencia y elija Puntos de conexión. En la siguiente imagen, puede ver un ejemplo.

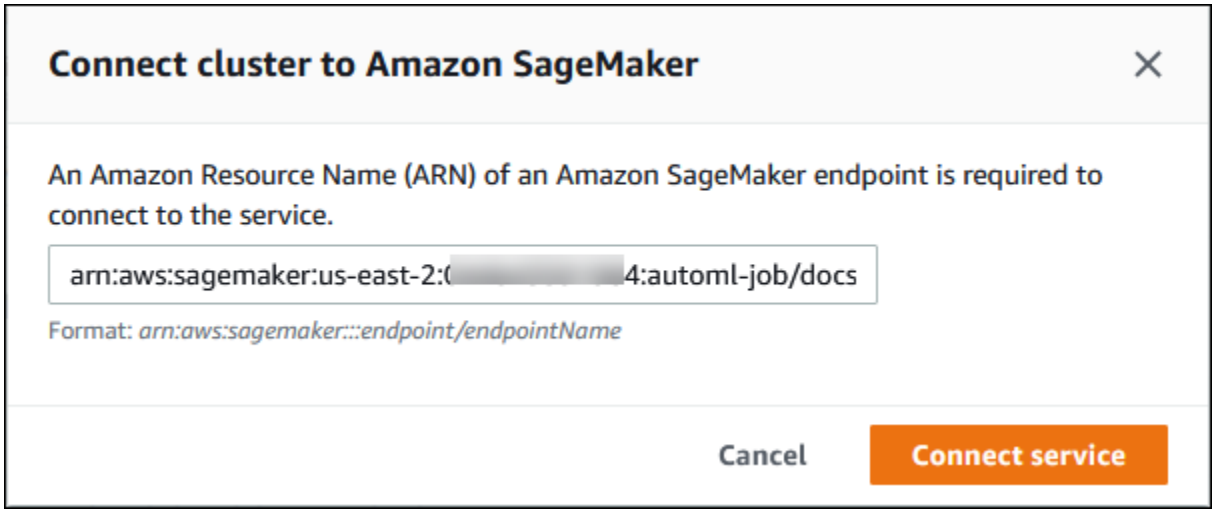


Configuración del clúster de base de datos Aurora MySQL para que utilice IA de SageMaker

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Base de datos en el menú de navegación de Amazon RDS y, a continuación elija el clúster de base de datos de Aurora MySQL que desee conectar a los servicios de IA de SageMaker.
3. Elija la pestaña Conectividad y seguridad.
4. Desplácese a la sección Administrar roles de IAM y, a continuación, elija Seleccione un servicio para conectarse a este clúster. Elija IA de SageMaker en el selector.



5. Elija Conectar servicio.
6. En el cuadro de diálogo Conectar el clúster a IA de SageMaker, escriba el ARN del punto de conexión de IA de SageMaker.



7. Aurora crea el rol de IAM. También crea la política que permite al clúster de base de datos de Aurora MySQL utilizar los servicios de IA de SageMaker y asocia la política al rol. Cuando se complete el proceso, puede encontrar el rol en la lista Roles de IAM actuales para este clúster.
8. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
9. Elija Roles en la sección Gestión de acceso del menú de navegación de AWS Identity and Access Management.
10. Busque el rol entre los que figuran en la lista. Su nombre utiliza el siguiente patrón.

```
rds-sagemaker-your-cluster-name-role-auto-generated-digits
```

11. Abra la página de resumen del rol y localice el ARN. Anote el ARN o cópielo con el widget de copia.
12. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
13. Elija el clúster de base de datos de Aurora MySQL y, a continuación, elija la pestaña Configuration (Configuración).
14. Localice el grupo de parámetros del clúster de base de datos y elija el enlace para abrir el grupo de parámetros del clúster de base de datos personalizado. Busque el parámetro `aws_default_sagemaker_role` e introduzca el ARN del rol de IAM en el campo Value (Valor) y guarde la configuración.
15. Reinicie la instancia principal del clúster de base de datos de Aurora MySQL para que se aplique esta configuración de parámetros.

La configuración de IAM ya se ha completado. Siga configurando su clúster de base de datos Aurora MySQL para que funcione con IA de SageMaker concediendo acceso a los usuarios de la base de datos adecuados.

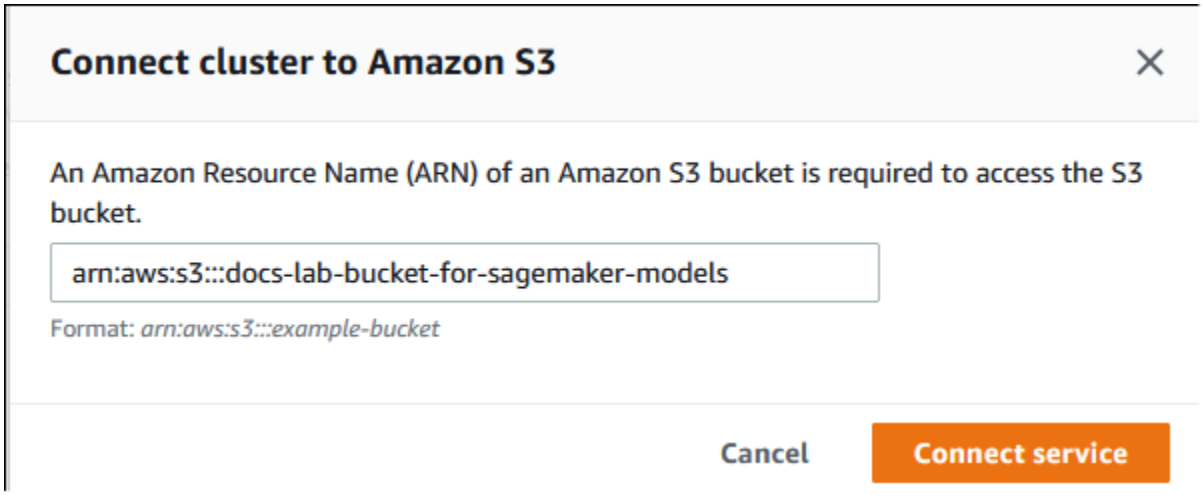
Si desea utilizar sus modelos de IA de SageMaker para el entrenamiento en lugar de utilizar componentes prediseñados de IA de SageMaker, también debe añadir el bucket de Amazon S3 a su clúster de base de datos de Aurora MySQL, tal y como se describe en Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon S3 para IA de SageMaker (opcional) a continuación.

Configuración del clúster de base de datos de Aurora MySQL para utilizar Amazon S3 para IA de SageMaker (opcional)

Para utilizar IA de SageMaker con sus propios modelos en lugar de utilizar los componentes prediseñados que ofrece IA de SageMaker, debe configurar un bucket de Amazon S3 para que lo utilice el clúster de base de datos de Aurora MySQL. Para obtener más información sobre la creación de un bucket de Amazon S3, consulte la sección de Creación de un bucket en la Guía del usuario de Amazon Simple Storage Service.

Configuración del clúster de base de datos Aurora MySQL para que utilice un bucket de Amazon S3 para IA de SageMaker

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. Elija Base de datos en el menú de navegación de Amazon RDS y, a continuación elija el clúster de base de datos de Aurora MySQL que desee conectar a los servicios de IA de SageMaker.
3. Elija la pestaña Conectividad y seguridad.
4. Desplácese a la sección Manage IAM roles (Administrar roles de IAM) y, a continuación, elija Select a service to connect to this clúster (Seleccione un servicio para conectarse a este clúster). Elija Amazon S3 en el selector.
5. Elija Conectar servicio.
6. En el cuadro de diálogo Conectar el clúster a Amazon S3, escriba el ARN del bucket de Amazon S3, según se muestra en la imagen siguiente.



7. Elija Conectar servicio para completar este proceso.

Para obtener más información sobre el uso de los buckets de Amazon S3 con IA de SageMaker, consulte Specify an Amazon S3 Bucket to Upload Training Datasets and Store Output Data en la Guía para desarrolladores de IA de Amazon SageMaker. Para obtener más información sobre cómo trabajar con IA de SageMaker, consulte Get Started with Amazon SageMaker AI Notebook Instances en la Guía para desarrolladores de Amazon SageMaker.

Concesión de acceso a los usuarios de bases de datos a machine learning de Aurora

Los usuarios de la base de datos deben tener permiso para invocar las funciones de machine learning de Aurora. La forma de conceder los permisos depende de la versión de MySQL que utilice para el clúster de base de datos Aurora MySQL, tal como se describe a continuación. La forma de hacerlo depende de la versión de MySQL que utilice el clúster de base de datos de Aurora MySQL.

- Para la versión 3 de Aurora MySQL (compatible con MySQL 8.0), los usuarios de bases de datos deben tener el rol de base de datos adecuado. Para obtener más información, consulte Using Roles en el Manual de referencia de MySQL 8.0.
- Para la versión 2 de Aurora MySQ (compatible con MySQL 5.7), los usuarios de bases de datos tienen privilegios. Para obtener más información, consulte Access Control and Account Management en el Manual de referencia de MySQL 5.7.

La siguiente tabla muestra los roles y los privilegios que los usuarios de bases de datos necesitan para trabajar con funciones de machine learning.

Aurora MySQL versión 3 (rol)	Aurora MySQL versión 2 (privilegio)
AWS_BEDROCK_ACCESS	–
AWS_COMPREHEND_ACCESS	INVOKE COMPREHEND
AWS_SAGEMAKER_ACCESS	INVOKE SAGEMAKER

Concesión de acceso a las funciones de Amazon Bedrock

Para dar a los usuarios de bases de datos acceso a las funciones de Amazon Bedrock, utilice la siguiente instrucción de SQL:

```
GRANT AWS_BEDROCK_ACCESS TO user@domain-or-ip-address;
```

Los usuarios de bases de datos también deben tener permisos `EXECUTE` para las funciones que cree para trabajar con Amazon Bedrock:

```
GRANT EXECUTE ON FUNCTION database_name.function_name TO user@domain-or-ip-address;
```

Por último, los usuarios de la base de datos deben tener sus roles configurados en:`AWS_BEDROCK_ACCESS`

```
SET ROLE AWS_BEDROCK_ACCESS;
```

Las funciones de Amazon Bedrock ya están disponibles para su uso.

Concesión de acceso a las funciones de Amazon Comprehend

Para dar a los usuarios de bases de datos acceso a las funciones de Amazon Comprehend, utilice la sentencia correspondiente para su versión de Aurora MySQL.

- Aurora MySQL versión 3 (compatible con MySQL 8.0)

```
GRANT AWS_COMPREHEND_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL versión 2 (compatible con MySQL 5.7)

```
GRANT INVOKE COMPREHEND ON *.* TO user@domain-or-ip-address;
```

Las funciones de Amazon Comprehend ya están disponibles para su uso. Para ejemplos de uso, consulte Uso de Amazon Comprehend con el clúster de base de datos de Aurora MySQL.

Concesión de acceso a las funciones de IA de SageMaker

Para dar a los usuarios de bases de datos acceso a las funciones de IA de SageMaker, utilice la sentencia correspondiente para su versión de Aurora MySQL.

- Aurora MySQL versión 3 (compatible con MySQL 8.0)

```
GRANT AWS_SAGEMAKER_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL versión 2 (compatible con MySQL 5.7)

```
GRANT INVOKE SAGEMAKER ON *.* TO user@domain-or-ip-address;
```

Los usuarios de bases de datos también deben tener permisos `EXECUTE` para las funciones que cree para trabajar con IA de SageMaker. Supongamos que ha creado dos funciones `db1.anomoly_score` y `db2.company_forecasts`, para invocar los servicios de su punto de

conexión de IA de SageMaker. Debe conceder privilegios de ejecución como se muestra en el siguiente ejemplo.

```
GRANT EXECUTE ON FUNCTION db1.anomaly_score TO user1@domain-or-ip-address1;
GRANT EXECUTE ON FUNCTION db2.company_forecasts TO user2@domain-or-ip-address2;
```

Las funciones de IA de SageMaker ya están disponibles para su uso. Para ejemplos de uso, consulte Uso de IA de SageMaker con el clúster de base de datos de Aurora MySQL.

Uso de Amazon Bedrock con el clúster de base de datos de Aurora MySQL

Para utilizar Amazon Bedrock, debe crear una función definida por el usuario (UDF) en la base de datos de Aurora MySQL que invoque un modelo. Para obtener más información, consulte Modelos compatibles en Amazon Bedrock en la Guía del usuario de Amazon Bedrock.

Una UDF utiliza la siguiente sintaxis:

```
CREATE FUNCTION function_name (argument type)
        [DEFINER = user]
    RETURNS mysql_data_type
    [SQL SECURITY {DEFINER | INVOKER}]
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'model_id'
    [CONTENT_TYPE 'content_type']
    [ACCEPT 'content_type']
    [TIMEOUT_MS timeout_in_milliseconds];
```

- Las funciones de Amazon Bedrock no admiten `RETURNS JSON`. Puede usar `CONVERT` o `CAST` para convertir `TEXT` en `JSON` si es necesario.
- Si no especifica `CONTENT_TYPE` ni `ACCEPT`, el valor predeterminado es `application/json`.
- Si no especifica `TIMEOUT_MS`, se utiliza el valor de `aurora_ml_inference_timeout`.

Por ejemplo, la siguiente UDF invoca el modelo Text Express de Amazon Titan:

```
CREATE FUNCTION invoke_titan (request_body TEXT)
    RETURNS TEXT
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'amazon.titan-text-express-v1'
    CONTENT_TYPE 'application/json'
```

```
    ACCEPT 'application/json';
```

Para permitir que un usuario de base de datos utilice esta función, utilice el siguiente comando de SQL:

```
GRANT EXECUTE ON FUNCTION database_name.invoke_titan TO user@domain-or-ip-address;
```

A continuación, el usuario puede llamar a `invoke_titan` como a cualquier otra función, como se muestra en el siguiente ejemplo. Asegúrese de formatear el cuerpo de la solicitud de acuerdo con los modelos de texto de Amazon Titan.

```
CREATE TABLE prompts (request varchar(1024));
INSERT INTO prompts VALUES (
'{
    "inputText": "Generate synthetic data for daily product sales in various categories
 - include row number, product name, category, date of sale and price. Produce output
 in JSON format. Count records and ensure there are no more than 5.",
    "textGenerationConfig": {
        "maxTokenCount": 1024,
        "stopSequences": [],
        "temperature":0,
        "topP":1
    }
}');

SELECT invoke_titan(request) FROM prompts;

{"inputTextTokenCount":44,"results":[{"tokenCount":296,"outputText":"
```tabular-data-json
{
    "rows": [
        {
            "Row Number": "1",
            "Product Name": "T-Shirt",
            "Category": "Clothing",
            "Date of Sale": "2024-01-01",
            "Price": "$20"
        },
        {
            "Row Number": "2",
            "Product Name": "Jeans",
            "Category": "Clothing",
```
```

```
            "Date of Sale": "2024-01-02",
            "Price": "$30"
        },
        {
            "Row Number": "3",
            "Product Name": "Hat",
            "Category": "Accessories",
            "Date of Sale": "2024-01-03",
            "Price": "$15"
        },
        {
            "Row Number": "4",
            "Product Name": "Watch",
            "Category": "Accessories",
            "Date of Sale": "2024-01-04",
            "Price": "$40"
        },
        {
            "Row Number": "5",
            "Product Name": "Phone Case",
            "Category": "Accessories",
            "Date of Sale": "2024-01-05",
            "Price": "$25"
        }
    ]
}
```","completionReason":"FINISH"}]}
```

Para otros modelos que utilice, asegúrese de formatear el cuerpo de la solicitud de forma adecuada para ellos. Para obtener más información, consulte Inference parameters for foundation models en la Guía del usuario de Amazon Bedrock.

## Uso de Amazon Comprehend con el clúster de base de datos de Aurora MySQL

Para Aurora MySQL, el machine learning de Aurora proporciona las dos funciones integradas siguientes para trabajar con Amazon Comprehend y sus datos de texto. Proporciona el texto para analizar (`input_data`) y especifica el idioma (`language_code`).
```

aws_comprehend_detect_sentiment

Esta función identifica que el texto tiene una postura emocional positiva, negativa, neutra o mixta. La documentación de referencia de esta función es la siguiente.

```
aws_comprehend_detect_sentiment(
  input_text,
  language_code
  [,max_batch_size]
)
```

Para obtener más información, consulte Sentiment en la Guía para desarrolladores de Amazon Comprehend.

aws_comprehend_detect_sentiment_confidence

Esta función mide el nivel de confianza del sentimiento detectado para un texto determinado. Devuelve un valor (type, `double`) que indica la confianza del sentimiento asignado al texto por la función aws_comprehend_detect_sentiment. La confianza es una métrica estadística entre 0 y 1. Cuanto mayor sea el nivel de confianza, mayor será el peso que se le pueda dar al resultado. A continuación se presenta un resumen de la documentación de la función.

```
aws_comprehend_detect_sentiment_confidence(
  input_text,
  language_code
  [,max_batch_size]
)
```

En ambas funciones (aws_comprehend_detect_sentiment_confidence, aws_comprehend_detect_sentiment), `max_batch_size` utiliza un valor por defecto de 25 si no se especifica ninguno. El tamaño del lote siempre debe ser mayor que 0. Puede usar `max_batch_size` para ajustar el rendimiento de las llamadas a funciones de Amazon Comprehend. Un tamaño de lote grande sacrifica un rendimiento más rápido por un mayor uso de la memoria en el clúster de base de datos de Aurora MySQL. Para obtener más información, consulte Consideraciones sobre el rendimiento para utilizar el machine learning de Aurora con Aurora MySQL.

Para obtener más información acerca de los parámetros y los tipos de valores devueltos en las funciones de detección de opiniones de Amazon Comprehend, consulte DetectSentiment.

Example Ejemplo: una consulta sencilla con las funciones de Amazon Comprehend

Este es un ejemplo de una consulta sencilla que invoca estas dos funciones para ver el grado de satisfacción de los clientes con su equipo de soporte. Supongamos que tiene una tabla de base de datos (`support`) que almacena los comentarios de los clientes después de cada solicitud de ayuda. En esta consulta de ejemplo se aplican ambas funciones integradas al texto de la columna `feedback` de la tabla y se obtienen los resultados. Los valores de confianza devueltos por la función son dobles entre 0,0 (0,0) y 1,0 Para obtener resultados más legibles, esta consulta redondea los resultados a 6 decimales. Para facilitar las comparaciones, esta consulta también ordena los resultados en orden descendente, empezando por el que tiene el mayor grado de confianza.

```
SELECT feedback AS 'Customer feedback',
    aws_comprehend_detect_sentiment(feedback, 'en') AS Sentiment,
    ROUND(aws_comprehend_detect_sentiment_confidence(feedback, 'en'), 6)
    AS Confidence FROM support
    ORDER BY Confidence DESC;
+-----------------------------------------------------------+-----------+------------+
| Customer feedback                                         | Sentiment | Confidence |
+-----------------------------------------------------------+-----------+------------+
| Thank you for the excellent customer support!             | POSITIVE  |   0.999771 |
| The latest version of this product stinks!                | NEGATIVE  |   0.999184 |
| Your support team is just awesome! I am blown away.       | POSITIVE  |   0.997774 |
| Your product is too complex, but your support is great.   | MIXED     |   0.957958 |
| Your support tech helped me in fifteen minutes.           | POSITIVE  |   0.949491 |
| My problem was never resolved!                            | NEGATIVE  |   0.920644 |
| When will the new version of this product be released?    | NEUTRAL   |   0.902706 |
| I cannot stand that chatbot.                              | NEGATIVE  |   0.895219 |
| Your support tech talked down to me.                      | NEGATIVE  |   0.868598 |
| It took me way too long to get a real person.             | NEGATIVE  |   0.481805 |
+-----------------------------------------------------------+-----------+------------+
 10 rows in set (0.1898 sec)
```

Example Ejemplo: determinar el sentimiento medio de un texto por encima de un nivel de confianza específico

Una consulta de Amazon Comprehend típica busca filas en las que la opinión sea un valor determinado, con un nivel de confianza superior a un número determinado. Por ejemplo, la siguiente consulta muestra cómo puede determinar el promedio de la opinión de los documentos de la base de datos. La consulta tiene en cuenta solo los documentos en los que la confianza de la evaluación sea al menos del 80 %.

```
SELECT AVG(CASE aws_comprehend_detect_sentiment(productTable.document, 'en')
  WHEN 'POSITIVE' THEN 1.0
  WHEN 'NEGATIVE' THEN -1.0
  ELSE 0.0 END) AS avg_sentiment, COUNT(*) AS total
FROM productTable
WHERE productTable.productCode = 1302 AND
    aws_comprehend_detect_sentiment_confidence(productTable.document, 'en') >= 0.80;
```

Uso de IA de SageMaker con el clúster de base de datos de Aurora MySQL

Para usar la funcionalidad de IA de SageMaker desde su clúster de base de datos de Aurora MySQL, debe crear funciones almacenadas que integren las llamadas en el punto de conexión de IA de SageMaker y sus funciones de inferencia. Se logra utilizando `CREATE FUNCTION` de MySQL generalmente de la misma manera que lo hace para otras tareas de procesamiento en su clúster de base de datos de Aurora MySQL.

Para utilizar los modelos implementados en IA de SageMaker para la inferencia, cree funciones definidas por el usuario mediante las instrucciones en lenguaje de definición de datos (DDL) de MySQL para las funciones almacenadas. Cada función almacenada representa el punto de conexión de IA de SageMaker que aloja el modelo. Al definir una función así, especifique los paramentos de entrada en el modelo, el punto de conexión de IA de SageMaker específico que desee invocar y el tipo de valor devuelto. La función devuelve la inferencia calculada por el punto de conexión de IA de SageMaker después de aplicar el modelo a los parámetros de entrada.

Todas las funciones almacenadas de machine learning de Aurora devuelven tipos numéricos o `VARCHAR`. Puede utilizar cualquier tipo numérico excepto `BIT`. No se permiten otros tipos, como `JSON`, `BLOB`, `TEXT` y `DATE`.

En el siguiente ejemplo, se muestra la sintaxis de `CREATE FUNCTION` para trabajar con IA de SageMaker.

```
CREATE FUNCTION function_name (
     arg1 type1,
     arg2 type2, ...)
     [DEFINER = user]
  RETURNS mysql_type
  [SQL SECURITY { DEFINER | INVOKER } ]
  ALIAS AWS_SAGEMAKER_INVOKE_ENDPOINT
  ENDPOINT NAME 'endpoint_name'
  [MAX_BATCH_SIZE max_batch_size];
```

Esta es una extensión de la instrucción regular de DDL de `CREATE FUNCTION`. En la instrucción `CREATE FUNCTION` que define la función de IA de SageMaker, no se especifica el cuerpo de la función. En cambio, se especifica la palabra clave `ALIAS` donde normalmente va el cuerpo de la función. Actualmente, el machine learning de Aurora solo admite `aws_sagemaker_invoke_endpoint` en esta sintaxis ampliada. Debe especificar el parámetro `endpoint_name`. Un punto de conexión de IA de SageMaker puede tener diferentes características en cada modelo.

> **Note**
>
> Para obtener más información sobre `CREATE FUNCTION`, consulte las instrucciones CREATE PROCEDURSE y CREATE FUNCTION en el Manual de referencia de MySQL 8.0.

El parámetro `max_batch_size` es opcional. De forma predeterminada, el tamaño máximo de lote es 10 000. Puede utilizar este parámetro en la función para restringir el número máximo de entradas procesadas en una solicitud por lotes a IA de SageMaker. El parámetro `max_batch_size` puede ayudar a evitar un error provocado por entradas demasiado grandes o a hacer que IA de SageMaker devuelva una respuesta más rápidamente. Este parámetro afecta al tamaño de un búfer interno utilizado para el procesamiento de solicitudes de IA de SageMaker. Especificar un valor demasiado grande en `max_batch_size` podría provocar una sobrecarga significativa de la memoria en la instancia de base de datos.

Recomendamos que deje la opción `MANIFEST` en su valor predeterminado de `OFF`. Aunque puede utilizar la opción `MANIFEST ON`, algunas características de IA de SageMaker no pueden utilizar directamente el CSV exportado con esta opción. El formato del manifiesto no es compatible con el formato esperado del manifiesto en IA de SageMaker.

Puede crear una función almacenada independiente para cada uno de sus modelos de IA de SageMaker. Esta asignación de funciones a los modelos es obligatoria, porque un punto de conexión se asocia con un modelo específico y cada modelo acepta diferentes parámetros. El uso de tipos de SQL para las entradas del modelo y el tipo de salida del modelo ayuda a evitar errores de conversión de tipos que transfieren datos de manera bidireccional entre los servicios de AWS. Puede controlar quién puede aplicar el modelo. También puede controlar las características de tiempo de ejecución especificando un parámetro que represente el tamaño máximo del lote.

Actualmente, todas las funciones de machine learning de Aurora cuentan con la propiedad `NOT DETERMINISTIC`. Si no especifica dicha propiedad de manera explícita, Aurora establece `NOT`

DETERMINISTIC automáticamente. Este requisito se debe a que el modelo de IA de SageMaker se puede modificar sin ninguna notificación dirigida a la base de datos. Si esto ocurre, las llamadas a una función de machine learning de Aurora podrían devolver resultados distintos para la misma entrada en una única transacción.

No puede utilizar las características CONTAINS SQL, NO SQL, READS SQL DATA o MODIFIES SQL DATA en la instrucción CREATE FUNCTION.

A continuación, se muestra un ejemplo del uso de la invocación de un punto de conexión de IA de SageMaker para detectar anomalías. Hay un punto de conexión de IA de SageMaker random-cut-forest-model. El algoritmo random-cut-forest ya ha entrenado el modelo correspondiente. En cada entrada, el modelo devuelve el origen de una anomalía. En este ejemplo, se muestran los puntos de datos cuya puntuación es superior a 3 desviaciones estándar (aproximadamente el percentil 99,9) con respecto de la puntuación media.

```
CREATE FUNCTION anomaly_score(value real) returns real
  alias aws_sagemaker_invoke_endpoint endpoint name 'random-cut-forest-model-demo';

set @score_cutoff = (select avg(anomaly_score(value)) + 3 * std(anomaly_score(value))
 from nyc_taxi);

select *, anomaly_detection(value) score from nyc_taxi
  where anomaly_detection(value) > @score_cutoff;
```

Requisito del conjunto de caracteres en las funciones de IA de SageMaker que devuelven cadenas

Recomendamos especificar un conjunto de caracteres de utf8mb4 como el tipo de valor devuelto en las funciones de IA de SageMaker que devuelven valores de cadena. Si no resulta útil, utilice una longitud de cadena lo suficientemente grande para que el tipo de valor devuelto retenga un valor representado en el conjunto de caracteres utf8mb4. En el siguiente ejemplo, se muestra cómo declarar el conjunto de caracteres utf8mb4 en la función.

```
CREATE FUNCTION my_ml_func(...) RETURNS VARCHAR(5) CHARSET utf8mb4 ALIAS ...
```

Actualmente, todas las funciones de IA de SageMaker que devuelven una cadena utilizan el conjunto de caracteres utf8mb4 en el valor devuelto. El valor devuelto utiliza este conjunto de caracteres aunque la función de IA de SageMaker declare implícita o explícitamente un conjunto de caracteres diferente para su tipo de valor devuelto. Si la función de IA de SageMaker declara otro conjunto

de caracteres para el valor devuelto, los datos devueltos podrían truncarse inadvertidamente si los almacena en la columna de una tabla que no sea lo suficientemente grande. Por ejemplo, una consulta con una cláusula `DISTINCT` crea una tabla temporal. Así, el resultado de la función de IA de SageMaker se podría truncar debido a la manera en que las cadenas se gestionan internamente durante una consulta.

Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado)

Le recomendamos que comience a usar el machine learning de Aurora e IA de SageMaker utilizando algunos de los algoritmos proporcionados, y que los científicos de datos de su equipo le proporcionen los puntos de conexión de IA de SageMaker que pueda usar con su código SQL. A continuación, encontrará información mínima sobre el uso de su propio bucket de Amazon S3 con sus propios modelos de IA de SageMaker y su clúster de base de datos Aurora MySQL.

El machine learning consta de dos pasos principales: entrenamiento e inferencia. Para entrenar modelos de IA de SageMaker, exporte los datos a un bucket de Amazon S3. Una instancia de cuaderno de Jupyter de IA de SageMaker utiliza el bucket de Amazon S3 para entrenar el modelo antes de que se implemente. La instrucción `SELECT INTO OUTFILE S3` permite consultar datos de un clúster de bases de datos Aurora MySQL y guardarlos directamente en archivos de texto almacenados en un bucket de Amazon S3. A continuación, la instancia de bloc de notas consume los datos del bucket de Amazon S3 para el entrenamiento.

El machine learning de Aurora amplía la sintaxis de `SELECT INTO OUTFILE` existente en Aurora MySQL para exportar los datos a formato CSV. Los modelos que necesitan este formato para el entrenamiento pueden consumir directamente el archivo CSV generado.

```
SELECT * INTO OUTFILE S3 's3_uri' [FORMAT {CSV|TEXT} [HEADER]] FROM table_name;
```

La ampliación admite el formato CSV estándar.

- El formato `TEXT` es el mismo que el formato de exportación de MySQL existente. Este es el formato predeterminado.
- El formato `CSV` es un formato introducido recientemente que sigue la especificación de RFC-4180.
- Si especifique la palabra clave opcional `HEADER`, el archivo de salida contiene una línea de encabezado. Las etiquetas de la línea de encabezado se corresponden con los nombres de las columnas de la instrucción `SELECT`.
- Puede seguir utilizando las palabras claves `CSV` y `HEADER` como identificadores.

La gramática y la sintaxis ampliadas de `SELECT INTO` son ahora las siguientes:

```
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
[FORMAT {CSV|TEXT} [HEADER]]
[{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']
]
```

Consideraciones sobre el rendimiento para utilizar el machine learning de Aurora con Aurora MySQL

Los servicios de Amazon Bedrock, Amazon Comprehend e IA de SageMaker realizan la mayor parte del trabajo cuando se invocan mediante una función de machine learning de Aurora. Esto significa que puede escalar esos recursos según sea necesario, de forma independiente. Para su clúster de base de datos Aurora MySQL, puede hacer que sus llamadas a funciones sean lo más eficientes posible. A continuación, encontrará algunas consideraciones de rendimiento que debe tener en cuenta al trabajar con el machine learning de Aurora.

Modelo y petición

El rendimiento al utilizar Amazon Bedrock depende en gran medida del modelo y de la petición que utilice. Elija un modelo y una petición que sean óptimos para su caso de uso.

Caché de consultas

La caché de consulta de Aurora MySQL no funciona para las funciones de machine learning de Aurora. Aurora MySQL no almacena los resultados de las consultas en la caché de consultas para las sentencias SQL que llaman a las funciones de machine learning de Aurora.

Optimización de lotes para las llamadas a las funciones de machine learning de Aurora

El aspecto principal del rendimiento de machine learning de Aurora en el que puede influir desde el clúster de Aurora es la opción de modo de lote para las llamadas a las funciones almacenadas de machine learning de Aurora. Las funciones de Machine Learning normalmente requieren una

sobrecarga sustancial, lo que hace que sea poco práctico llamar a un servicio externo por separado para cada fila. El machine learning de Aurora puede minimizar esta sobrecarga combinando las llamadas al servicio de machine learning de Aurora externo para muchas filas en un solo lote. El machine learning de Aurora recibe las respuestas de un lote de filas de entrada y, luego, devuelve las respuestas a la consulta en ejecución una fila a la vez. Esta optimización mejora el rendimiento y la latencia de las consultas de Aurora sin modificar los resultados.

Al crear una función almacenada de Aurora conectada a un punto de conexión de IA de SageMaker, se define el parámetro de tamaño del lote. Este parámetro influye en el número de filas que se transfieren en cada llamada subyacente a IA de SageMaker. En las consultas que procesan un número elevado de filas, la sobrecarga para realizar una llamada de IA de SageMaker independiente en cada fila puede ser significativa. Cuanto más grande sea el conjunto de datos procesado por el procedimiento almacenado, mayor puede ser el tamaño del lote.

Puede comprobar si se puede aplicar la optimización del modo de lote a una función de IA de SageMaker consultando el plan de consulta generado por la instrucción `EXPLAIN PLAN`. En este caso, la columna `extra` del plan de ejecución incluye `Batched machine learning`. En el siguiente ejemplo, se muestra una llamada a una función de IA de SageMaker que utiliza el modo de lote.

```
mysql> CREATE FUNCTION anomaly_score(val real) returns real alias
 aws_sagemaker_invoke_endpoint endpoint name 'my-rcf-model-20191126';
Query OK, 0 rows affected (0.01 sec)

mysql> explain select timestamp, value, anomaly_score(value) from nyc_taxi;
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows | filtered | Extra                   |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
|  1 | SIMPLE      | nyc_taxi | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |   48 |   100.00 | Batched machine learning |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
1 row in set, 1 warning (0.01 sec)
```

Al llamar a una de las funciones de Amazon Comprehend integradas, puede controlar el tamaño del lote especificando el parámetro `max_batch_size` opcional. Este parámetro restringe el número máximo de valores `input_text` procesados en cada lote. El envío de varios elementos a la vez

reduce el número de recorridos de ida y vuelta entre Aurora y Amazon Comprehend. Resulta útil limitar el tamaño del lote en situaciones como una consulta con una cláusula `LIMIT`. Al utilizar un valor pequeño para `max_batch_size`, puede evitar invocar Amazon Comprehend más veces que los textos de entrada que tenga.

La optimización de lotes para la evaluación de funciones de machine learning de Aurora se aplica en los siguientes casos:

- Llamadas a funciones de la lista de selección o la cláusula `WHERE` de instrucciones `SELECT`
- Llamadas a funciones presentes en la lista `VALUES` de instrucciones `INSERT` y `REPLACE`
- Funciones de IA de SageMaker en valores `SET` de instrucciones `UPDATE`:

```
INSERT INTO MY_TABLE (col1, col2, col3) VALUES
  (ML_FUNC(1), ML_FUNC(2), ML_FUNC(3)),
  (ML_FUNC(4), ML_FUNC(5), ML_FUNC(6));
UPDATE MY_TABLE SET col1 = ML_FUNC(col2), SET col3 = ML_FUNC(col4) WHERE ...;
```

Monitorización del machine learning de Aurora

Puede monitorear las operaciones por lotes de machine learning de Aurora consultando varias variables globales, como en el siguiente ejemplo.

```
show status like 'Aurora_ml%';
```

Puede restablecer las variables de estado utilizando una instrucción `FLUSH STATUS`. Así, todas las cifras representan los valores totales, los promedios, etc., desde la última vez que se restableció la variable.

`Aurora_ml_logical_request_cnt`

La cantidad de solicitudes lógicas que la instancia de base de datos ha evaluado para enviarse a los servicios de machine learning de Aurora desde el último restablecimiento de estado. Dependiendo de si se ha utilizado el procesamiento por lotes, este valor puede ser superior a `Aurora_ml_actual_request_cnt`.

`Aurora_ml_logical_response_cnt`

El recuento acumulado de respuestas que recibe Aurora MySQL de los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos.

`Aurora_ml_actual_request_cnt`

El recuento acumulado de solicitudes que hace Aurora MySQL a los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos.

`Aurora_ml_actual_response_cnt`

El recuento acumulado de respuestas que recibe Aurora MySQL de los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos.

`Aurora_ml_cache_hit_cnt`

El número acumulado de aciertos de la caché interna que Aurora MySQL recibe de los servicios de machine learning de Aurora en todas las consultas ejecutadas por usuarios de la instancia de base de datos.

`Aurora_ml_retry_request_cnt`

La cantidad de solicitudes reintentadas que la instancia de base de datos ha enviado a los servicios de machine learning de Aurora desde el último restablecimiento de estado.

`Aurora_ml_single_request_cnt`

El número acumulado de funciones de machine learning de Aurora evaluadas por un modo distinto al modo de lote en todas las consultas ejecutadas por usuarios de la instancia de base de datos.

Para obtener información acerca del monitoreo del rendimiento de las operaciones de SageMaker llamadas desde las funciones del machine learning de Aurora, consulte Monitoreo de IA de Amazon SageMaker.

Uso de machine learning de Amazon Aurora con Aurora PostgreSQL

Al utilizar el machine learning de Amazon Aurora con su clúster de base de datos de Aurora PostgreSQL, puede utilizar Amazon Comprehend, IA de Amazon SageMaker o Amazon Bedrock, según sus necesidades. Cada uno de estos servicios admite casos de uso de machine learning específicos.

El machine learning de Aurora se admite en determinadas Regiones de AWS y solo para versiones de Aurora PostgreSQL. Antes de intentar configurar el machine learning de Aurora, compruebe la disponibilidad para su versión de Aurora PostgreSQL y su región. Para obtener más información, consulte Machine learning de Aurora con Aurora PostgreSQL.

Temas

- Requisitos para usar machine learning de Aurora con Aurora PostgreSQL
- Funciones y limitaciones compatibles del machine learning de Aurora con Aurora PostgreSQL
- Configuración del clúster de base de datos Aurora PostgreSQL para utilizar el machine learning de Aurora
- Uso de Amazon Bedrock con el clúster de base de datos de Aurora PostgreSQL
- Uso de Amazon Comprehend con el clúster de base de datos de Aurora PostgreSQL
- Uso de IA de SageMaker con el clúster de base de datos de Aurora PostgreSQL
- Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado)
- Consideraciones sobre el rendimiento para utilizar el machine learning de Aurora con Aurora PostgreSQL
- Monitorización del machine learning de Aurora

Requisitos para usar machine learning de Aurora con Aurora PostgreSQL

Los servicios de machine learning de AWS son servicios administrados que se configuran y ejecutan en sus propios entornos de producción. El machine learning de Aurora admite la integración con Amazon Comprehend, IA de SageMaker y Amazon Bedrock. Antes de intentar configurar el clúster de base de datos de Aurora PostgreSQL para usar el machine learning de Aurora, asegúrese de comprender los siguientes requisitos y requisitos previos.

- Los servicios Amazon Comprehend, IA de SageMaker y Amazon Bedrock deben ejecutarse en la misma Región de AWS que el clúster de base de datos de Aurora PostgreSQL. No puede usar los servicios de Amazon Comprehend, IA de SageMaker o Amazon Bedrock desde un clúster de base de datos de Aurora PostgreSQL en una región diferente.
- Si su clúster de base de datos de Aurora PostgreSQL se encuentra en una nube pública virtual (VPC) diferente basada en el servicio Amazon VPC que los servicios Amazon Comprehend e IA de SageMaker, el grupo de seguridad de la VPC debe permitir las conexiones salientes al servicio de machine learning de Aurora de destino. Para obtener más información, consulte Habilitación de la comunicación de red desde Amazon Aurora a otros servicios de AWS.
- Para IA de SageMaker, los componentes de machine learning que desee usar para las inferencias deben estar configurados y listos para usarse. Durante el proceso de configuración del clúster de base de datos Aurora PostgreSQL, debe tener disponible el nombre de recurso de Amazon (ARN) del punto de conexión de IA de SageMaker. Es probable que los científicos de datos de su equipo sean los más capacitados para trabajar con IA de SageMaker para preparar los modelos y gestionar otras tareas similares. Para empezar a utilizar IA de Amazon SageMaker, consulte Get Started with Amazon SageMaker AI. Para obtener más información sobre inferencias y puntos de conexión, consulte Inferencia en tiempo real.
- Para Amazon Bedrock, debe tener disponible el ID de modelo de los modelos de Bedrock que desee utilizar para las inferencias durante el proceso de configuración del clúster de base de datos de Aurora PostgreSQL. Es probable que los científicos de datos de su equipo sean los más capacitados para trabajar con Bedrock y decidir qué modelos utilizar, ajustarlos si es necesario y gestionar otras tareas similares. Para empezar a usar Amazon Bedrock, consulte Cómo configurar Bedrock.
- Los usuarios de Amazon Bedrock deben solicitar acceso a los modelos para que estén disponibles para su uso. Si desea agregar modelos adicionales para la generación de texto, chat e imágenes, debe solicitar el acceso a los modelos de Amazon Bedrock. Para obtener más información, consulte Acceso a modelos.

Funciones y limitaciones compatibles del machine learning de Aurora con Aurora PostgreSQL

El machine learning de Aurora admite los puntos de conexión de IA de SageMaker que puedan leer y escribir el formato de valores separados por comas (CSV), a través de un valor `ContentType` de `text/csv`. Los algoritmos incorporados de IA de SageMaker que actualmente aceptan este formato son los siguientes.

- Linear Learner
- Random Cut Forest
- XGBoost

  Para obtener más información sobre estos algoritmos, consulte Choose an Algorithm en la Guía para desarrolladores de IA de Amazon SageMaker.

Al usar Amazon Bedrock con el machine learning de Aurora, se aplican las siguientes limitaciones:

- Las funciones definidas por el usuario (UDF) proporcionan una forma nativa de interactuar con Amazon Bedrock. Las UDF no tienen requisitos específicos de solicitud o respuesta, por lo que pueden utilizar cualquier modelo.
- Puede utilizar las UDF para crear cualquier flujo de trabajo que desee. Por ejemplo, puede combinar primitivas básicas, como `pg_cron`, para ejecutar una consulta, obtener datos, generar inferencias y escribir en tablas para servir las consultas directamente.
- Las UDF no admiten llamadas por lotes o paralelas.
- La extensión de Aurora Machine Learning no admite interfaces vectoriales. Como parte de la extensión, hay una función disponible para generar las incrustaciones de la respuesta del modelo en el formato `float8[]` para almacenar esas incrustaciones en Aurora. Para obtener más información sobre el uso de `float8[]`, consulte Uso de Amazon Bedrock con el clúster de base de datos de Aurora PostgreSQL.

Configuración del clúster de base de datos Aurora PostgreSQL para utilizar el machine learning de Aurora

Para que el machine learning de Aurora funcione con su clúster de base de datos de Aurora PostgreSQL, debe crear un rol de AWS Identity and Access Management (IAM) para cada uno de los servicios que desee utilizar. El rol de IAM permite que el clúster de base de datos de Aurora PostgreSQL utilice el servicio de machine learning de Aurora en nombre del clúster. También debe instalar la extensión de machine learning de Aurora. En los siguientes temas, puede encontrar los procedimientos de configuración para cada uno de estos servicios de machine learning de Aurora.

Temas

- Configuración de Aurora PostgreSQL para usar Amazon Bedrock
- Configuración de Aurora PostgreSQL para usar Amazon Comprehend

- Configuración de Aurora PostgreSQL para usar IA de Amazon SageMaker
  - Configuración de Aurora PostgreSQL para usar Amazon S3 para IA de SageMaker (avanzado)
- Instalación de la extensión de machine learning de Aurora

Configuración de Aurora PostgreSQL para usar Amazon Bedrock

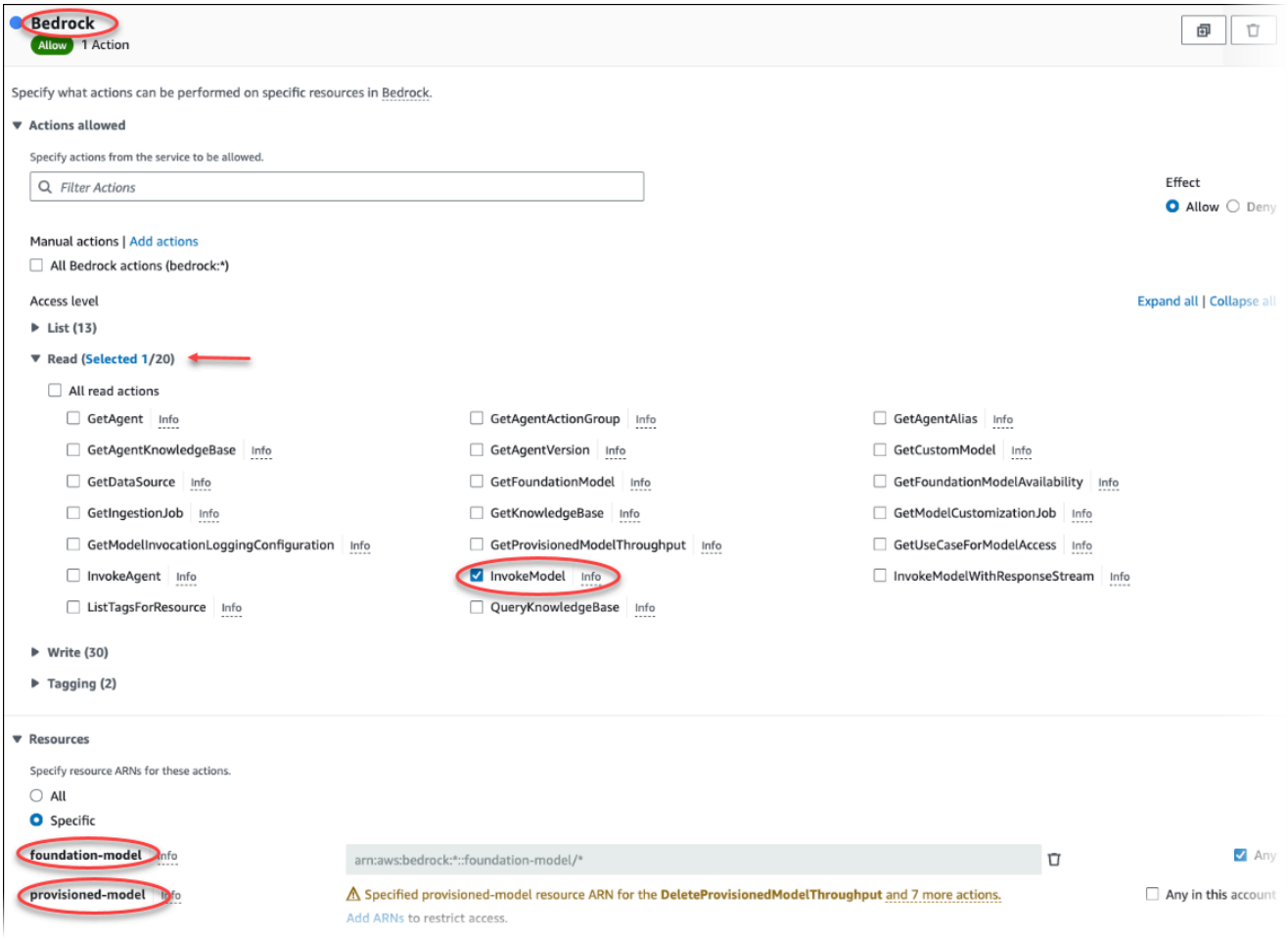
En el procedimiento siguiente, primero debe crear la política y el rol de IAM que otorgan a Aurora PostgreSQL el permiso para usar Amazon Bedrock en nombre del clúster. A continuación, adjunte la política a un rol de IAM que el clúster de base de datos Aurora PostgreSQL utilice para trabajar con Amazon Bedrock. Por motivos de simplicidad, este procedimiento utiliza la Consola de administración de AWS para completar todas las tareas.

Para configurar el clúster de base de datos de Aurora PostgreSQL para utilizar Amazon Bedrock

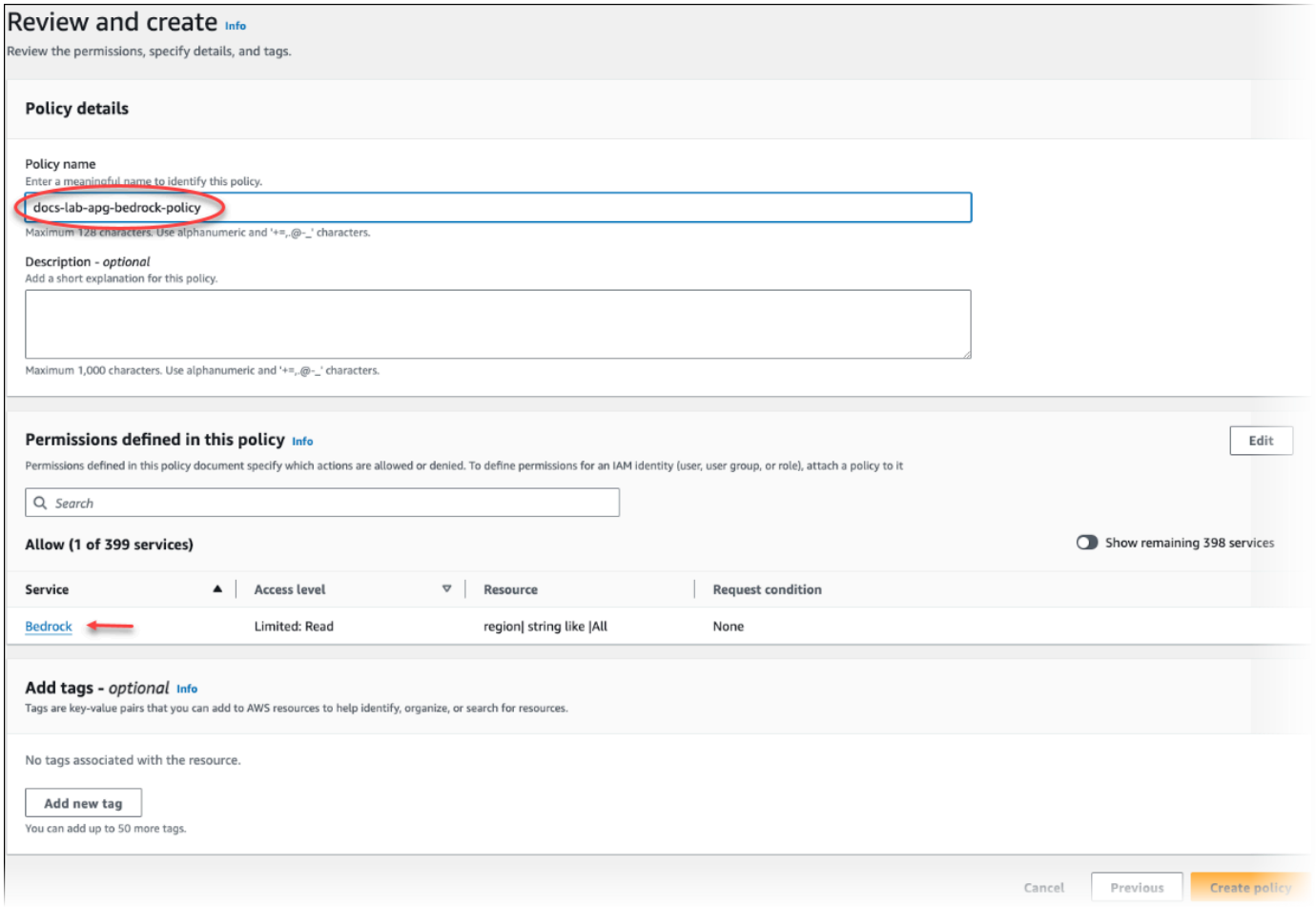
1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
3. Elija Políticas (en Administración de acceso) en el menú de la consola de AWS Identity and Access Management (IAM).

   a. Seleccione Crear política. En la página del editor visual, elija Servicio y, a continuación, escriba Bedrock en el campo de selección de servicio. Expanda el nivel de acceso de lectura. Elija InvokeModel en la configuración de lectura de Amazon Bedrock.

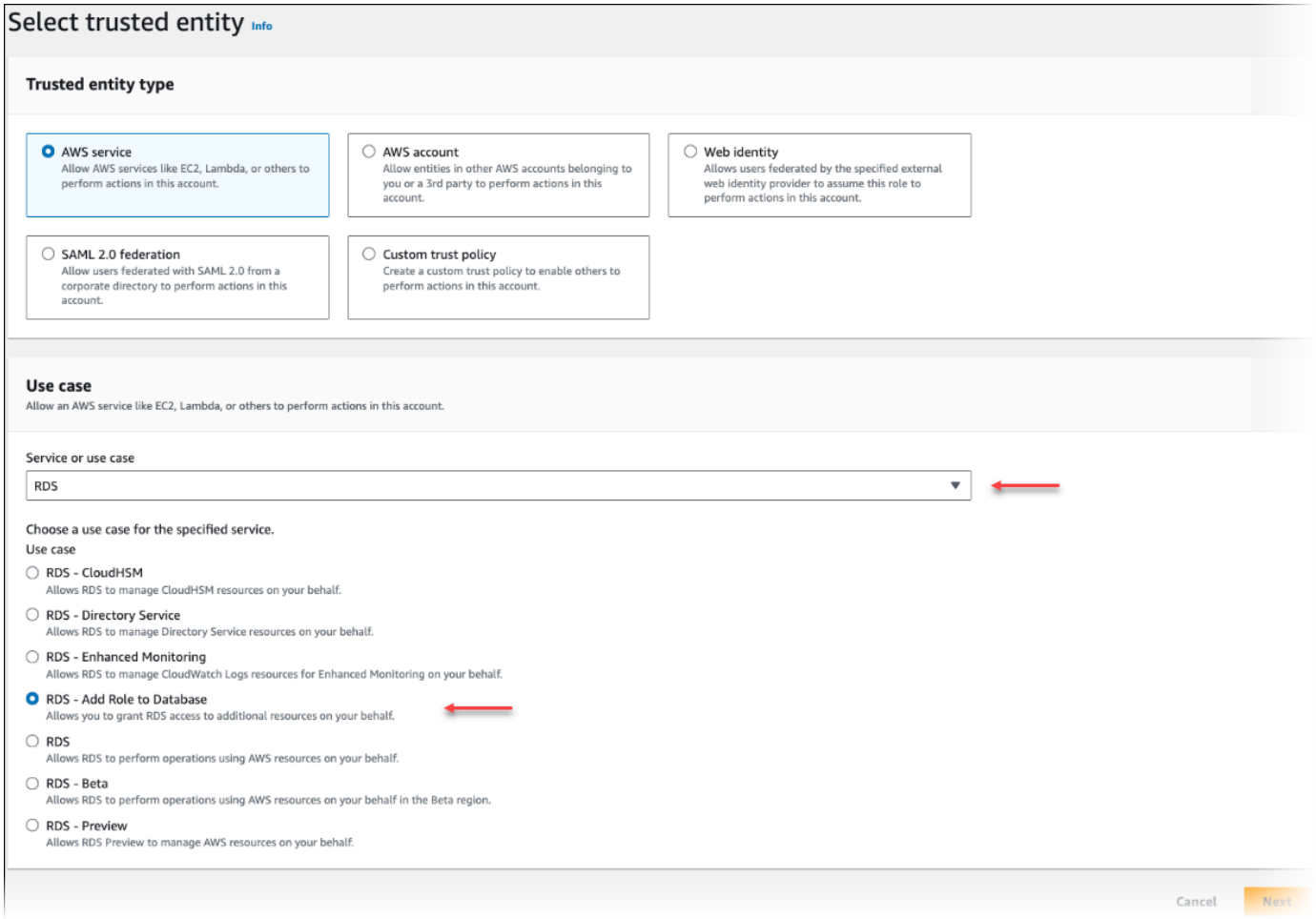
   b. Elija el modelo fundacional o aprovisionado al que desee conceder el acceso de lectura a través de la política.



4. Elija Siguiente: Etiquetas y defina las etiquetas que desee (esto es opcional). Elija Siguiente: Revisar. Introduzca un nombre para la política y una descripción, como se muestra en la imagen.

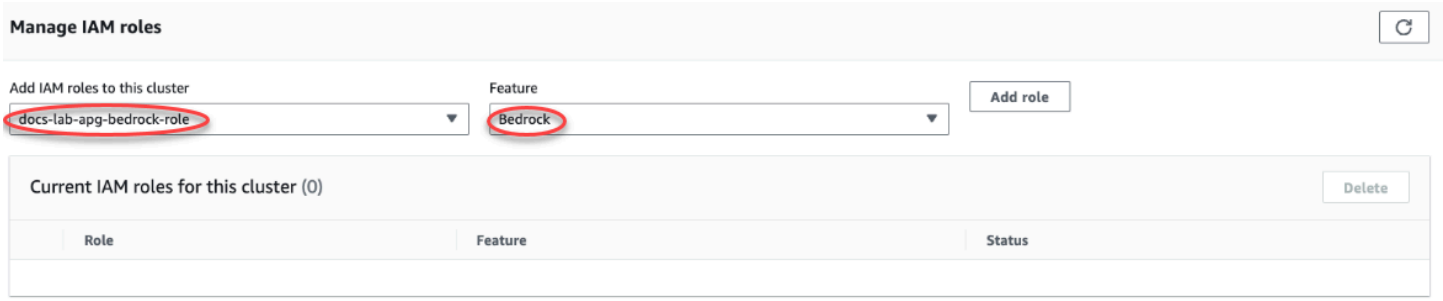


5. Seleccione Crear política. La consola muestra una alerta cuando se guarda la política. Puede encontrarla en la lista de políticas.
6. Elija Roles (en Administración de acceso) en el menú de la consola de IAM.
7. Seleccione Crear rol.
8. En la página Seleccionar entidad de confianza, elija el mosaico de servicio de AWS y, a continuación, elija RDS para abrir el selector.
9. Elija RDS: Añadir rol a la base de datos.



10. Elija Siguiente. En la página Añadir permisos, busque la política que creó en el paso anterior y la elija de entre las de la lista. Elija Siguiente.
11. Siguiente: Revisar. Introduzca un nombre y la descripción del rol de IAM.
12. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
13. Vaya hasta la Región de AWS donde se encuentra el clúster de base de datos de Aurora PostgreSQL.
14. En el panel de navegación, elija Bases de datos y, a continuación, elija el clúster de base de datos Aurora PostgreSQL que desea utilizar con Bedrock.
15. En la pestaña Conectividad y seguridad y desplácese hasta la sección Administrar roles de IAM de la página. En el selector Añadir roles de IAM a este clúster, elija el rol que creó en los pasos anteriores. En el selector Característica, elija Bedrock y, a continuación, seleccione Añadir rol.

El rol (con su política) está asociado al clúster de base de datos de Aurora PostgreSQL. Cuando el proceso se completa, el rol aparece en la lista de roles de IAM actuales para este clúster, como se muestra a continuación.



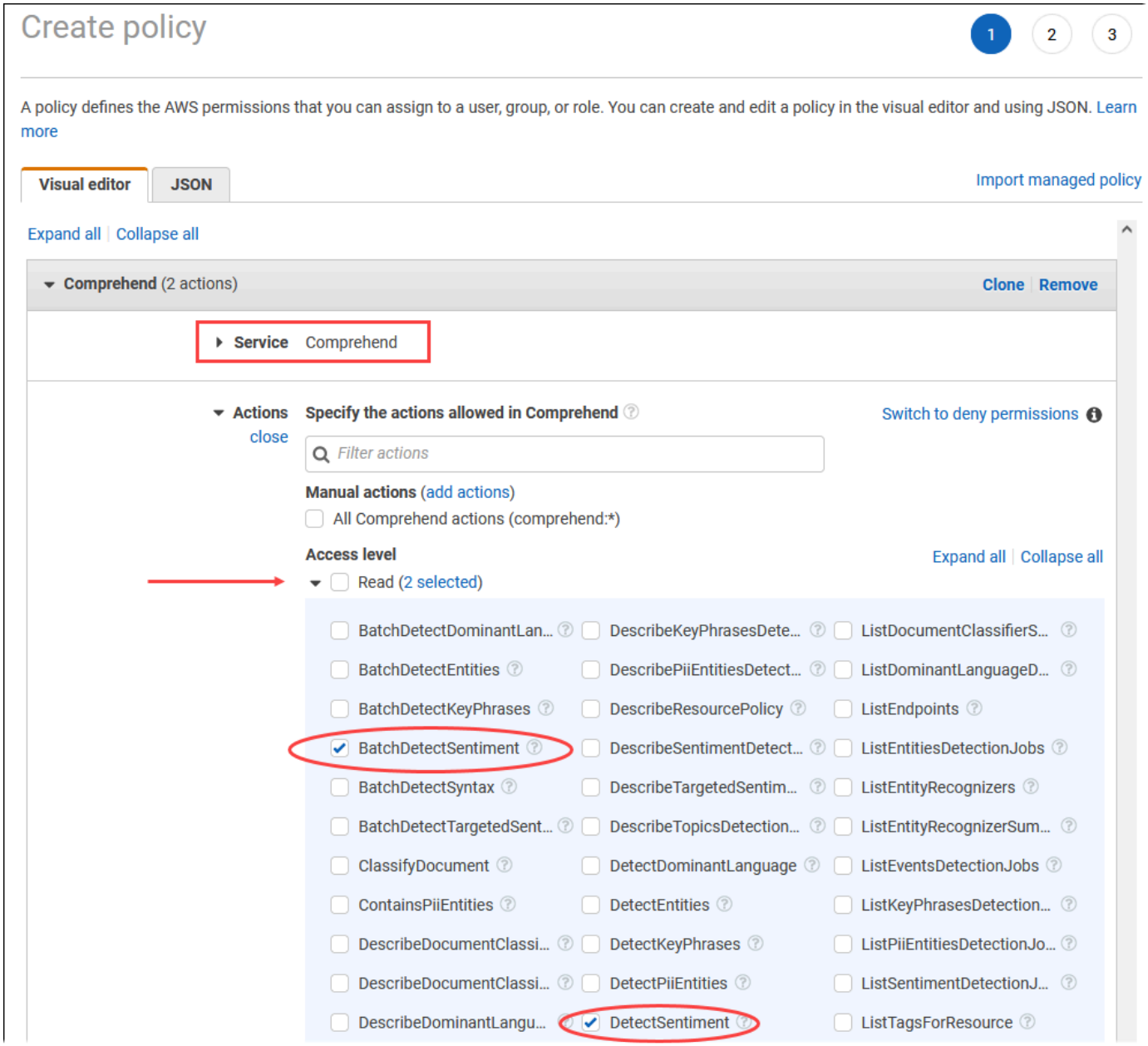
La configuración de IAM para Amazon Bedrock está completa. Siga configurando Aurora PostgreSQL para que funcione con el machine learning de Aurora. Para ello, instale la extensión tal y como se detalla en Instalación de la extensión de machine learning de Aurora

Configuración de Aurora PostgreSQL para usar Amazon Comprehend

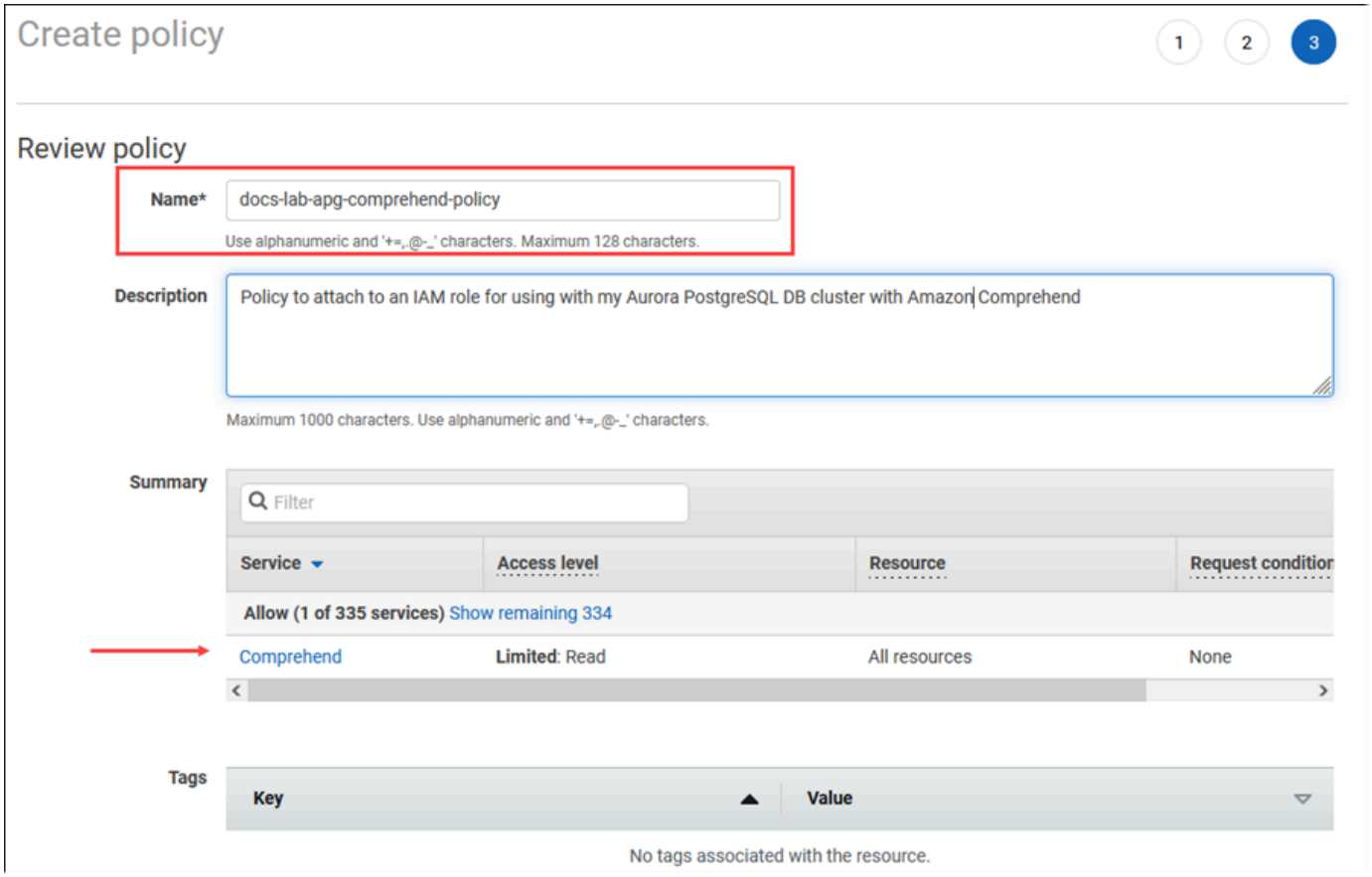
En el procedimiento siguiente, primero debe crear la política y el rol de IAM que otorgan a Aurora PostgreSQL el permiso para usar Amazon Comprehend en nombre del clúster. A continuación, adjunte la política a un rol de IAM que el clúster de base de datos de Aurora PostgreSQL utiliza para trabajar con Amazon Comprehend. Por motivos de simplicidad, este procedimiento utiliza la Consola de administración de AWS para completar todas las tareas.

Para configurar el clúster de base de datos de Aurora PostgreSQL para utilizar Amazon Comprehend

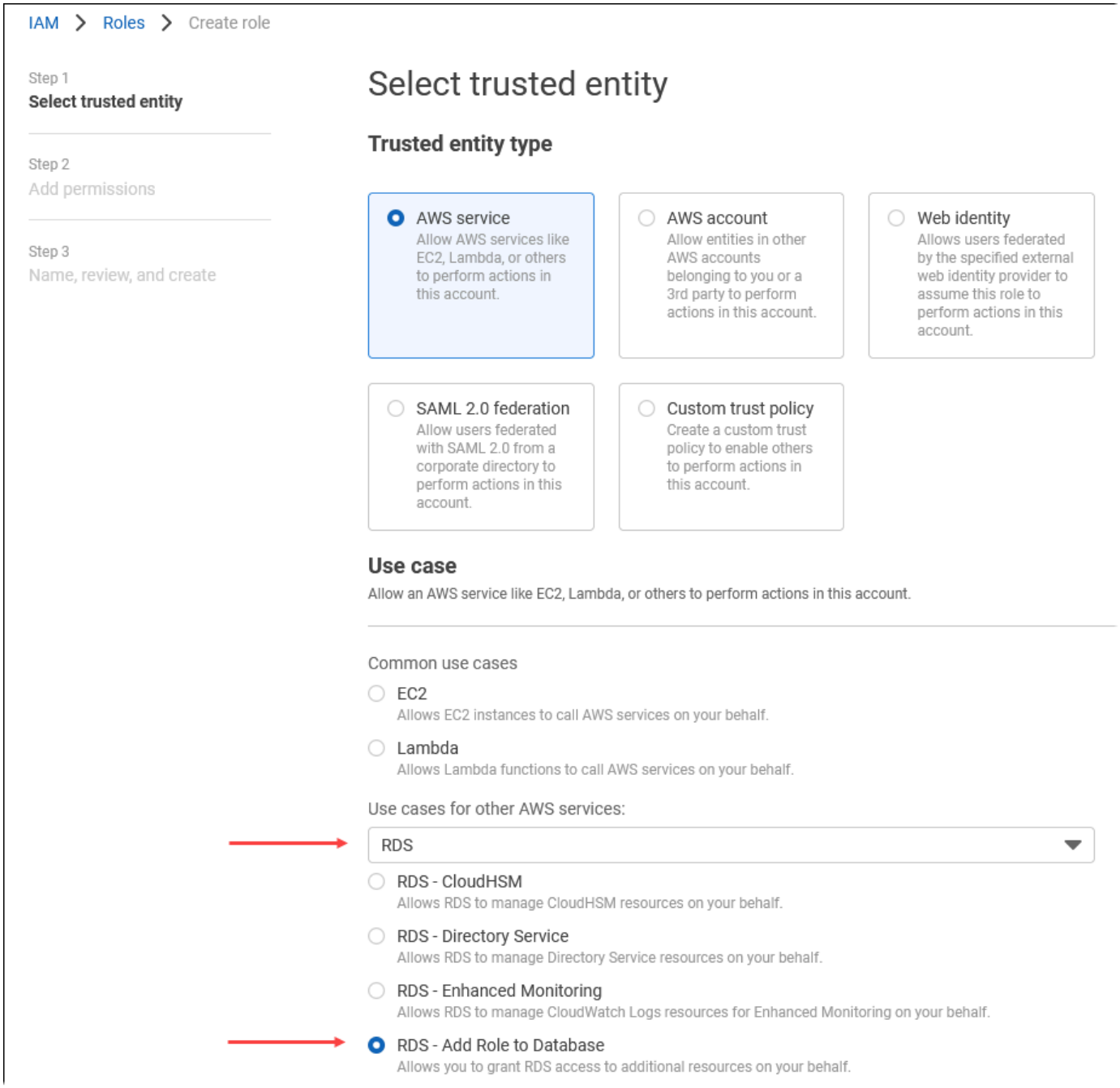
1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
3. Elija Policies (Políticas) (en Administración de acceso) en el menú de la consola de AWS Identity and Access Management (IAM).



4. Seleccione Crear política. En la página del editor visual, elija Service (Servicio) y, a continuación, escriba Comprehend en el campo de selección de servicio. Expanda el nivel de acceso de lectura. Elija BatchDetectSentiment y DetectSentiment en la configuración de lectura de Amazon Comprehend
5. Elija Next: Tags (Siguiente: Etiquetas)y defina las etiquetas que desee (esto es opcional). Elija Siguiente: Revisar. Introduzca un nombre para la política y una descripción, como se muestra en la imagen.



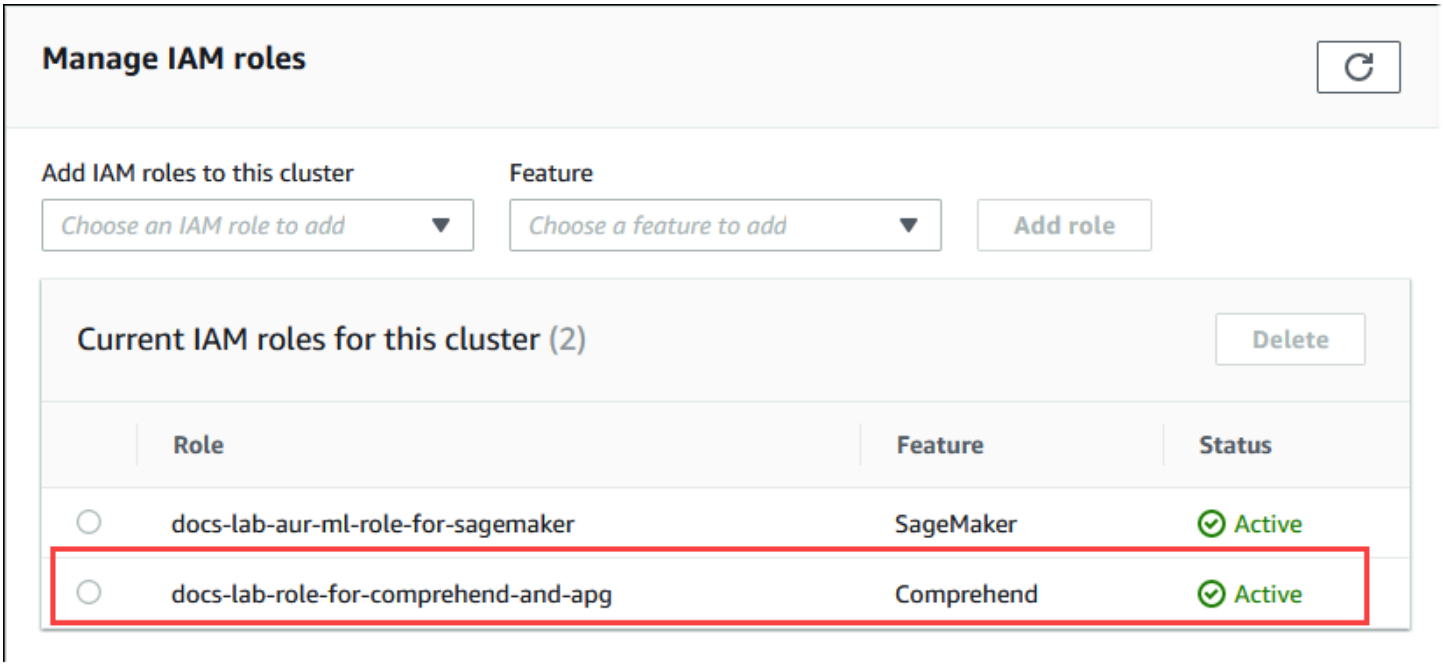
6. Elija Crear política. La consola muestra una alerta cuando se guarda la política. Puede encontrarla en la lista de políticas.
7. Elija Roles (en Administración de acceso) en el menú de la consola de IAM.
8. Seleccione Crear rol.
9. En la página Seleccionar entidad de confianza, elija el mosaico de servicio de AWS y, a continuación, elija RDS para abrir el selector.
10. Elija RDS – Add Role to Database (RDS: Añadir rol a la base de datos).



11. Elija Siguiente. En la página Añadir permisos, busque la política que creó en el paso anterior y la elija de entre las de la lista. Elija Siguiente.
12. Siguiente: Revisar. Introduzca un nombre y la descripción del rol de IAM.
13. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
14. Vaya hasta la Región de AWS donde se encuentra el clúster de base de datos de Aurora PostgreSQL.

15. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija el clúster de base de datos Aurora PostgreSQL que desea utilizar con Amazon Comprehend.
16. En la pestaña Connectivity & Security (Conectividad y seguridad) y desplácese hasta la sección Manage IAM roles (Administrar roles de IAM) de la página. En el selector Añadir roles de IAM a este clúster, elija el rol que creó en los pasos anteriores. En el selector Característica, elija Comprehend y, a continuación, seleccione Agregar rol.

El rol (con su política) está asociado al clúster de base de datos de Aurora PostgreSQL. Cuando el proceso se completa, el rol aparece en la lista de roles de IAM actuales para este clúster, como se muestra a continuación.



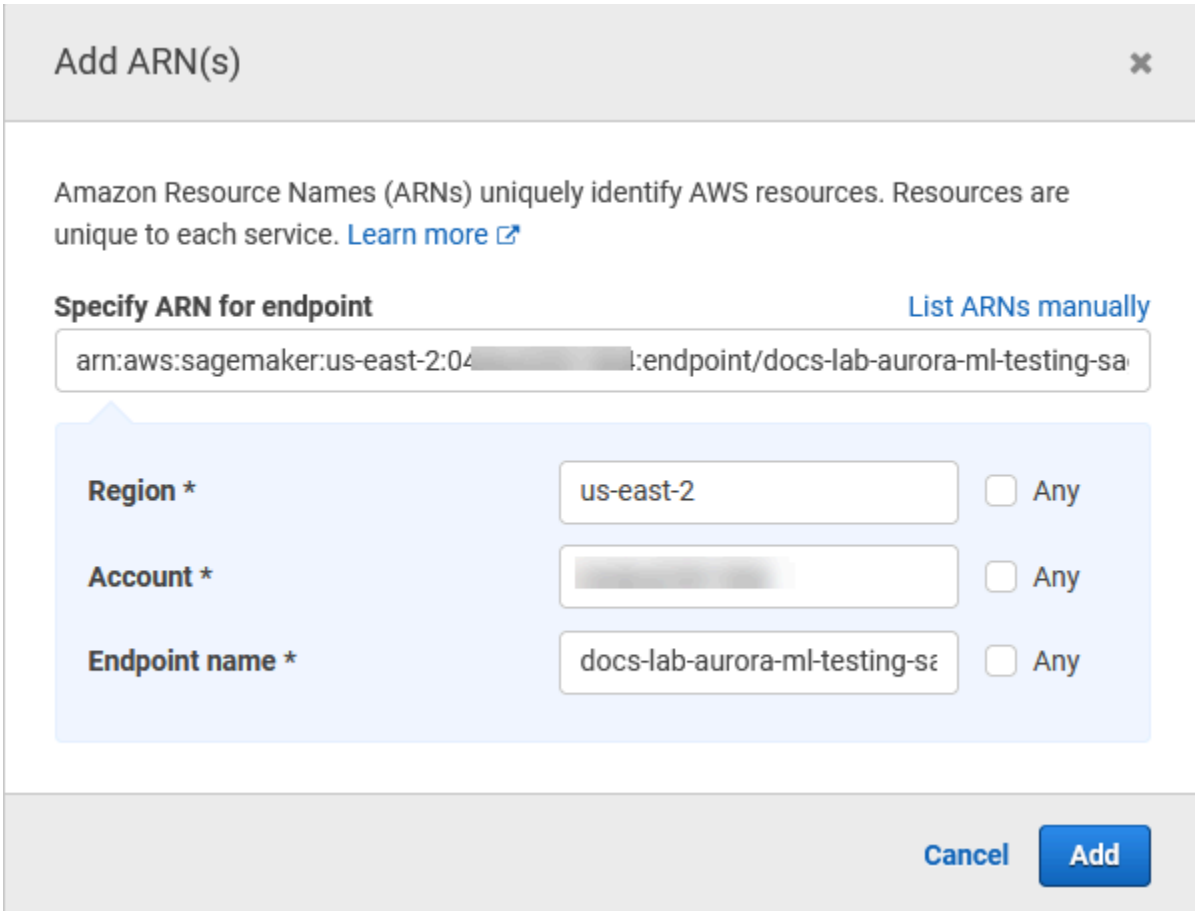
La configuración de IAM para Amazon Comprehend está completa. Siga configurando Aurora PostgreSQL para que funcione con el machine learning de Aurora. Para ello, instale la extensión tal y como se detalla en Instalación de la extensión de machine learning de Aurora

Configuración de Aurora PostgreSQL para usar IA de Amazon SageMaker

Antes de poder crear la política y el rol de IAM para su clúster de base de datos de Aurora PostgreSQL, debe tener la configuración del modelo de IA de SageMaker y su punto de conexión disponible.

Configuración del clúster de base de datos Aurora PostgreSQL para que utilice IA de SageMaker

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Elija Policies (Políticas) (en Administración de acceso) en el menú de la consola de AWS Identity and Access Management (IAM) y, a continuación, elija Create policy (Crear política). En el editor visual, elija SageMaker para el servicio. Para Actions (Acciones), abra el selector de lectura (en el nivel de acceso) y elija InvokeEndpoint. Al hacer esto, aparece un icono de advertencia.
3. Abra el selector de recursos y elija el enlace Agregar ARN para restringir el acceso en la sección Especificar el ARN del recurso de punto de conexión para la acción InvokeEndpoint.
4. Introduzca la Región de AWS de los recursos de IA de SageMaker y el nombre de su punto de conexión. Su cuenta de AWS ya está precargada.



5. Seleccione Añadir para guardar. Elija Siguiente: Etiquetas y Siguiente: Revisar para ir a la última página del proceso de creación de políticas.

6. Escriba un nombre y una descripción para esta política y, a continuación, elija Crear política. La política se crea y se añade a la lista de políticas. Verá una alerta en la consola cuando esto ocurre.
7. En la consola de IAM, seleccione Roles.
8. Seleccione Crear rol.
9. En la página Seleccionar entidad de confianza, elija el mosaico de servicio de AWS y, a continuación, elija RDS para abrir el selector.
10. Elija RDS: Añadir rol a la base de datos.
11. Elija Siguiente. En la página Añadir permisos, busque la política que creó en el paso anterior y la elija de entre las de la lista. Elija Siguiente.
12. Siguiente: Revisar. Introduzca un nombre y la descripción del rol de IAM.
13. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
14. Vaya hasta la Región de AWS donde se encuentra el clúster de base de datos de Aurora PostgreSQL.
15. En el panel de navegación, elija Bases de datos y, a continuación, elija el clúster de base de datos Aurora PostgreSQL que desea utilizar con IA de SageMaker.
16. En la pestaña Conectividad y seguridad y desplácese hasta la sección Administrar roles de IAM de la página. En el selector Añadir roles de IAM a este clúster, elija el rol que creó en los pasos anteriores. En el selector de Característica, elija IA de SageMaker y después seleccione Añadir rol.

El rol (con su política) está asociado al clúster de base de datos de Aurora PostgreSQL. Cuando el proceso se completa, el rol aparece en la lista de roles de IAM actuales para esta lísta de clúster.

La configuración de IAM para IA de SageMaker está completa. Siga configurando Aurora PostgreSQL para que funcione con el machine learning de Aurora. Para ello, instale la extensión tal y como se detalla en Instalación de la extensión de machine learning de Aurora

Configuración de Aurora PostgreSQL para usar Amazon S3 para IA de SageMaker (avanzado)

Para utilizar IA de SageMaker con sus propios modelos en lugar de utilizar los componentes prediseñados que ofrece IA de SageMaker, debe configurar un bucket de Amazon Simple Storage Service (Amazon S3) para que lo utilice el clúster de base de datos Aurora PostgreSQL. Se trata de un tema avanzado y no está completamente documentado en esta Guía del usuario de Amazon

Aurora. El proceso general es el mismo que para integrar el soporte para IA de SageMaker, de la siguiente manera.

1. Cree el rol y la política de IAM para Amazon S3.
2. Agregue el rol de IAM y la importación o exportación de Amazon S3 como una característica en la pestaña Conectividad y seguridad de su clúster de base de datos de Aurora PostgreSQL.
3. Añada el ARN del rol a su grupo de parámetros de clúster de base de datos personalizado para su clúster de base de datos de Aurora.

Para obtener información de uso básica, consulte Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado).

Instalación de la extensión de machine learning de Aurora

Las extensiones `aws_ml 1.0` de machine learning de Aurora proporcionan dos funciones que puede usar para invocar los servicios Amazon Comprehend e IA de SageMaker, y `aws_ml 2.0` proporciona dos funciones adicionales que puede usar para invocar los servicios de Amazon Bedrock. La instalación de estas extensiones en el clúster de base de datos de Aurora PostgreSQL también crea una función administrativa para la característica.

> **Note**
>
> El uso de estas funciones depende de que la configuración de IAM para el servicio de machine learning de Aurora (Amazon Comprehend, IA de SageMaker, Amazon Bedrock) esté completa, tal como se detalla en Configuración del clúster de base de datos Aurora PostgreSQL para utilizar el machine learning de Aurora.

- aws_comprehend.detect_sentiment: utilice esta función para aplicar el análisis de opinión al texto almacenado en la base de datos de su clúster de base de datos de Aurora PostgreSQL.
- aws_sagemaker.invoke_endpoint: esta función se utiliza en el código SQL para comunicarse con el punto de conexión de IA de SageMaker desde el clúster.
- aws_bedrock.invoke_model: esta función se utiliza en el código SQL para comunicarse con los modelos de Bedrock del clúster. La respuesta de esta función tendrá el formato TEXT, por lo que si un modelo responde en el formato de un cuerpo JSON, el resultado de esta función se retransmitirá en formato de cadena al usuario final.

- aws_bedrock.invoke_model_get_embeddings: esta función se utiliza en el código SQL para invocar los modelos Bedrock, que devuelven incrustaciones de salida dentro de una respuesta JSON. Esto se puede aprovechar cuando desee extraer las incrustaciones directamente asociadas a la json-key para optimizar la respuesta con cualquier flujo de trabajo autoadministrado.

Para instalar la extensión de machine learning de Aurora en su clúster de bases de datos de Aurora PostgreSQL

- Use `psql` para conectarse a la instancia de escritor de su clúster de base de datos de Aurora PostgreSQL. Conecte a la base de datos específica en la que desee instalar la extensión `aws_ml`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=labdb
```

```
labdb=> CREATE EXTENSION IF NOT EXISTS aws_ml CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
labdb=>
```

La instalación de las extensiones `aws_ml` también crea el rol administrativo `aws_ml` y tres esquemas nuevos, tal como se indica a continuación.

- `aws_comprehend`: esquema del servicio de Amazon Comprehend y origen de la función `detect_sentiment` (`aws_comprehend.detect_sentiment`).
- `aws_sagemaker`: esquema del servicio de IA de SageMaker y origen de la función `invoke_endpoint` (`aws_sagemaker.invoke_endpoint`).
- `aws_bedrock`: esquema del servicio de Amazon Bedrock y origen de las funciones `invoke_model(aws_bedrock.invoke_model)` y `invoke_model_get_embeddings(aws_bedrock.invoke_model_get_embeddings)`.

Al rol `rds_superuser` se le concede el rol administrativo `aws_ml` y se crea el `OWNER` de estos tres esquemas de machine learning de Aurora. Para permitir que otros usuarios de bases de datos accedan a las funciones de machine learning de Aurora, el `rds_superuser` tiene que conceder privilegios `EXECUTE` en las funciones de machine learning de Aurora. De forma predeterminada,

los privilegios `EXECUTE` se revocan de `PUBLIC` en las funciones de los dos esquemas de machine learning de Aurora.

En una configuración de base de datos multiusuario, puede evitar que los inquilinos accedan a las funciones de machine learning de Aurora utilizando `REVOKE USAGE` en esquema de machine learning de Aurora específico que desee proteger.

Uso de Amazon Bedrock con el clúster de base de datos de Aurora PostgreSQL

Para Aurora PostgreSQL, el machine learning de Aurora proporciona la siguiente función de Amazon Bedrock para trabajar con los datos de texto. Esta función solo está disponible después de instalar la extensión `aws_ml` 2.0 y completar todos los procedimientos de configuración. Para obtener más información, consulte Configuración del clúster de base de datos Aurora PostgreSQL para utilizar el machine learning de Aurora.

aws_bedrock.invoke_model

Esta función toma texto formateado en JSON como entrada y lo procesa para diversos modelos alojados en Amazon Bedrock y recupera la respuesta de texto JSON del modelo. Esta respuesta puede contener texto, imagen o incrustaciones. A continuación se presenta un resumen de la documentación de la función.

```
aws_bedrock.invoke_model(
   IN model_id      varchar,
   IN content_type  text,
   IN accept_type   text,
   IN model_input   text,
   OUT model_output varchar)
```

Las entradas y salidas de esta función son las siguientes.

- `model_id`: identificador del modelo.
- `content_type`: el tipo de solicitud al modelo de Bedrock.
- `accept_type`: el tipo de respuesta que cabe esperar del modelo de Bedrock. Por lo general, application/JSON para la mayoría de los modelos.
- `model_input`: peticiones; un conjunto específico de entradas al modelo en el formato especificado por content_type. Para obtener más información sobre el formato o la estructura

de las solicitudes que acepta el modelo, consulte Parámetros de inferencia para modelos fundacionales.

- `model_output`: la salida del modelo de Bedrock como texto.

El siguiente ejemplo muestra cómo invocar un modelo de Anthropic Claude 2 para Bedrock mediante invoke_model.

Example Ejemplo: Una consulta sencilla con las funciones de Amazon Bedrock

```
SELECT aws_bedrock.invoke_model (
    model_id    := 'anthropic.claude-v2',
    content_type:= 'application/json',
    accept_type := 'application/json',
    model_input := '{"prompt": "\n\nHuman: You are a helpful assistant that answers
 questions directly and only using the information provided in the context below.
\nDescribe the answer
    in detail.\n\nContext: %s \n\nQuestion: %s \n
\nAssistant:","max_tokens_to_sample":4096,"temperature":0.5,"top_k":250,"top_p":0.5,"stop_seque
[]}'
);
```

aws_bedrock.invoke_model_get_embeddings

La salida del modelo puede apuntar a incrustaciones vectoriales en algunos casos. Dado que la respuesta varía según el modelo, se puede utilizar otra función, invoke_model_get_embeddings, que funciona exactamente igual que invoke_model, pero genera las incrustaciones especificando la json-key adecuada.

```
aws_bedrock.invoke_model_get_embeddings(
   IN model_id      varchar,
   IN content_type  text,
   IN json_key      text,
   IN model_input   text,
   OUT model_output float8[])
```

Las entradas y salidas de esta función son las siguientes.

- `model_id`: identificador del modelo.

- `content_type`: el tipo de solicitud al modelo de Bedrock. Aquí, el accept_type se establece en el valor predeterminado `application/json`.
- `model_input`: peticiones; un conjunto específico de entradas al modelo en el formato especificado por content_type. Para obtener más información sobre el formato o la estructura de las solicitudes que acepta el modelo, consulte Parámetros de inferencia para modelos fundacionales.
- `json_key`: referencia al campo del que se va a extraer la incrustación. Esto puede variar si cambia el modelo de incrustación.
- `model_output`: la salida del modelo de Bedrock como una matriz de incrustaciones con decimales de 16 bits.

El siguiente ejemplo muestra cómo generar una incrustación utilizando el modelo Titan Embeddings G1: modelo de incrustación de texto para la frase “vistas de monitorización de PostgreSQL I/O”.

Example Ejemplo: Una consulta sencilla con las funciones de Amazon Bedrock

```
SELECT aws_bedrock.invoke_model_get_embeddings(
   model_id      := 'amazon.titan-embed-text-v1',
   content_type  := 'application/json',
   json_key      := 'embedding',
   model_input   := '{ "inputText": "PostgreSQL I/O monitoring views"}') AS embedding;
```

Uso de Amazon Comprehend con el clúster de base de datos de Aurora PostgreSQL

Para Aurora PostgreSQL, el machine learning de Aurora proporciona la siguiente función de Amazon Comprehend para trabajar con los datos de texto. Esta función solo está disponible después de instalar la extensión `aws_ml` y completar todos los procedimientos de configuración. Para obtener más información, consulte Configuración del clúster de base de datos Aurora PostgreSQL para utilizar el machine learning de Aurora.

aws_comprehend.detect_sentiment

Esta función toma el texto como entrada y evalúa si el texto tiene una postura emocional positiva, negativa, neutra o mixta. Genera este sentimiento junto con un nivel de confianza para su evaluación. A continuación se presenta un resumen de la documentación de la función.

```
aws_comprehend.detect_sentiment(
```

```
    IN input_text varchar,
    IN language_code varchar,
    IN max_rows_per_batch int,
    OUT sentiment varchar,
    OUT confidence real)
```

Las entradas y salidas de esta función son las siguientes.

- `input_text`: el texto para evaluar y asignar el sentimiento (negativo, positivo, neutro, mixto).
- `language_code`: el idioma del `input_text` identificado mediante el identificador ISO 639-1 de 2 letras con subetiqueta regional (según sea necesario) o el código de tres letras ISO 639-2, según proceda. Por ejemplo, en es el código del inglés, zh es el código del chino simplificado. Para obtener más información, consulte los idiomas admitidos en la Guía para desarrolladores de Amazon Comprehend.
- `max_rows_per_batch`: el número máximo de filas por lote para el procesamiento por lotes. Para obtener más información, consulte Descripción del modo por lotes y las funciones de machine learning de Aurora.
- `sentiment`: el sentimiento del texto de entrada, identificado como POSITIVE, NEGATIVE, NEUTRAL o MIXED.
- `confidence`: el grado de confianza en la precisión del `sentiment` especificado. Los valores están comprendidos entre 0,0 y 1,0.

A continuación, puede encontrar ejemplos de cómo utilizar esta función.

Example Ejemplo: una consulta sencilla con las funciones de Amazon Comprehend

A continuación, se muestra un ejemplo de una consulta sencilla que invoca esta función para evaluar la satisfacción del cliente con su equipo de soporte. Supongamos que tiene una tabla de base de datos (`support`) que almacena los comentarios de los clientes después de cada solicitud de ayuda. En este ejemplo de consulta se aplica la función `aws_comprehend.detect_sentiment` al texto de la columna `feedback` de la tabla y muestra el sentimiento y el nivel de confianza de ese sentimiento. Esta consulta también muestra los resultados en orden descendente.

```
SELECT feedback, s.sentiment,s.confidence
    FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
    ORDER BY s.confidence DESC;
 feedback                        | sentiment | confidence
```

```
-------------------------------------------------------------+-----------+------------
Thank you for the excellent customer support!                | POSITIVE  |   0.999771
The latest version of this product stinks!                   | NEGATIVE  |   0.999184
Your support team is just awesome! I am blown away.          | POSITIVE  |   0.997774
Your product is too complex, but your support is great.      | MIXED     |   0.957958
Your support tech helped me in fifteen minutes.              | POSITIVE  |   0.949491
My problem was never resolved!                               | NEGATIVE  |   0.920644
When will the new version of this product be released?       | NEUTRAL   |   0.902706
I cannot stand that chatbot.                                 | NEGATIVE  |   0.895219
Your support tech talked down to me.                         | NEGATIVE  |   0.868598
It took me way too long to get a real person.                | NEGATIVE  |   0.481805

(10 rows)
```

Para evitar que la detección de sentimientos se cobre más de una vez por cada fila de la tabla, puede materializar los resultados. Haga esto en las filas de interés. Por ejemplo, se están actualizando las notas del médico para que solo las que estén en francés (`fr`) usen la función de detección de sentimiento.

```
UPDATE clinician_notes
SET sentiment = (aws_comprehend.detect_sentiment (french_notes, 'fr')).sentiment,
    confidence = (aws_comprehend.detect_sentiment (french_notes, 'fr')).confidence
WHERE
    clinician_notes.french_notes IS NOT NULL AND
    LENGTH(TRIM(clinician_notes.french_notes)) > 0 AND
    clinician_notes.sentiment IS NULL;
```

Para obtener más información sobre cómo optimizar las llamadas a las funciones, consulte Consideraciones sobre el rendimiento para utilizar el machine learning de Aurora con Aurora PostgreSQL.

Uso de IA de SageMaker con el clúster de base de datos de Aurora PostgreSQL

Tras configurar el entorno de IA de SageMaker e integrarlo con Aurora PostgreSQL, tal como se describe en Configuración de Aurora PostgreSQL para usar IA de Amazon SageMaker , puede invocar operaciones mediante la función `aws_sagemaker.invoke_endpoint`. La función `aws_sagemaker.invoke_endpoint` solo se conecta a un punto de conexión de modelo en la misma Región de AWS. Si la instancia de la base de datos tiene réplicas en varias Regiones de

AWS, asegúrese de configurar e implementar cada modelo de IA de SageMaker en cada Región de AWS.

Las llamadas a `aws_sagemaker.invoke_endpoint` se autentican mediante el rol de IAM que configuró para asociar su clúster de base de datos de Aurora PostgreSQL al servicio de IA de SageMaker y al punto de conexión que proporcionó durante el proceso de configuración. El ámbito de los puntos de conexión del modelo de IA de SageMaker se aplica a una cuenta individual y no son públicos. La URL `endpoint_name` no contiene el ID de la cuenta. IA de SageMaker determina el ID de cuenta a partir del token de autenticación proporcionado por el rol de IAM de SageMaker de la instancia de base de datos.

aws_sagemaker.invoke_endpoint

Esta función toma el punto de conexión de IA de SageMaker como entrada y el número de filas que se deben procesar como un lote. También toma como entrada los distintos parámetros esperados por el punto de conexión del modelo de IA de SageMaker. La documentación de referencia de esta función es la siguiente.

```
aws_sagemaker.invoke_endpoint(
  IN endpoint_name varchar,
  IN max_rows_per_batch int,
  VARIADIC model_input "any",
  OUT model_output varchar
  )
```

Las entradas y salidas de esta función son las siguientes.

- `endpoint_name`: una URL de punto de conexión independiente de la Región de AWS.
- `max_rows_per_batch`: el número máximo de filas por lote para el procesamiento por lotes. Para obtener más información, consulte Descripción del modo por lotes y las funciones de machine learning de Aurora.
- `model_input`: uno o más parámetros de entrada para el modelo. Estos pueden ser cualquier tipo de datos que el modelo de IA de SageMaker necesite. PostgreSQL le permite especificar hasta 100 parámetros de entrada para una función. Los tipos de datos de matriz deben ser unidimensionales, pero pueden contener tantos elementos como se esperan en el modelo de IA de SageMaker. El número de entradas de un modelo de IA de SageMaker solo está limitado por el límite de tamaño de los mensajes de IA de SageMaker, que es de 6 MB.
- `model_output`: la salida del modelo de IA de SageMaker como texto.

Creación de una función definida por el usuario para invocar un modelo de IA de SageMaker

Cree una función definida por el usuario independiente para llamar a `aws_sagemaker.invoke_endpoint` para cada uno de sus modelos de IA de SageMaker. Cada función definida por el usuario almacenada representa el punto de conexión de IA de SageMaker que aloja el modelo. La función `aws_sagemaker.invoke_endpoint` se ejecuta dentro de la función definida por el usuario. Las funciones definidas por el usuario proporcionan muchas ventajas:

- Puede dar a su modelo de IA de SageMaker su propio nombre en lugar de solo llamar a `aws_sagemaker.invoke_endpoint` para todos sus modelos de IA de SageMaker.
- Puede especificar la dirección URL del punto de conexión del modelo en un solo lugar en el código de la aplicación SQL.
- Puede controlar los privilegios de `EXECUTE` de cada función de machine learning de Aurora de forma independiente.
- Puede declarar los tipos de entrada y salida del modelo mediante tipos de SQL. SQL aplica el número y el tipo de argumentos pasados al modelo de IA de SageMaker y realiza la conversión de tipos si es necesario. El uso de tipos de SQL también traducirá `SQL NULL` al valor predeterminado adecuado esperado por su modelo de IA de SageMaker.
- Puede reducir el tamaño máximo del lote si desea devolver las primeras filas un poco más rápido.

Para especificar una función definida por el usuario, utilice la instrucción de lenguaje de definición de datos SQL (DDL) `CREATE FUNCTION`. Al definir la función, puede especificar lo siguiente:

- Los parámetros de entrada para el modelo.
- El punto de conexión de IA de SageMaker específico que se va a invocar.
- El tipo de devolución.

La función definida por el usuario devuelve la inferencia calculada por el punto de conexión de IA de SageMaker después de ejecutar el modelo en los parámetros de entrada. En el siguiente ejemplo se crea una función definida por el usuario para un modelo de IA de SageMaker con dos parámetros de entrada.

```
CREATE FUNCTION classify_event (IN arg1 INT, IN arg2 DATE, OUT category INT)
AS $$
```

```
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', NULL,
        arg1, arg2                        -- model inputs are separate arguments
        )::INT                            -- cast the output to INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Tenga en cuenta lo siguiente:

- La entrada de la función `aws_sagemaker.invoke_endpoint` puede ser uno o más parámetros de cualquier tipo de datos.
- En este ejemplo se utiliza un tipo de salida INT. Si convierte la salida de un tipo `varchar` a otro tipo, entonces debe convertirse a un tipo escalar integrado de PostgreSQL como `INTEGER`, `REAL`, `FLOAT` o `NUMERIC`. Para obtener más información acerca de estos tipos, consulte Data Types en la documentación de PostgreSQL.
- Especifique `PARALLEL SAFE` para habilitar el procesamiento de consultas paralelas. Para obtener más información, consulte Mejora de los tiempos de respuesta con el procesamiento de consultas en paralelo.
- Especifique `COST 5000` para estimar el costo de ejecución de la función. Utilice un número positivo que indique el costo de ejecución estimado para la función, en unidades de `cpu_operator_cost`.

Transmisión de una matriz como entrada de un modelo de IA de SageMaker

La función `aws_sagemaker.invoke_endpoint` puede tener hasta 100 parámetros de entrada, que es el límite para las funciones de PostgreSQL. Si el modelo de IA de SageMaker precisa más de 100 parámetros del mismo tipo, transfiera los parámetros del modelo como una matriz.

En el siguiente ejemplo se define una función definida por el usuario que transfiere una matriz como entrada al modelo de regresión de IA de SageMaker. La salida se convierte en un valor `REAL`.

```
CREATE FUNCTION regression_model (params REAL[], OUT estimate REAL)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
      'sagemaker_model_endpoint_name',
      NULL,
      params
      )::REAL
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Especificación del tamaño del lote al invocar un modelo de IA de SageMaker

En el ejemplo siguiente, se crea una función definida por el usuario para un modelo de IA de SageMaker que establece el valor predeterminado del tamaño del lote en NULL. La función también le permite proporcionar un tamaño de lote diferente cuando lo invoque.

```
CREATE FUNCTION classify_event (
    IN event_type INT, IN event_day DATE, IN amount REAL, -- model inputs
    max_rows_per_batch INT DEFAULT NULL,  -- optional batch size limit
    OUT category INT)                     -- model output
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', max_rows_per_batch,
        event_type, event_day, COALESCE(amount, 0.0)
        )::INT              -- casts output to type INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Tenga en cuenta lo siguiente:

- Utilice el parámetro `max_rows_per_batch` opcional para proporcionar el control del número de filas para una invocación de la función en modo de lote. Si utiliza un valor NULL, el optimizador de consultas elige automáticamente el tamaño máximo del lote. Para obtener más información, consulte Descripción del modo por lotes y las funciones de machine learning de Aurora.
- De forma predeterminada, transferir NULL como un valor de parámetro se traduce en una cadena vacía antes de transferirlo a IA de SageMaker. En este ejemplo, las entradas tienen diferentes tipos.
- Si tiene una entrada que no sea de texto o una entrada de texto que necesita de forma predeterminada un valor distinto de una cadena vacía, utilice la instrucción `COALESCE`. Se utiliza `COALESCE` para traducir NULL al valor de reemplazo nulo deseado en la llamada a `aws_sagemaker.invoke_endpoint`. Para el parámetro `amount` de este ejemplo, un valor NULL se convierte en 0.0.

Invocación de un modelo de IA de SageMaker que tenga varias salidas

En el ejemplo siguiente, se crea una función definida por el usuario para un modelo de IA de SageMaker que devuelva varias salidas. Su función necesita convertir la salida de la función `aws_sagemaker.invoke_endpoint` a un tipo de datos correspondiente. Por ejemplo, podría

usar el tipo de punto de PostgreSQL integrado para pares (x, y) o un tipo compuesto definido por el usuario.

Esta función definida por el usuario devuelve valores de un modelo que devuelve varias salidas mediante el uso de un tipo compuesto para las salidas.

```
CREATE TYPE company_forecasts AS (
    six_month_estimated_return real,
    one_year_bankruptcy_probability float);
CREATE FUNCTION analyze_company (
    IN free_cash_flow NUMERIC(18, 6),
    IN debt NUMERIC(18,6),
    IN max_rows_per_batch INT DEFAULT NULL,
    OUT prediction company_forecasts)
AS $$
    SELECT (aws_sagemaker.invoke_endpoint('endpt_name',
       max_rows_per_batch,free_cash_flow, debt))::company_forecasts;


$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Para el tipo compuesto, utilice los campos en el mismo orden que aparecen en la salida del modelo y convierta la salida de `aws_sagemaker.invoke_endpoint` al tipo compuesto. El intermediario puede extraer los campos individuales ya sea por nombre o con la notación ".*" de PostgreSQL.

Exportación de datos a Amazon S3 para el entrenamiento de modelos de IA de SageMaker (avanzado)

Le recomendamos que se familiarice con el machine learning de Aurora e IA de SageMaker utilizando los algoritmos y ejemplos proporcionados, en lugar de intentar entrenar sus propios modelos. Para obtener más información, consulte Get Started with Amazon SageMaker AI.

Para entrenar modelos de IA de SageMaker, exporte los datos a un bucket de Amazon S3. IA de SageMaker utiliza el bucket de Amazon S3 para entrenar el modelo antes de implementarlo. Puede consultar datos de un clúster de base de datos de Aurora PostgreSQL y guardarlos directamente en archivos almacenados en un bucket de Amazon S3. A continuación, IA de SageMaker consume los datos del bucket de Amazon S3 para el entrenamiento. Para obtener más información sobre el entrenamiento de modelos de IA de SageMaker, consulte Train a model with Amazon SageMaker AI.

> **Note**
>
> Al crear un bucket de Amazon S3 para el entrenamiento del modelo de IA de SageMaker o la puntuación por lotes, utilice `sagemaker` en el nombre del bucket de Amazon S3. Para obtener más información, consulte Especificar un bucket de S3 para cargar conjuntos de datos de entrenamiento y almacenar datos de salida en la Guía para desarrolladores de IA de Amazon SageMaker.

Para obtener más información acerca de la exportación de datos, consulte Exportación de datos de una Aurora PostgreSQL de base de datos de clústerde Amazon S3.

Consideraciones sobre el rendimiento para utilizar el machine learning de Aurora con Aurora PostgreSQL

Los servicios de Amazon Comprehend e IA de SageMaker realizan la mayor parte del trabajo cuando se invocan mediante una función de machine learning de Aurora. Esto significa que puede escalar esos recursos según sea necesario, de forma independiente. Para su clúster de base de datos Aurora PostgreSQL, puede hacer que sus llamadas a funciones sean lo más eficientes posible. A continuación, encontrará algunas consideraciones de rendimiento que debe tener en cuenta al trabajar con el machine learning de Aurora PostgreSQL.

Temas

- Descripción del modo por lotes y las funciones de machine learning de Aurora
- Mejora de los tiempos de respuesta con el procesamiento de consultas en paralelo
- Uso de vistas materializadas y columnas materializadas

Descripción del modo por lotes y las funciones de machine learning de Aurora

Normalmente, PostgreSQL ejecuta funciones una fila a la vez. El machine learning de Aurora puede reducir esta sobrecarga combinando las llamadas al servicio externo de machine learning de Aurora para muchas filas en lotes con un enfoque llamado ejecución en modo por lotes. En el modo por lotes, el machine learning de Aurora recibe las respuestas de un lote de filas de entrada y, a continuación, devuelve las respuestas a la consulta en ejecución una fila a la vez. Esta optimización mejora el rendimiento de las consultas de Aurora sin limitar el optimizador de consultas de PostgreSQL.

Aurora utiliza automáticamente el modo por lotes si se hace referencia a la función desde la lista SELECT, una cláusula WHERE o una cláusula HAVING. Tenga en cuenta que las expresiones CASE simples de nivel superior son elegibles para la ejecución en el modo por lotes. Las expresiones CASE buscadas de nivel superior también son elegibles para la ejecución en el modo por lotes siempre que la primera cláusula WHEN sea un predicado simple con una llamada a la función en el modo por lotes.

Su función definida por el usuario debe ser una función LANGUAGE SQL y debe especificar PARALLEL SAFE y COST 5000.

Migración de funciones de la instrucción SELECT a la cláusula FROM

Normalmente, Aurora migra automáticamente una función aws_ml que es elegible para la ejecución en el modo por lotes a la cláusula FROM.

La migración de funciones elegibles para el modo por lotes a la cláusula FROM se puede examinar manualmente en el nivel de consulta. Para ello, utilice instrucciones EXPLAIN (y ANALYZE y VERBOSE) y busque la información "Procesamiento por lotes" debajo de cada modo por lotes Function Scan. También puede usar EXPLAIN (con VERBOSE) sin ejecutar la consulta. A continuación, observe si las llamadas a la función aparecen como un Function Scan bajo una unión de bucle anidado que no se especificó en la instrucción original.

En el siguiente ejemplo, la el operador de unión de bucle anidado en el plan muestra que Aurora migró la función anomaly_score. Esta función se migró de la lista SELECT a la cláusula FROM, donde es elegible para la ejecución en el modo por lotes.

```
EXPLAIN (VERBOSE, COSTS false)
SELECT anomaly_score(ts.R.description) from ts.R;
                          QUERY PLAN
-------------------------------------------------------------
 Nested Loop
   Output: anomaly_score((r.description)::text)
   ->  Seq Scan on ts.r
         Output: r.id, r.description, r.score
   ->  Function Scan on public.anomaly_score
         Output: anomaly_score.anomaly_score
         Function Call: anomaly_score((r.description)::text)
```

Para deshabilitar la ejecución en el modo por lotes, establezca el parámetro apg_enable_function_migration en false. Esto impide la migración de funciones de aws_ml de SELECT a la cláusula FROM. El siguiente ejemplo muestra cómo.

```
SET apg_enable_function_migration = false;
```

El parámetro `apg_enable_function_migration` es un parámetro Grand Unified Configuration (GUC) reconocido por la extensión `apg_plan_mgmt` de Aurora PostgreSQL para la administración de planes de consulta. Para deshabilitar la migración de funciones en una sesión, utilice la administración de planes de consulta para guardar el plan resultante como un plan `approved`. En tiempo de ejecución, la administración de planes de consulta aplica el plan `approved` con su configuración `apg_enable_function_migration`. Esta aplicación se produce independientemente de la configuración del parámetro GUC `apg_enable_function_migration`. Para obtener más información, consulte Administración de planes de ejecución de consultas para Aurora PostgreSQL.

Uso del parámetro max_rows_per_batch

Tanto la función `aws_comprehend.detect_sentiment` como `aws_sagemaker.invoke_endpoint` tienen un parámetro `max_rows_per_batch`. Este parámetro especifica el número de filas que se pueden enviar al servicio de machine learning de Aurora. Cuanto mayor sea el conjunto de datos procesado por su función, mayor será el tamaño del lote.

Las funciones del modo por lotes mejoran la eficiencia gracias a la creación de lotes de filas que distribuyen el costo de las llamadas a las funciones del machine learning de Aurora en un gran número de filas. Sin embargo, si una instrucción `SELECT` termina de forma prematura debido a una cláusula `LIMIT`, entonces el lote se puede construir en más filas de las que utiliza la consulta. Esta estrategia puede dar lugar a cargos adicionales en su cuenta de AWS. Para obtener los beneficios de la ejecución en el modo por lotes pero evitar la creación de lotes demasiado grandes, utilice un valor más pequeño para el parámetro `max_rows_per_batch` en las llamadas a funciones.

Si realiza un `EXPLAIN` (`VERBOSE`, `ANALYZE`) de una consulta que utiliza la ejecución en el modo por lotes, verá un operador `FunctionScan` que está por debajo de una unión de bucle anidado. El número de bucles comunicados por `EXPLAIN` es igual al número de veces que se ha obtenido una fila del operador `FunctionScan`. Si una instrucción utiliza una cláusula LIMIT, el número de recuperaciones es coherente. Para optimizar el tamaño del lote, establezca el parámetro `max_rows_per_batch` en este valor. Sin embargo, si se hace referencia a la función del modo por lotes en un predicado en la cláusula `WHERE` o la cláusula `HAVING`, entonces probablemente no pueda saber el número de recuperaciones por adelantado. En este caso, utilice los bucles como guía y experimente con `max_rows_per_batch` para encontrar un ajuste que optimice el rendimiento.

Verificación de la ejecución en el modo por lotes

Para ver si una función se ejecuta en modo por lotes, utilice `EXPLAIN ANALYZE`. Si se utilizó la ejecución en el modo por lotes, el plan de consulta incluirá la información en una sección "Procesamiento por lotes".

```
EXPLAIN ANALYZE SELECT user-defined-function();
 Batch Processing: num batches=1 avg/min/max batch size=3333.000/3333.000/3333.000
                                 avg/min/max batch call time=146.273/146.273/146.273
```

En este ejemplo, había 1 lote que contenía 3333 filas, que tardó 146 273 ms en procesarse. La sección "Procesamiento por lotes" muestra lo siguiente:

- Cuántos lotes había para esta operación de escaneo de funciones
- El tamaño promedio, mínimo y máximo del lote
- El tiempo de ejecución por lotes promedio, mínimo y máximo

Normalmente, el lote final es más pequeño que el resto, lo que a menudo resulta en un tamaño mínimo de lote mucho más pequeño que el promedio.

Para devolver las primeras filas más rápidamente, establezca el parámetro `max_rows_per_batch` en un valor más pequeño.

Para reducir el número de llamadas del modo por lotes al servicio de ML cuando se utiliza un `LIMIT` en la función definida por el usuario, establezca el parámetro `max_rows_per_batch` en un valor menor.

Mejora de los tiempos de respuesta con el procesamiento de consultas en paralelo

Para obtener resultados lo más rápido posible de un gran número de filas, puede combinar el procesamiento de consultas en paralelo con el procesamiento en modo por lotes. Puede utilizar el procesamiento de consultas en paralelo para instrucciones `SELECT`, `CREATE TABLE AS SELECT` y `CREATE MATERIALIZED VIEW`.

> **Note**
>
> PostgreSQL aún no admite consultas en paralelo para instrucciones de lenguaje de manipulación de datos (DML).

El procesamiento de consultas en paralelo se produce tanto dentro de la base de datos como dentro del servicio de ML. El número de núcleos de la clase de instancia de la base de datos limita el grado de paralelismo que se puede utilizar cuando se ejecuta una consulta. El servidor de la base de datos puede construir un plan de ejecución de consultas en paralelo que divide la tarea entre un conjunto de subprocesos de trabajo paralelos. A continuación, cada uno de estos subprocesos de trabajo puede crear solicitudes por lotes que contengan decenas de miles de filas (o tantas como permita cada servicio).

Las solicitudes por lotes de todos los trabajadores paralelos se envían al punto de conexión de IA de SageMaker. El grado de paralelismo que puede admitir el punto de conexión está limitado por el número y el tipo de instancias que lo admiten. Para obtener un paralelismo de K grados, se necesita una clase de instancia de base de datos que tenga al menos K núcleos. También debe configurar el punto de conexión de IA de SageMaker para que su modelo tenga K instancias iniciales de una clase de instancia de rendimiento suficientemente alto.

Para usar el procesamiento de consultas en paralelo, puede establecer el parámetro de almacenamiento `parallel_workers` de la tabla que contiene los datos que planea transferir. Se establece `parallel_workers` en una función de modo por lotes como `aws_comprehend.detect_sentiment`. Si el optimizador elige un plan de consultas en paralelo, se puede llamar a los servicios de ML de AWS tanto en lote como en paralelo.

Puede utilizar los siguientes parámetros con la función `aws_comprehend.detect_sentiment` para obtener un plan con paralelismo de cuatro vías. Si cambia uno de los dos parámetros siguientes, debe reiniciar la instancia de la base de datos para que los cambios surtan efecto

```
-- SET max_worker_processes to 8;  -- default value is 8
-- SET max_parallel_workers to 8;  -- not greater than max_worker_processes
SET max_parallel_workers_per_gather to 4;  -- not greater than max_parallel_workers

-- You can set the parallel_workers storage parameter on the table that the data
-- for the Aurora machine learning function is coming from in order to manually
 override the degree of
-- parallelism that would otherwise be chosen by the query optimizer
--
ALTER TABLE yourTable SET (parallel_workers = 4);

-- Example query to exploit both batch-mode execution and parallel query
EXPLAIN (verbose, analyze, buffers, hashes)
SELECT aws_comprehend.detect_sentiment(description, 'en')).*
FROM yourTable
```

```
WHERE id < 100;
```

Para obtener más información sobre el control de consultas en paralelo, consulte Planes paralelos en la documentación de PostgreSQL.

Uso de vistas materializadas y columnas materializadas

Cuando invoca un servicio de AWS como IA de SageMaker o Amazon Comprehend desde su base de datos, se cobrará a su cuenta de acuerdo con la política de precios de dicho servicio. Para minimizar los cargos a su cuenta, puede materializar el resultado de las llamadas al servicio de AWS en una columna materializada para que no se llame al servicio de AWS más de una vez por fila de entrada. Si lo desea, puede agregar una columna de marca de tiempo `materializedAt` para registrar la hora a la que se materializaron las columnas.

La latencia de una instrucción `INSERT` de una sola fila ordinaria suele ser mucho menor que la latencia de llamar a una función de modo por lotes. Por lo tanto, es posible que no pueda cumplir los requisitos de latencia de su aplicación si invoca la función de modo por lotes para cada fila única `INSERT` que realiza la aplicación. Para materializar el resultado de llamar a un servicio de AWS en una columna materializada, las aplicaciones de alto rendimiento generalmente tienen que rellenar las columnas materializadas. Para ello, emiten periódicamente una instrucción `UPDATE` que opera en un gran lote de filas al mismo tiempo.

`UPDATE` toma un bloqueo de nivel de fila que puede afectar a una aplicación en ejecución. Por lo tanto, es posible que tenga que utilizar `SELECT ... FOR UPDATE SKIP LOCKED` o `MATERIALIZED VIEW`.

Las consultas analíticas que operan en un gran número de filas en tiempo real pueden combinar la materialización en modo por lotes con el procesamiento en tiempo real. Para ello, estas consultas ensamblan un `UNION ALL` de los resultados prematerializados con una consulta sobre las filas que aún no tienen resultados materializados. En algunos casos, `UNION ALL` es necesario en varios lugares, o una aplicación de terceros genera la consulta. Si es así, puede crear un `VIEW` para encapsular la operación `UNION ALL` para que este detalle no esté expuesto al resto de la aplicación SQL.

Puede utilizar una vista materializada para materializar los resultados de una instrucción `SELECT` arbitraria en una instantánea en el tiempo. También puede utilizarse para actualizar la vista materializada en cualquier momento en el futuro. Actualmente, PostgreSQL no admite la actualización incremental, por lo que cada vez que se actualiza la vista materializada, la vista materializada se vuelve a calcular por completo.

Puede actualizar vistas materializadas con la opción `CONCURRENTLY`, que actualiza el contenido de la vista materializada sin tener un bloqueo exclusivo. Al hacer esto, una aplicación SQL puede leer desde la vista materializada mientras se actualiza.

Monitorización del machine learning de Aurora

Puede supervisar las funciones `aws_ml` estableciendo el parámetro `track_functions` de su grupo de parámetros de clúster de base de datos personalizado en `all`. De forma predeterminada, este parámetro está configurado en `pl`, lo que significa que solo se rastrean las funciones del lenguaje de procedimientos. Al cambiarlo a `all`, también se hace un seguimiento de las funciones `aws_ml`. Para obtener más información, consulte Run-time Statistics en la documentación de PostgreSQL.

Para obtener información acerca de la supervisión del rendimiento de las operaciones de IA de SageMaker llamadas desde las funciones del machine learning de Aurora, consulte Monitor Amazon SageMaker AI en la Guía para desarrolladores de IA de Amazon SageMaker.

Si `track_functions` está configurado en `all`, puede consultar la vista `pg_stat_user_functions` para obtener estadísticas sobre las funciones que define y utiliza para invocar los servicios de machine learning de Aurora. Para cada función, la vista incluye el número de `calls`, `total_time` y `self_time`.

Para ver las estadísticas de `aws_sagemaker.invoke_endpoint` y las funciones `aws_comprehend.detect_sentiment`, puede filtrar los resultados por nombre de esquema mediante la siguiente consulta.

```
SELECT * FROM pg_stat_user_functions
    WHERE schemaname
    LIKE 'aws_%';
```

Para borrar las estadísticas, haga lo siguiente.

```
SELECT pg_stat_reset();
```

Puede obtener los nombres de las funciones de SQL que llaman a la función `aws_sagemaker.invoke_endpoint` consultando el catálogo del sistema `pg_proc` de PostgreSQL. Este catálogo almacena información sobre funciones, procedimientos y mucho más. Para obtener más información consulte pg_proc, en la documentación de PostgreSQL. El siguiente es un ejemplo de consulta en la tabla para obtener los nombres de las funciones (`proname`) cuyo origen (`prosrc`) incluye el texto invoke_endpoint.

```
SELECT proname FROM pg_proc WHERE prosrc LIKE '%invoke_endpoint%';
```

Ejemplos de código de Aurora con SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar Aurora con un kit de desarrollo de software (SDK) de AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción

Introducción a Aurora

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Aurora.

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
```

```
namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Código del archivo de CMake CMakeLists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código del archivo de origen hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
```

```
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
```

```
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
```

```
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)
```

```
# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
```

```
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Para obtener información sobre la API, consulte DescribeDBClusters en Referencia de la API de AWS SDK para Rust.

Ejemplos de código

- Ejemplos básicos para Aurora con AWS SDK
  - Introducción a Aurora
  - Explicación de los conceptos básicos de Aurora con un AWS SDK
  - Acciones de Aurora con SDK de AWS
    - Cómo utilizar CreateDBCluster con un AWS SDK
    - Cómo utilizar CreateDBClusterParameterGroup con un AWS SDK
    - Cómo utilizar CreateDBClusterSnapshot con un AWS SDK
    - Cómo utilizar CreateDBInstance con un AWS SDK
    - Cómo utilizar DeleteDBCluster con un AWS SDK

- Cómo utilizar DeleteDBClusterParameterGroup con un AWS SDK
- Cómo utilizar DeleteDBInstance con un AWS SDK
- Cómo utilizar DescribeDBClusterParameterGroups con un AWS SDK
- Cómo utilizar DescribeDBClusterParameters con un AWS SDK
- Cómo utilizar DescribeDBClusterSnapshots con un AWS SDK
- Cómo utilizar DescribeDBClusters con un AWS SDK
- Cómo utilizar DescribeDBEngineVersions con un AWS SDK
- Cómo utilizar DescribeDBInstances con un AWS SDK
- Uso de DescribeOrderableDBInstanceOptions con un SDK de AWS o la CLI
- Cómo utilizar ModifyDBClusterParameterGroup con un AWS SDK

- Escenarios en Aurora en los que se utilizan SDK de AWS
  - Creación de una API de REST de biblioteca de préstamos
  - Crear un rastreador de elementos de trabajo de Aurora Serverless

Ejemplos básicos para Aurora con AWS SDK

En los siguientes ejemplos de código, se muestra cómo utilizar los principios básicos de Amazon Aurora con AWS SDK.

Ejemplos

- Introducción a Aurora
- Explicación de los conceptos básicos de Aurora con un AWS SDK
- Acciones de Aurora con SDK de AWS
  - Cómo utilizar CreateDBCluster con un AWS SDK
  - Cómo utilizar CreateDBClusterParameterGroup con un AWS SDK
  - Cómo utilizar CreateDBClusterSnapshot con un AWS SDK
  - Cómo utilizar CreateDBInstance con un AWS SDK
  - Cómo utilizar DeleteDBCluster con un AWS SDK
  - Cómo utilizar DeleteDBClusterParameterGroup con un AWS SDK
  - Cómo utilizar DeleteDBInstance con un AWS SDK
  - Cómo utilizar DescribeDBClusterParameterGroups con un AWS SDK

- Cómo utilizar DescribeDBClusterParameters con un AWS SDK
- Cómo utilizar DescribeDBClusterSnapshots con un AWS SDK
- Cómo utilizar DescribeDBClusters con un AWS SDK
- Cómo utilizar DescribeDBEngineVersions con un AWS SDK
- Cómo utilizar DescribeDBInstances con un AWS SDK
- Uso de DescribeOrderableDBInstanceOptions con un SDK de AWS o la CLI
- Cómo utilizar ModifyDBClusterParameterGroup con un AWS SDK

Introducción a Aurora

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Aurora.

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Código del archivo de CMake CMakeLists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código del archivo de origen hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
```

```
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
```

```
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
```

```
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API del SDK de AWS para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)
```

```
# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
```

```
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Para obtener información acerca de la API, consulte DescribeDBClusters en Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Explicación de los conceptos básicos de Aurora con un AWS SDK

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de parámetros de clúster de base de datos de Aurora y defina los valores de los parámetros.
- Cree un clúster de base de datos que utilice el grupo de parámetros.
- Cree una instancia de base de datos que contenga una base de datos.

- Realice una instantánea del clúster de base de datos y luego limpie los recursos.

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario interactivo en un símbolo del sistema.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
```

```
   4.  Get some parameters in the group using the
DescribeDBClusterParametersAsync method.
   5.  Parse and display some parameters in the group.
   6.  Modify the auto_increment_offset and auto_increment_increment parameters
       using the ModifyDBClusterParameterGroupAsync method.
   7.  Get and display the updated parameters using the
DescribeDBClusterParametersAsync method with a source of "user".
   8.  Get a list of allowed engine versions using the
DescribeDBEngineVersionsAsync method.
   9.  Create an Aurora DB cluster that contains a MySql database and uses the
parameter group.
       using the CreateDBClusterAsync method.
   10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
method.
   11. Display and select from a list of instance classes available for the
selected engine and version
       using the paginated DescribeOrderableDBInstanceOptions method.
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances
method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
```

```
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<AuroraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<AuroraScenario>();

        auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);
```

```
            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
```

```
   /// Choose the Aurora DB parameter group family from a list of available
options.
   /// </summary>
   /// <returns>The selected parameter group family.</returns>
   public static async Task<string> ChooseParameterGroupFamilyAsync()
   {
       Console.WriteLine(sepBar);
       // 1. Get a list of available engines.
       var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

       Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

       var parameterGroupFamilies =
           engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
       for (var i = 1; i <= parameterGroupFamilies.Count; i++)
       {
           var parameterGroupFamily = parameterGroupFamilies[i - 1];
           // List the available parameter group families.
           Console.WriteLine(
               $"\t{i}. Family: {parameterGroupFamily.Key}");
       }

       var choiceNumber = 0;
       while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
       {
           Console.WriteLine("2. Select an available DB parameter group family
by entering a number from the preceding list:");
           var choice = Console.ReadLine();
           Int32.TryParse(choice, out choiceNumber);
       }
       var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
       Console.WriteLine(sepBar);
       return parameterGroupFamilyChoice.Key;
   }

   /// <summary>
   /// Create and get information on a DB parameter group.
   /// </summary>
   /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
   /// <returns>The new DBParameterGroup.</returns>
```

```
    public static async Task<DBClusterParameterGroup>
 CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
 {dbParameterGroupFamily}:");

        var parameterGroup = await
 auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
 auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet

        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);
```

```
        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
```

```
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
group.");

        var parameters =
            await
auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
"user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }
```

```
        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
```

```
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
```

```
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
```

```
   public static async Task<DBInstance?> CreateNewInstance(
       string clusterIdentifier,
       string engineName,
       string engineVersion,
       string instanceClass,
       string instanceIdentifier)
   {
       Console.WriteLine(sepBar);
       Console.WriteLine($"12. Create a new DB instance with identifier
{instanceIdentifier}.");
       bool isInstanceReady = false;
       DBInstance newInstance;
       var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
       isInstanceReady = instances.FirstOrDefault(i =>
           i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

       if (isInstanceReady)
       {
           Console.WriteLine("Instance already created.");
           newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
       }
       else
       {
           newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
               clusterIdentifier,
               instanceIdentifier,
               engineName,
               engineVersion,
               instanceClass
           );

           Console.WriteLine("13. Waiting for DB instance to be ready...");
           while (!isInstanceReady)
           {
               Console.Write(".");
               Thread.Sleep(5000);
               instances = await
auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
               isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
               newInstance = instances.First();
           }
```

```
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
 {cluster.DBClusterIdentifier}.");
        var snapshot = await
 auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
```

```
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
```

```
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
 {newCluster.DBClusterIdentifier}.");
            await
 auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
 newCluster.DBClusterIdentifier);
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
```

```
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Métodos de contenedor que llama el escenario para administrar las acciones de Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;

/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
```

```
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
 DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
```

```
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
 _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
```

```
   /// <returns>The name of the group that was modified.</returns>
   public async Task<string> ModifyIntegerParametersInGroupAsync(string
groupName, List<Parameter> parameters, int newValue = 0)
   {
       foreach (var p in parameters)
       {
           if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
           {
               while (newValue == 0)
               {
                   Console.WriteLine(
                       $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                   var choice = Console.ReadLine();
                   int.TryParse(choice, out newValue);
               }

               p.ParameterValue = newValue.ToString();
           }
       }

       var request = new ModifyDBClusterParameterGroupRequest
       {
           Parameters = parameters,
           DBClusterParameterGroupName = groupName,
       };

       var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
       return result.DBClusterParameterGroupName;
   }


   /// <summary>
   /// Get a list of orderable DB instance options for a specific
   /// engine and engine version.
   /// </summary>
   /// <param name="engine">Name of the engine.</param>
   /// <param name="engineVersion">Version of the engine.</param>
   /// <returns>List of OrderableDBInstanceOptions.</returns>
   public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
```

```
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
```

```
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
```

```
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
```

```
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });
```

```
        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });
```

```
        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
 dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para .NET.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBclústerParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBclústerSnapshots
  - DescribeDBclústers

- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
```

```
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
 std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " <<
 DB_ENGINE
```

```
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
```

```
              << std::endl;

   Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
   // 4.  Get the parameters in the DB cluster parameter group.
   if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                               NO_SOURCE,
                               autoIncrementParameters,
                               client)) {
       cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
       return false;
   }

   Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

   for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
       if (autoIncParameter.GetIsModifiable() &&
           (autoIncParameter.GetDataType() == "integer")) {
           std::cout << "The " << autoIncParameter.GetParameterName()
                     << " is described as: " <<
                     autoIncParameter.GetDescription() << "." << std::endl;
           if (autoIncParameter.ParameterValueHasBeenSet()) {
               std::cout << "The current value is "
                         << autoIncParameter.GetParameterValue()
                         << "." << std::endl;
           }
           std::vector<int> splitValues = splitToInts(
                   autoIncParameter.GetAllowedValues(), '-');
           if (splitValues.size() == 2) {
               int newValue = askQuestionForIntRange(
                       Aws::String("Enter a new value between ") +
                       autoIncParameter.GetAllowedValues() + ": ",
                       splitValues[0], splitValues[1]);
               autoIncParameter.SetParameterValue(std::to_string(newValue));
               updateParameters.push_back(autoIncParameter);

           }
           else {
               std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                         << std::endl;
           }
       }
   }
```

```
   {
       // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
       Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
       request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
       request.SetParameters(updateParameters);

       Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
               client.ModifyDBClusterParameterGroup(request);

       if (outcome.IsSuccess()) {
           std::cout << "The DB cluster parameter group was successfully
modified."
                     << std::endl;
       }
       else {
           std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
       }
   }

   std::cout
           << "You can get a list of parameters you've set by specifying a
source of 'user'."
           << std::endl;

   Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
   // 6.  Display the modified parameters in the DB cluster parameter group.
   if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                               userParameters,
                               client)) {
       cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
       return false;
   }

   for (const auto &userParameter: userParameters) {
       std::cout << "  " << userParameter.GetParameterName() << ", " <<
                 userParameter.GetDescription() << ", parameter value - "
                 << userParameter.GetParameterValue() << std::endl;
   }
```

```
    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
    else {
        std::cout << "Let's create a DB cluster." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of engine versions for the parameter group family.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group family are:"
                  << std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "   " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
```

```
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB cluster is configured to use your custom cluster
parameter group '"
                  << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }
```

```
    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }
```

```
    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
```

```
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();
```

```
   if (askYesNoQuestion(
           "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
       Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                              Aws::String(Aws::Utils::UUID::RandomUUID()));
       {
           std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
           std::cout << "This typically takes a few minutes." << std::endl;

           // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
           Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
           request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
           request.SetDBClusterSnapshotIdentifier(snapshotID);

           Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                   client.CreateDBClusterSnapshot(request);

           if (outcome.IsSuccess()) {
               std::cout << "Snapshot creation has started."
                         << std::endl;
           }
           else {
               std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
               cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
               return false;
           }
       }

       std::cout << "Waiting for the snapshot to become available." <<
std::endl;

       Aws::RDS::Model::DBClusterSnapshot snapshot;
       counter = 0;
       do {
           std::this_thread::sleep_for(std::chrono::seconds(1));
           ++counter;
           if (counter > 600) {
               std::cerr << "Wait for snapshot to be available timed out ofter "
                         << counter
```

```
                        << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
```

```
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }
```

```
    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
```

```
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
```

```
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
```

```
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
```

```
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
```

```
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
 parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
```

```
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
```

```
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
        }
    }

    return result;
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para C++.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBclústerParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBclústerSnapshots
  - DescribeDBclústers
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
```

```
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
```

```
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
```

```
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
```

```
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
```

```
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
   log.Println("Cluster created and available.")
  }
 }
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
```

```
if err != nil {
 panic(err)
}
if dbInstance == nil {
 log.Println("Let's create a database instance in your DB cluster.")
 log.Println("First, choose a DB instance type:")
 instOpts, err := scenario.dbClusters.GetOrderableInstances(
  ctx, *cluster.Engine, *cluster.EngineVersion)
 if err != nil {
  panic(err)
 }
 var instChoices []string
 for _, opt := range instOpts {
  instChoices = append(instChoices, *opt.DBInstanceClass)
 }
 slices.Sort(instChoices)
 instChoices = slices.Compact(instChoices)
 instIndex := scenario.questioner.AskChoice(
  "Which DB instance class do you want to use?\n", instChoices)
 log.Println("Creating a database instance. This typically takes several
minutes.")
 dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
  ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
*cluster.Engine,
  instChoices[instIndex])
 if err != nil {
  panic(err)
 }
 for *dbInstance.DBInstanceStatus != "available" {
  scenario.helper.Pause(30)
  dbInstance, err = scenario.dbClusters.GetInstance(ctx,
*cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
 }
}
log.Println("Instance data:")
log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
log.Printf("\tEngine: %v\n", *dbInstance.Engine)
log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
log.Println(strings.Repeat("-", 88))
```

```
 return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
```

```
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
```

```
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Defina las funciones a las que llama el escenario para administrar las acciones de Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}


// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
```

```
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {
```

```
 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
```

```
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
```

```
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
```

```
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Go.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBclústerParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBclústerSnapshots
  - DescribeDBclústers
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
```

```
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
+
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\"\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
```

```
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
```

```
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
```

```
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
```

```
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
```

```
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();
```

```
            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
```

```
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
```

```
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
```

```
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance

- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBclústerParameterGroups
- DescribeDBClusterParameters
- DescribeDBclústerSnapshots
- DescribeDBclústers
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
```

```
This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }
```

```
    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)
```

```
    println("10. Get a list of instance classes available for the selected
 engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
```

```
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
```

```
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
```

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
```

```
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
```

```
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
```

```
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Kotlin.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBclústerParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBclústerSnapshots
  - DescribeDBclústers
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Ejecutar un escenario interactivo en un símbolo del sistema.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
 engine and
        create a DB cluster parameter group that is compatible with a selected
 engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
 parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
 {parameter_group_name}."
        )
        parameter_group =
 self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
```

```
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
```

```
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.

        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
```

```
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
```

```
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
```

```
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)

    def create_snapshot(self, cluster_name):
        """
        Shows how to create a DB cluster snapshot and wait until it's available.

        :param cluster_name: The name of a DB cluster to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB cluster (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.aurora_wrapper.create_cluster_snapshot(
                snapshot_id, cluster_name
            )
            while snapshot.get("Status") != "available":
                wait(30)
                snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
            pp(snapshot)
```

```
            print("-" * 88)

    def cleanup(self, db_inst, cluster, parameter_group):
        """
        Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
        Before the DB cluster parameter group can be deleted, all associated DB
instances and
        DB clusters must first be deleted.

        :param db_inst: The DB instance to delete.
        :param cluster: The DB cluster to delete.
        :param parameter_group: The DB cluster parameter group to delete.
        """
        cluster_name = cluster["DBClusterIdentifier"]
        parameter_group_name = parameter_group["DBClusterParameterGroupName"]
        if q.ask(
            "\nDo you want to delete the database instance, DB cluster, and
parameter "
            "group (y/n)? ",
            q.is_yesno,
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)
```

```
    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
 db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
 started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
 parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
 parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina las funciones a las que llama el escenario para administrar las acciones de Aurora.

```
class AuroraWrapper:
```

```
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

```
    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
```

```
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
```

```
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
```

```
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

```
    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return db_inst


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
```

```
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance

- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBclústerParameterGroups
- DescribeDBClusterParameters
- DescribeDBclústerSnapshots
- DescribeDBclústers
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Biblioteca que contiene las funciones específicas del escenario Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
```

```
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
```

```
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
```

```
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }
```

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };

       let version_count = describe_db_engine_versions
           .as_ref()
           .map(|o| o.db_engine_versions().len())
           .unwrap_or_default();
       info!(version_count, "got list of versions");

       // Create a map of engine families to their available versions.
       let mut versions = HashMap::<String, Vec<String>>::new();
       describe_db_engine_versions
           .unwrap()
           .db_engine_versions()
           .iter()
           .filter_map(
               |v| match (&v.db_parameter_group_family, &v.engine_version) {
                   (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                   _ => None,
               },
           )
           .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

       Ok(versions)
   }

   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
       let describe_orderable_db_instance_options_items = self
```

```
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
```

```
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
```

```
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
    // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
    // Parse the ParameterName, Description, and AllowedValues values and display
them.
    pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
        let parameters_output = self
            .rds
            .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
            .await;

        if let Err(err) = parameters_output {
            return Err(ScenarioError::new(
                format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
                &err,
            ));
        }

        let parameters = parameters_output
            .unwrap()
            .into_iter()
            .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
```

```
            .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    pub async fn update_auto_increment(
        &self,
        offset: u8,
        increment: u8,
    ) -> Result<(), ScenarioError> {
        let modify_db_cluster_parameter_group = self
            .rds
            .modify_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }
```

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
    pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
        let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
        let snapshot = self
            .rds
            .snapshot_cluster(id, format!("{id}_{name}").as_str())
            .await;
        match snapshot {
            Ok(output) => match output.db_cluster_snapshot {
                Some(snapshot) => Ok(snapshot),
                None => Err(ScenarioError::with("Missing Snapshot")),
            },
            Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
        }
    }

    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
```

```
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
```

```
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
```

```
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
```

```
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Hace pruebas para la biblioteca que utiliza bloqueos automáticos alrededor del encapsulador del cliente de RDS.

```
use crate::rds::MockRdsImpl;

use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });
```

```
    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

.db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
```

```
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
```

```
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
```

```
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
```

```
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
```

```
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
```

```
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);
```

```
    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
```

```
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
```

```
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
```

```
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
```

```
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

 .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
```

```
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
```

```
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}

#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Binario que permite ejecutar el escenario de principio a fin, utilizando la herramienta de consulta para que el usuario pueda tomar determinadas decisiones.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }

    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
```

```
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
```

```
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
```

```
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
```

```
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();
```

```
    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;
```

```
    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Encapsulador alrededor del servicio Amazon RDS que permite bloquear automáticamente las pruebas.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;
```

```
#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
```

```
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }
```

```
    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
```

```
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
```

```
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
```

```
    }
}
```

El Cargo.toml con las dependencias utilizadas en este escenario.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Rust.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster

- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBclústerParameterGroups
- DescribeDBClusterParameters
- DescribeDBclústerSnapshots
- DescribeDBclústers
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones de Aurora con SDK de AWS

Los siguientes ejemplos de código muestran cómo llevar a cabo acciones individuales de Aurora con SDK de AWS. En cada ejemplo se incluye un enlace a GitHub, con instrucciones de configuración y ejecución del código.

Estos fragmentos llaman a la API de Aurora y son fragmentos de código de programas más grandes que deben ejecutarse en contexto. Puede ver las acciones en contexto en Escenarios en Aurora en los que se utilizan SDK de AWS.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la Referencia de la API de Amazon Aurora.

Ejemplos

- Cómo utilizar CreateDBCluster con un AWS SDK
- Cómo utilizar CreateDBClusterParameterGroup con un AWS SDK
- Cómo utilizar CreateDBClusterSnapshot con un AWS SDK
- Cómo utilizar CreateDBInstance con un AWS SDK
- Cómo utilizar DeleteDBCluster con un AWS SDK
- Cómo utilizar DeleteDBClusterParameterGroup con un AWS SDK

- Cómo utilizar DeleteDBInstance con un AWS SDK
- Cómo utilizar DescribeDBClusterParameterGroups con un AWS SDK
- Cómo utilizar DescribeDBClusterParameters con un AWS SDK
- Cómo utilizar DescribeDBClusterSnapshots con un AWS SDK
- Cómo utilizar DescribeDBClusters con un AWS SDK
- Cómo utilizar DescribeDBEngineVersions con un AWS SDK
- Cómo utilizar DescribeDBInstances con un AWS SDK
- Uso de DescribeOrderableDBInstanceOptions con un SDK de AWS o la CLI
- Cómo utilizar ModifyDBClusterParameterGroup con un AWS SDK

Cómo utilizar **`CreateDBCluster`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `CreateDBCluster`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
/// <param name="dbEngine">The engine to use for the new cluster.</param>
/// <param name="dbEngineVersion">The version of the engine to use.</param>
```

```
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }
```

- Para obtener información sobre la API, consulte CreateDBCluster en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Para obtener información sobre la API, consulte CreateDBCluster en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
```

```
 }
}
```

- Para obtener información sobre la API, consulte CreateDBCluster en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
```

```
    }
```

- Para obtener información sobre la API, consulte CreateDBCluster en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Para obtener información sobre la API, consulte CreateDBCluster en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
```

```
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Para obtener información sobre la API, consulte CreateDBCluster en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
```

```
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
```

```
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
```

```
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
```

```
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
```

```
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
```

```
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
```

```
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });
```

```
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener información acerca de la API, consulte CreateDBCluster en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`CreateDBClusterParameterGroup`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `CreateDBClusterParameterGroup`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
```

```
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Para obtener información sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obtener información sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

- Para obtener información sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obtener información sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Para obtener información sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obtener información sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
```

```
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
 do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
```

```
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Para obtener detalles sobre la API, consulte CreateDBClusterParameterGroup en la Referencia de la API del SDK AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`CreateDBClusterSnapshot`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `CreateDBClusterSnapshot`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Para obtener información sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Para obtener información sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
```

```
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- Para obtener información sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obtener información sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Para obtener información sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
```

```
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para obtener información sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
```

```
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }

       info!(
           "Started a db cluster: {}",
           self.db_cluster_identifier
               .as_deref()
               .unwrap_or("Missing ARN")
       );
```

```
        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
```

```
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }
```

```
        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
```

```
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener detalles sobre la API, consulte CreateDBClusterSnapshot en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`CreateDBInstance`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `CreateDBInstance`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Para obtener información sobre la API, consulte CreateDBInstance en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
```

- Para obtener información sobre la API, consulte CreateDBInstance en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
```

```
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Para obtener información sobre la API, consulte CreateDBInstance en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obtener información sobre la API, consulte CreateDBInstance en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Para obtener información sobre la API, consulte CreateDBInstance en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
 that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
```

```
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obtener información sobre la API, consulte CreateDBInstance en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));
```

```
            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
```

```
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
```

```
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
```

```
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
```

```
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
```

```
 .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener información acerca de la API, consulte CreateDBInstance en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DeleteDBCluster`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DeleteDBCluster`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });
```

```
        return response.DBCluster;
    }
```

- Para obtener información sobre la API, consulte DeleteDBCluster en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
```

```
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Para obtener información sobre la API, consulte DeleteDBCluster en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
```

```
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obtener información sobre la API, consulte DeleteDBCluster en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte DeleteDBCluster en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Para obtener información sobre la API, consulte DeleteDBCluster en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
```

```
            raise
```

- Para obtener información sobre la API, consulte DeleteDBCluster en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
    let mut clean_up_errors: Vec<ScenarioError> = vec![];

    // Delete the instance. rds.DeleteDbInstance.
    let delete_db_instance = self
        .rds
        .delete_db_instance(
            self.db_instance_identifier
                .as_deref()
                .expect("instance identifier"),
        )
        .await;
    if let Err(err) = delete_db_instance {
        let identifier = self
            .db_instance_identifier
            .as_deref()
            .unwrap_or("Missing Instance Identifier");
        let message = format!("failed to delete db instance {identifier}");
        clean_up_errors.push(ScenarioError::new(message, &err));
    } else {
        // Wait for the instance to delete
        let waiter = Waiter::default();
        while waiter.sleep().await.is_ok() {
```

```
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
```

```
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
```

```
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
```

```
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
```

```
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
```

```
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
```

```
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
```

```
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener información acerca de la API, consulte DeleteDBCluster en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DeleteDBClusterParameterGroup`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DeleteDBClusterParameterGroup`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obtener información sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obtener información sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obtener información sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }

           DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                   .builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
           System.out.println(dbClusterGroupName + " was deleted.");

       } catch (RdsException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
```

```
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Para obtener información sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
```

```
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obtener información sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
```

```
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

   mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

   let mut scenario = AuroraScenario::new(mock_rds);
   scenario.db_cluster_identifier = Some(String::from("MockCluster"));
   scenario.db_instance_identifier = Some(String::from("MockInstance"));
   scenario.db_cluster_parameter_group = Some(
       DbClusterParameterGroup::builder()
           .db_cluster_parameter_group_name("MockParamGroup")
           .build(),
   );

   tokio::time::pause();
   let assertions = tokio::spawn(async move {
       let clean_up = scenario.clean_up().await;
       assert!(clean_up.is_err());
       let errs = clean_up.unwrap_err();
       assert_eq!(errs.len(), 2);
       assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
       assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
   });

   tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener detalles sobre la API, consulte DeleteDBClusterParameterGroup en la Referencia de la API del AWS SDK para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DeleteDBInstance`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DeleteDBInstance`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
```

```
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Para obtener información sobre la API, consulte DeleteDBInstance en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);
```

```
            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Para obtener información sobre la API, consulte DeleteDBInstance en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obtener información sobre la API, consulte DeleteDBInstance en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obtener información sobre la API, consulte DeleteDBInstance en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
```

```
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Para obtener información sobre la API, consulte DeleteDBInstance en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obtener información sobre la API, consulte DeleteDBInstance en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
```

```
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
```

```
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
```

```
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
```

```
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
```

```
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener información acerca de la API, consulte DeleteDBInstance en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DescribeDBClusterParameterGroups`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DescribeDBClusterParameterGroups`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
```

```
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameterGroups en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
```

```
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameterGroups en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameterGroups en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();
```

```
            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameterGroups en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
```

```
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameterGroups en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
```

```
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Para obtener información acerca de las API, consulte DescribeDBClusterParameterGroups en la Referencia de la API del SDK de AWS para Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DescribeDBClusterParameters`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DescribeDBClusterParameters`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameters en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBClusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String
 &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameters en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
```

```
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameters en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
```

```
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameters en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameters en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Para obtener información sobre la API, consulte DescribeDBClusterParameters en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
```

```
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Para obtener información acerca de la API, consulte DescribeDBClusterParameters en Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DescribeDBClusterSnapshots`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DescribeDBClusterSnapshots`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /// <summary>
   /// Return a list of DB snapshots for a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>List of DB snapshots.</returns>
   public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
   {
       var results = new List<DBClusterSnapshot>();

       DescribeDBClusterSnapshotsResponse response;
       DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
       {
           DBClusterIdentifier = dbClusterIdentifier
       };
       // Get the full list if there are multiple pages.
       do
```

```
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Para obtener información sobre la API, consulte DescribeDBClusterSnapshots en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
```

```
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Para obtener información sobre la API, consulte DescribeDBClusterSnapshots en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterSnapshots en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();
```

```
            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte DescribeDBClusterSnapshots en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
```

```
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Para obtener información sobre la API, consulte DescribeDBClusterSnapshots en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para obtener detalles sobre la API, consulte DescribeDBClusterSnapshots en la Referencia de la API del SDK de AWS para Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DescribeDBClusters`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DescribeDBClusters`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
```

```
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
```

```
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
```

```
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener detalles sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Para obtener información sobre la API, consulte DescribeDBClusters en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
```

```
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
```

```
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
```

```
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener información acerca de la API, consulte DescribeDBClusters en Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DescribeDBEngineVersions`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DescribeDBEngineVersions`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Para obtener información sobre la API, consulte DescribeDBEngineVersions en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Para obtener información sobre la API, consulte DescribeDBEngineVersions en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Para obtener información sobre la API, consulte DescribeDBEngineVersions en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obtener información sobre la API, consulte DescribeDBEngineVersions en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Para obtener información sobre la API, consulte DescribeDBEngineVersions en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
                                       engine versions to those that are
 compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
```

```
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Para obtener información sobre la API, consulte DescribeDBEngineVersions en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");
```

```
        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }
```

```
#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}
```

```
#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Para obtener información acerca de la API, consulte DescribeDBEngineVersions en Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`DescribeDBInstances`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `DescribeDBInstances`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }
```

- Para obtener información sobre la API, consulte DescribeDBInstances en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
```

```
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Para obtener información sobre la API, consulte DescribeDBInstances en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Para obtener información sobre la API, consulte DescribeDBInstances en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    // Waits until the database instance is available.
```

```
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
                       instanceReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }
           System.out.println("Database cluster is available!");

       } catch (RdsException | InterruptedException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obtener información sobre la API, consulte DescribeDBInstances en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Para obtener información sobre la API, consulte DescribeDBInstances en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Para obtener información sobre la API, consulte DescribeDBInstances en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
```

```
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
```

```
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
```

```
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
```

```
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
```

```
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Cluster
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
```

```
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obtener información acerca de la API, consulte DescribeDBInstances en Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **`DescribeOrderableDBInstanceOptions`** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeOrderableDBInstanceOptions`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Para obtener información sobre la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
```

```
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Para obtener información sobre la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
```

```
 })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obtener información sobre la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();
```

```
            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de la API de AWS SDK for Java 2.x.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este ejemplo enumera las versiones del motor de base de datos que admiten una clase de instancia de base de datos concreta en una Región de AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Ejemplo 2: este ejemplo enumera las clases de instancia de base de datos que se admiten para una versión de motor de base de datos concreta en una Región de AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
```

```
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para obtener detalles de la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de Herramientas de AWS para PowerShell Cmdlet (V4).

Herramientas para PowerShell V5

Ejemplo 1: este ejemplo enumera las versiones del motor de base de datos que admiten una clase de instancia de base de datos concreta en una Región de AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Ejemplo 2: este ejemplo enumera las clases de instancia de base de datos que se admiten para una versión de motor de base de datos concreta en una Región de AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para obtener detalles de la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de Herramientas de AWS para PowerShell Cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
 instance.
        :param db_engine_version: The engine version that must be supported by
 the DB instance.
        :return: The list of DB instance options that can be used to create a
 compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Para obtener información sobre la API, consulte DescribeOrderableDBInstanceOptions en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
```

```
                    .map(|o|
 o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
 classes", &err))
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Para obtener información acerca de la API, consulte DescribeOrderableDBInstanceOptions en la Referencia del SDK de AWS para la API de Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo utilizar **`ModifyDBClusterParameterGroup`** con un AWS SDK

Los siguientes ejemplos de código muestran cómo utilizar `ModifyDBClusterParameterGroup`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Conceptos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
```

```
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Para obtener información sobre la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
```

```
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obtener información sobre la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
```

```
    DBClusterParameterGroupName: aws.String(parameterGroupName),
    Parameters:                  params,
   })
  if err != nil {
   log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
   return err
  } else {
   return nil
  }
}
```

- Para obtener información sobre la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
```

```
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obtener información sobre la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API de AWS SDK for Java 2.x.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Para obtener información sobre la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
```

```
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obtener información sobre la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
```

```
    pub async fn update_auto_increment(
        &self,
        offset: u8,
        increment: u8,
    ) -> Result<(), ScenarioError> {
        let modify_db_cluster_parameter_group = self
            .rds
            .modify_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
```

```
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Para obtener información acerca de la API, consulte ModifyDBClusterParameterGroup en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Escenarios en Aurora en los que se utilizan SDK de AWS

En los siguientes ejemplos de código se muestra cómo implementar situaciones comunes en Aurora con SDK de AWS. Estos escenarios muestran cómo llevar a cabo tareas específicas con llamadas a varias funciones dentro de Aurora o en combinación con otros Servicios de AWS. En cada escenario se incluye un enlace al código fuente completo, con instrucciones de configuración y ejecución del código.

Los escenarios requieren un nivel intermedio de experiencia para ayudarlo a entender las acciones de servicio en su contexto.

Ejemplos

- Creación de una API de REST de biblioteca de préstamos
- Crear un rastreador de elementos de trabajo de Aurora Serverless

Creación de una API de REST de biblioteca de préstamos

En el siguiente ejemplo de código se muestra cómo crear una biblioteca de préstamos en la que los usuarios puedan pedir prestados y devolver libros mediante una API de REST respaldada por una base de datos de Amazon Aurora.

Python

SDK para Python (Boto3)

Muestra cómo utilizar AWS SDK para Python (Boto3) con la API de Amazon Relational Database Service (Amazon RDS) y AWS Chalice para crear una API de REST respaldada por una base de datos de Amazon Aurora. El servicio web es totalmente sin servidor y representa una biblioteca de préstamos sencilla en la que los usuarios pueden pedir prestados libros y devolverlos. Aprenda cómo:

- Crear y administrar un clúster de base de datos Aurora sin servidor.
- Usar AWS Secrets Manager para administrar las credenciales de la base de datos.
- Implementar una capa de almacenamiento de datos que utilice Amazon RDS para mover datos dentro y fuera de la base de datos.
- Usar AWS Chalice para implementar una API de REST sin servidor en Amazon API Gateway y AWS Lambda.
- Utilice el paquete Requests para enviar solicitudes al servicio web.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Crear un rastreador de elementos de trabajo de Aurora Serverless

Los siguientes ejemplos de código muestran cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora sin servidor y use Amazon Simple Email Service (Amazon SES) para enviar informes.

.NET

SDK para .NET

Muestra cómo utilizar AWS SDK para .NET para crear una aplicación web que haga un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora y envíe informes por correo electrónico mediante Amazon Simple Email Service (Amazon SES). Este ejemplo usa un frontend creado con React.js para interactuar con un backend .NET RESTful.

- Integre una aplicación web de React con los servicios de AWS.
- Muestre, agregue, actualice y elimine elementos en una tabla de Aurora.
- Envíe un informe por correo electrónico de elementos de trabajo filtrados con Amazon SES.
- Implemente y administre recursos de ejemplo con el script de AWS CloudFormation.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

C++

SDK para C++

Muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon Aurora sin servidor e informe al respecto.

Para obtener el código fuente completo e instrucciones sobre cómo configurar una API de REST de C++ que consulta datos de Amazon Aurora sin servidor y para su uso por parte de una aplicación React, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Java

SDK para Java 2.x

Muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon RDS e informe al respecto.

Para obtener el código fuente completo e instrucciones sobre cómo configurar una API de REST de Spring que consulta datos de Amazon Aurora Serverless y para su uso por parte de una aplicación React, consulte el ejemplo completo en GitHub.

Para obtener el código fuente completo e instrucciones sobre cómo configurar y ejecutar el ejemplo que utiliza la API de JDBC, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Muestra cómo utilizar AWS SDK para JavaScript (v3) para crear una aplicación web que realice un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora y envíe informes por correo electrónico mediante Amazon Simple Email Service (Amazon SES). Este ejemplo usa un frontend creado con React.js para interactuar con un backend de Node.js de Express.

- Integre una aplicación web de React.js con Servicios de AWS.
- Cree una lista, agregue y actualice elementos en una tabla de Aurora.
- Envíe un informe por correo electrónico de elementos de trabajo filtrados con Amazon SES.
- Implemente y administre recursos de ejemplo con el script de AWS CloudFormation incluido.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Kotlin

SDK para Kotlin

Muestra cómo crear una aplicación web que haga un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon RDS e informe al respecto.

Para obtener el código fuente completo e instrucciones sobre cómo configurar una API de REST de Spring que consulta datos de Amazon Aurora Serverless y para su uso por parte de una aplicación React, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora

- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

PHP

SDK para PHP

Muestra cómo utilizar AWS SDK para PHP para crear una aplicación web que haga un seguimiento de los elementos de trabajo de una base de datos de Amazon RDS y envíe informes por correo electrónico mediante Amazon Simple Email Service (Amazon SES). Este ejemplo usa un front-end creado con React.js para interactuar con un backend PHP RESTful.

- Integre una aplicación web de React.js con los servicios de AWS.
- Enumere, agregue, actualice y elimine elementos de una tabla de Amazon RDS.
- Envíe un informe por correo electrónico de elementos de trabajo filtrados con Amazon SES.
- Implemente y administre recursos de ejemplo con el script de AWS CloudFormation.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Python

SDK para Python (Boto3)

Muestra cómo utilizar AWS SDK para Python (Boto3) para crear un servicio REST que haga un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora sin servidor y envíe informes por correo electrónico mediante Amazon Simple Email Service (Amazon SES). En este ejemplo se utiliza el marco web de Flask para gestionar el enrutamiento HTTP y se integra con una página web de React para presentar una aplicación web completamente funcional.

- Cree un servicio REST de Flask que se integre con Servicios de AWS.
- Lea, escriba y actualice los elementos de trabajo almacenados en una base de datos de Aurora Serverless.
- Cree un secreto de AWS Secrets Manager que contenga las credenciales de la base de datos y utilícelo para autenticar las llamadas a la base de datos.
- Utilice Amazon SES para enviar informes de elementos de trabajo por correo electrónico.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte Cómo utilizar este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Prácticas recomendadas con Amazon Aurora

En este tema se explican prácticas recomendadas generales y opciones para usar los datos o migrarlos a un clúster de base de datos de Amazon Aurora.

Algunas de las prácticas recomendadas para Amazon Aurora son específicas de un motor de base de datos determinado. Para obtener más información sobre las prácticas recomendadas de Aurora específicas de los motores de base de datos, consulte lo siguiente:

Motor de base de datos	Prácticas recomendadas
MySQL de Amazon Aurora	Consulte Prácticas recomendadas con Amazon Aurora MySQL
PostgreSQL de Amazon Aurora	Consulte Prácticas recomendadas con Amazon Aurora PostgreSQL

> **Note**
>
> Para ver recomendaciones frecuentes para Aurora, consulte Recomendaciones para Amazon Aurora.

Temas

- Directrices operativas básicas de Amazon Aurora
- Recomendaciones de RAM de las instancias de base de datos
- Controladores de bases de datos de AWS
- Monitorización de Amazon Aurora
- Trabajo con los grupos de parámetros de base de datos y grupos de parámetros de clúster de base de datos
- Vídeo sobre las prácticas recomendadas de Amazon Aurora

Directrices operativas básicas de Amazon Aurora

A continuación se detallan las directrices operativas básicas que se deben seguir al trabajar con Amazon Aurora. El acuerdo de nivel de servicio de Amazon RDS requiere que se sigan estas directrices.

- Monitorice el uso de la memoria, la CPU y el almacenamiento. Puede configurar Amazon CloudWatch para notificar cuándo cambian los patrones de uso o cuándo se está llegando al límite de capacidad de la implementación. De esta forma, puede mantener el rendimiento y la disponibilidad del sistema.
- Si la aplicación cliente almacena en caché los datos del Servicio de nombres de dominio (DNS) de las instancias de base de datos, defina un valor de tiempo de vida (TTL) de menos de 30 segundos. La dirección IP subyacente de una instancia de base de datos puede cambiar después de producirse una conmutación por error. Por lo tanto, almacenar en caché los datos DNS durante un período prolongado puede provocar errores de conexión si la aplicación intenta conectarse a una dirección IP que ya no está en servicio. Los clústeres de base de datos de Aurora con múltiples réplicas de lectura pueden tener errores de conexión también cuando las conexiones usan el punto de enlace del lector y una de las instancias de réplica de lectura está en mantenimiento o se elimina.
- Realice una conmutación por error de prueba en el clúster para saber cuánto tiempo tarda el proceso en su caso de uso. Realizar una conmutación por error de prueba puede ayudarle a asegurarse de que la aplicación que accede al clúster de base de datos puede conectarse automáticamente al nuevo clúster después de una conmutación por error.

Recomendaciones de RAM de las instancias de base de datos

Para optimizar el rendimiento, asigne suficiente RAM para que el conjunto de trabajo resida casi por completo en la memoria. Para determinar si el conjunto de trabajo está en la memoria casi en su totalidad, examine la siguiente métrica en Amazon CloudWatch:

- `VolumeReadIOPS`: esta métrica mide el número medio de operaciones de E/S de lectura de un volumen del clúster (se muestra en intervalos de 5 minutos). El valor de `VolumeReadIOPS` debe ser pequeño y estable. En algunos casos, es posible que descubra que las operaciones de E/S de lectura están aumentando o son más elevadas de lo habitual. Si es así, investigue las instancias de base de datos del clúster de base de datos para determinar cuáles están causando este aumento de las operaciones de E/S.

> **Tip**
>
> Si su clúster de Aurora MySQL utiliza consulta en paralelo, es posible que vea un aumento en los valores `VolumeReadIOPS`. Las consultas en paralelo no utilizan el grupo de búfer. Por lo tanto, si bien las consultas son rápidas, este procesamiento optimizado puede dar como resultado un aumento de las operaciones de lectura y los cargos asociados.

- `BufferCacheHitRatio`: esta métrica mide el porcentaje de solicitudes que se responden desde la caché del búfer de una instancia de base de datos en su clúster de base de datos. Esta métrica proporciona información sobre la cantidad de datos que se está sirviendo desde la memoria.

  Una tasa de aciertos alta indica que la instancia de base de datos tiene suficiente memoria disponible. Si la tasa de aciertos es baja, eso indica que las consultas de esta instancia de base de datos van al disco con frecuencia. Investigue la carga de trabajo para ver qué consultas están provocando este comportamiento.

Si tras investigar la carga de trabajo determina que necesita más memoria, considere la posibilidad de escalar la instancia de base de datos a una clase con más RAM. Una vez hecho esto, puede investigar las métricas mencionadas anteriormente y seguir escalando el sistema según sea necesario. Si el clúster de Aurora tiene más de 40 TB, no utilice las clases de instancia db.t2, db.t3 ni db.t4.

Para obtener más información, consulte Métricas de Amazon CloudWatch para Amazon Aurora.

Controladores de bases de datos de AWS

Recomendamos el conjunto de controladores de AWS para la conectividad de las aplicaciones. Los controladores se han diseñado para permitir tiempos de transición y conmutación por error más rápidos y autenticarse con AWS Secrets Manager, AWS Identity and Access Management (IAM) e identidad federada. Los controladores de AWS se basan en la supervisión del estado del clúster de base de datos y en el conocimiento de la topología del clúster para determinar quién es el nuevo escritor. Este enfoque reduce los tiempos de transición y conmutación por error a segundos de un solo dígito, en comparación con las decenas de segundos de los controladores de código abierto.

A medida que se introducen nuevas características de servicio, el objetivo del conjunto de controladores de AWS es contar con soporte integrado para estas características de servicio.

Para obtener más información, consulte Conexión a clústeres de bases de datos Aurora con los controladores de AWS.

Monitorización de Amazon Aurora

Amazon Aurora proporciona diversas métricas e información de Amazon Aurora que puede monitorizar para determinar el estado y el rendimiento del clúster de base de datos de Aurora. Utilice diferentes herramientas, como la Consola de administración de AWS, la AWS CLI y la API de CloudWatch , para ver las métricas de Aurora. Puede ver las métricas combinadas de Información de rendimiento y CloudWatch en el panel de Información de rendimiento y monitorizar su instancia de base de datos. Para utilizar esta vista de monitorización, es necesario activar Información de rendimiento para la instancia de base de datos. Para obtener más información sobre la monitorización, consulte Visualización de las métricas combinadas en la consola de Amazon RDS.

Puede crear un informe de análisis de rendimiento para un período de tiempo específico y ver la información identificada y las recomendaciones para resolver los problemas. Para obtener más información, consulte, Creación de un informe de análisis de rendimiento en Información de rendimiento.

Trabajo con los grupos de parámetros de base de datos y grupos de parámetros de clúster de base de datos

Es recomendable que pruebe los cambios de los grupos de parámetros de base de datos y de grupos de parámetro de clústeres de base de datos en un clúster de base de datos prueba antes de aplicar cambios de grupos de parámetros al clúster de base de datos de producción. Si se configuran de forma incorrecta los parámetros del motor de base de datos, pueden producirse efectos adversos no deseados, como la degradación del rendimiento y la inestabilidad del sistema.

Tenga cuidado siempre que modifique los parámetros del motor de base de datos y cree una copia de seguridad del clúster de base de datos antes de modificar un grupo de parámetros de base de datos. Para obtener información acerca del procedimiento para realizar la copia de seguridad del clúster de base de datos, consulte Copias de seguridad y restauración de un clúster de base de datos de Amazon Aurora.

Vídeo sobre las prácticas recomendadas de Amazon Aurora

En el canal AWS Online Tech Talks de YouTube se incluye un vídeo de presentación de las prácticas recomendadas para crear y configurar un clúster de base de datos de Amazon Aurora más seguro y con alta disponibilidad. Vea el vídeo sobre Prácticas recomendadas para obtener una alta disponibilidad de Amazon Aurora.

Ejecución de una prueba de concepto de Amazon Aurora

A continuación, encontrará una explicación de cómo configurar una prueba de concepto de Aurora. Una prueba de concepto es una investigación que se realiza para ver si Aurora se adapta bien a su aplicación. La prueba de concepto le ayuda a comprender las características de Aurora en el contexto de sus propias aplicaciones de base de datos y cuál es el rendimiento de Aurora en comparación con el de su entorno de base de datos actual. También es útil para ver cuánto trabajo tendrá que dedicar a trasladar datos, migrar código SQL, ajustar el rendimiento y adaptar los procedimientos de administración actuales.

En este tema, se proporciona información e instrucciones generales para los procedimientos y decisiones de tipo general que deben adoptarse al ejecutar una prueba de concepto. Si desea consultar instrucciones detalladas, haga clic en los enlaces siguientes para llegar a la documentación de temas específicos.

Información general de una prueba de concepto de Aurora

Con una prueba de concepto de Amazon Aurora aprenderá cuánto le costará migrar los datos y las aplicaciones SQL existentes a Aurora. En dicha prueba, se aplican a escala aspectos importantes de Aurora, con un volumen de datos y una actividad representativos de su entorno de producción. El objetivo es asegurarse y confiar en que los puntos fuertes de Aurora estarán a la altura de los retos que le impulsan a ampliar su infraestructura de base de datos anterior. Cuando acabe la prueba de concepto, tendrá un plan sólido para realizar una comparación a mayor escala del rendimiento y la prueba de aplicación. En este punto, ya comprenderá cuáles son los principales elementos que tiene que desarrollar para llegar a una implementación de producción.

El siguiente consejo sobre prácticas recomendadas puede serle útil para evitar errores que se producen con frecuencia y que pueden provocar problemas durante la comparación. Sin embargo, en este tema no se cubre el proceso detallado paso a paso de realización de una comparación y un ajuste del rendimiento. Este procedimiento varía en función de la carga de trabajo y de las características de Aurora que use. Para obtener información detallada, consulte la documentación relacionada con el rendimiento, como Administración del rendimiento y el escalado para clústeres de base de datos Aurora, Mejoras del rendimiento de Amazon Aurora MySQL, Rendimiento y escalado para Amazon Aurora PostgreSQL y Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

La información que se proporciona en este tema se aplica principalmente a aplicaciones en las que la organización escribe el código y diseña el esquema, y que son compatibles con los motores de base de datos de código abierto MySQL y PostgreSQL. Si va a probar una aplicación comercial o un código generado por un marco de aplicaciones, puede que no tenga suficiente flexibilidad para aplicar todas las directrices. En dicho caso, contáctese con el representante de AWS para saber si existen prácticas recomendadas o casos prácticos de Aurora para su tipo de aplicación.

1. Identifique sus objetivos

Al evaluar Aurora como parte de una prueba de concepto, usted elige las medidas que van a realizarse y cómo determinar si un ejercicio se ha realizado correctamente.

Tiene que asegurarse de que la funcionalidad completa de la aplicación sea compatible con Aurora. Como las versiones principales de Aurora son compatibles por conexión con las versiones principales correspondientes de MySQL y PostgreSQL, la mayoría de las aplicaciones desarrolladas para estos motores también son compatibles con Aurora. Sin embargo, debe validar la compatibilidad aplicación por aplicación.

Por ejemplo, algunas de las opciones de configuración de un clúster de Aurora influyen en si se puede o se deben usar características determinadas de la base de datos. Puede comenzar, por ejemplo, por un clúster de Aurora de ámbito más general, conocido como clúster aprovisionado. Después, puede que le interese decidir si una configuración especializada como una consulta paralela o sin servidor le sería útil a su carga de trabajo.

Formúlese las siguientes preguntas para establecer y cuantificar sus objetivos:

- ¿Es compatible Aurora con todos los casos de uso funcionales de su carga de trabajo?
- ¿Qué volumen de conjunto de datos o nivel de carga quiere? ¿Puede escalar a dicho nivel?
- ¿Cuáles son sus requisitos específicos de latencia o desempeño de consultas? ¿Puede conseguirlos?
- ¿Cuál es el tiempo mínimo de inactividad planificada o sin planificar aceptable para su carga de trabajo? ¿Puede conseguirlo?
- ¿Cuáles son las métricas necesarias para un funcionamiento eficiente? ¿Puede monitorizarlas con precisión?
- ¿Es compatible Aurora con sus objetivos empresariales concretos como, por ejemplo, una reducción de costos, el aumento de la implementación o la velocidad de aprovisionamiento? ¿Tiene alguna forma de cuantificar dichos objetivos?

- ¿Puede cumplir todos los requisitos de seguridad y conformidad de su carga de trabajo?

Reserve tiempo para mejorar sus conocimientos sobre los motores de base de datos y las capacidades de plataforma de Aurora, y revise la documentación del servicio. Tome nota de todas las características que le pueden ser útiles para obtener los resultados deseados. Por ejemplo, puede interesarle la característica de consolidación de cargas de trabajo que se describe en la publicación del blog de bases de datos de AWS que trata de cómo planificar y optimizar Amazon Aurora con compatibilidad de MySQL para cargas de trabajo consolidadas. También puede interesarle la característica de escalado en función de la demanda, que se describe en Amazon Aurora Auto Scaling con réplicas de Aurora, en la Guía del usuario de Amazon Aurora, u otras características como las ganancias de rendimiento o las operaciones de bases de datos simplificadas.

2. Conozca las características de su carga de trabajo

Evalúe Aurora en el contexto del caso de uso para el que está destinado. Aurora es una buen opción para las cargas de trabajo de procesamiento de transacciones en línea (OLTP). También puede ejecutar informes en el clúster que contiene los datos OLTP en tiempo real sin aprovisionar un clúster de almacén de datos independiente. Para saber si su caso de uso entra dentro de estas categorías, averigüe si tiene las siguientes características:

- Un alto grado de simultaneidad, con decenas, centenas o miles de clientes a la vez.
- Un alto volumen de consultas con baja latencia (de milisegundos a segundos).
- Transacciones breves en tiempo real.
- Patrones de consulta altamente selectivos, con búsquedas basadas en índices.
- En el caso de HTAP, consultas analíticas que pueden aprovechar la característica de consultas paralelas de Aurora.

Cuando se elige una base de datos, uno de los factores clave es la velocidad de los datos. Si la velocidad es elevada, los datos se insertan y se actualizan con mucha frecuencia. En un sistema de este tipo, puede haber miles de conexiones y centenares de miles de consultas simultáneas que leen una base de datos y escriben en ella. Por lo general, las consultas que se realizan en sistemas de alta velocidad afectan a un número relativamente pequeño de filas y suelen obtener acceso a varias columnas de la misma fila.

Aurora está diseñado para trabajar con datos de alta velocidad. En función de la carga de trabajo, un clúster de Aurora con una única instancia de base de datos r4.16xlarge puede procesar más de 600 000 instrucciones `SELECT` por segundo. De nuevo, según la carga de trabajo, un clúster de este tipo puede procesar 200 000 sentencias `INSERT`, `UPDATE` y `DELETE` por segundo. Aurora es una base de datos de almacenamiento de filas ideal para las cargas de trabajo OLTP de alto volumen, alto rendimiento y alto paralelismo.

Aurora también puede ejecutar consultas en el mismo clúster que gestiona la carga de trabajo OLTP. Aurora soporta hasta 15 réplicas, cada una de las cuales está, en promedio, a 10 o 20 milisegundos de la instancia principal. Los analistas pueden consultar los datos OLTP en tiempo real sin copiar los datos en un clúster de almacenes de datos independiente. Dado que los clústeres de Aurora pueden usar la característica de consulta en paralelo, puede descargar un gran volumen de trabajo de procesamiento, filtrado y agregación en el subsistema de almacenamiento de Aurora de distribución masiva.

Aproveche esta fase de planificación para familiarizarse con las capacidades de Aurora, otros servicios de AWS, la Consola de administración de AWS y AWS CLI. Además, compruebe cómo funcionan con otras herramientas que tiene previsto utilizar en la prueba de concepto.

3. Practique con la Consola de administración de AWS o con la AWS CLI

En el siguiente paso, tendrá que practicar con la Consola de administración de AWS o con AWS CLI para familiarizarse con estas herramientas y Aurora.

Practique con la Consola de administración de AWS

Las siguientes actividades iniciales que se realizan con clústeres de la base de datos de Aurora tienen como objetivo principal familiarizarlo con el entorno de la Consola de administración de AWS y hacerle practicar la configuración y la modificación de clústeres de Aurora. Si usa motores de base de datos compatibles con MySQL y PostgreSQL junto con Amazon RDS, puede aprovechar sus conocimientos cuando utilice Aurora.

Aproveche el modelo de almacenamiento compartido y las características de Aurora como la replicación y las instantáneas; puede tratar clústeres completos de bases de datos como si fueran otro tipo de objeto que puede manipular a su voluntad. Puede configurar, eliminar y cambiar con frecuencia la capacidad de los clústeres de Aurora durante la prueba de concepto. Las elecciones de

capacidad, configuración de base de datos y disposición física de los datos que realice al principio no lo limitan.

Para comenzar, configure un clúster de Aurora vacío. Elija el tipo de capacidad provisioned (aprovisionado) y la ubicación regional para sus experimentos iniciales.

Establezca conexión con el clúster mediante un programa cliente como una aplicación de línea de comandos de SQL. Al principio establezca conexión utilizando el punto de enlace del clúster. Conéctese al punto de enlace para realizar cualquier operación de escritura, como instrucciones de lenguaje de definición de datos (DDL) y procesos de extracción, transformación y carga (ETL). Más adelante, en la prueba de concepto, conectará sesiones de uso intensivo de consultas usando el punto de enlace del lector, que se encargará de distribuir la carga de trabajo de las consultas entre numerosas instancias de base de datos del clúster.

Escale el clúster añadiendo más réplicas de Aurora. Para informarse sobre estos procedimientos, consulte Replicación con Amazon Aurora. Aumente o reduzca el escalado de las instancias de base de datos cambiando la clase de instancia de AWS. Comprenda cómo Aurora simplifica estos tipos de operaciones, ya que si más adelante sus estimaciones iniciales de la capacidad del sistema demuestran ser poco precisas, pueda modificarlas sin tener que volver a comenzar.

Cree una instantánea y restáurela en otro clúster.

Examine las métricas del clúster para ver la actividad en el tiempo y cómo las métricas se aplican a las instancias de base de datos del clúster.

Al principio puede serle útil familiarizarse con el uso de la Consola de administración de AWS para realizar estas tareas. Cuando comprenda qué puede hacer con Aurora, podrá automatizar dichas operaciones con AWS CLI. En las secciones siguientes encontrará información detallada acerca de los procedimientos y las prácticas recomendadas de estas actividades durante el período de prueba de concepto.

Practique con la AWS CLI

Recomendamos que automatice los procedimientos de implementación y administración, incluso en un entorno de prueba de concepto. Para ello, familiarícese con la AWS CLI si todavía no está familiarizado. Si usa motores de base de datos compatibles con MySQL y PostgreSQL junto con Amazon RDS, puede aprovechar sus conocimientos cuando utilice Aurora.

Normalmente, Aurora trabaja con grupos de instancias de bases de datos ordenadas en clústeres. Esto hace que en muchas operaciones se tenga que determinar qué instancias de base de datos

están asociadas a un clúster y luego se realicen operaciones administrativas en bucle en todas las instancias.

Por ejemplo, puede automatizar pasos como la creación de clústeres de Aurora y luego escalarlos mediante ampliación con clases de instancias más grandes o ampliarlos con instancias de bases de datos adicionales. Esta característica es útil si desea repetir cualquier etapa de su prueba de concepto y explorar escenarios condicionales con diferentes tipos de configuraciones de clústeres de Aurora.

Aprenda cuáles son las capacidades y limitaciones de las herramientas de implementación de infraestructura como AWS CloudFormation. Puede darse el caso de que actividades que realiza en un contexto de prueba de concepto no sean adecuadas en un entorno de producción. Por ejemplo, el comportamiento de CloudFormation en una modificación consiste en crear una instancia nueva y eliminar la actual, incluidos sus datos. Para obtener más detalles acerca de este comportamiento, consulte Comportamientos de actualización de los recursos de la pila en la guía del usuario de AWS CloudFormation.

4. Cree su clúster de Aurora

Con Aurora, puede explorar escenarios condicionales añadiendo instancias de base de datos al clúster y escalando mediante ampliación las instancias de base de datos a clases de instancias más potentes. También puede crear clústeres con distintos ajustes de configuración para que ejecuten la misma carga de trabajo uno al lado de otro. Aurora aporta una gran flexibilidad que le permite configurar, eliminar y volver a configurar clústeres de bases de datos. Teniendo en cuenta estas técnicas, le será útil practicarlas en las etapas iniciales del proceso de prueba de concepto. Para informarse de los procedimientos generales para crear clústeres de Aurora, consulte Creación de un clúster de base de datos de Amazon Aurora.

Para practicar, comience con un clúster que tenga la configuración que indicamos más abajo. Omita este paso solo si tiene en mente algunos casos de uso específicos. Por ejemplo, puede que le interese omitir este paso si su caso de uso necesita un tipo especializado de clúster de Aurora, O bien, puede que le interese omitirlo si necesita una combinación específica de motor de base de datos y versión.

- Desactivación de Easy create (Creación sencilla). Para la prueba de concepto, le recomendamos que preste mucha atención a todos los ajustes que elija a fin de que pueda crear clústeres idénticos o muy parecidos más adelante.

- Use una versión del motor de base de datos reciente. Estas combinaciones de motor de base de datos y versión tienen una compatibilidad de amplio espectro con otras características de Aurora y un uso de clientes sustancial para aplicaciones de producción.
  - Aurora MySQL versión 3.x (compatibilidad con MySQL 8.0)
  - Aurora PostgreSQL versión 15.x o 16.x
- Elija la plantilla Dev/Test (Desarrollo/Prueba). Esta opción no es significativa para sus actividades de prueba de concepto.
- Para DB instance class (Clase de instancia de base de datos), elija Memory Optimized classes (Clases optimizadas para memoria) y una de las clases de instancias xlarge. Más adelante, podrá subir o bajar el nivel de la clase de instancia.
- En Multi-AZ Deployment (Implementación Multi-AZ), elija Create an Aurora Replica or Reader node in different AZ (Crear una Réplica de Aurora o un nodo de lector en una AZ diferente). En muchos de los aspectos más útiles de Aurora se usan clústeres compuestos por numerosas instancias de base de datos. En todos los clústeres nuevos es normal comenzar siempre por dos instancias de base de datos como mínimo. Si se usa una zona de disponibilidad diferente para la otra instancia de base de datos, será más fácil probar diferentes escenarios de alta disponibilidad.
- Cuando elija un nombre para una instancia de base de datos, utilice una convención de nomenclatura genérica. Nunca se refiera a una instancia de base de datos del clúster como la instancia “escritora”, ya que, según las necesidades, esto rol lo asumirán instancias de base de datos diferentes. Le recomendamos que use algo parecido a `clustername-az-serialnumber`; por ejemplo `myprodappdb-a-01`. De esta manera, puede identificar de forma exclusiva la instancia de base de datos y su ubicación.
- Configure la retención de la copia de seguridad en un valor elevado para el clúster de Aurora. Si el período de retención es largo, puede realizar una recuperación a un momento dado (PITR) para un período de 35 días como máximo. Podrá restablecer la base de datos en un estado conocido después de haber ejecutado las pruebas con instrucciones DDL y de lenguaje de manipulación de datos (DML). También puede hacer una recuperación si por error borra o cambia datos.
- Habilite las características adicionales de recuperación, registro y monitorización al crear el clúster. Active todas las opciones disponibles bajo Backtrack, Información sobre rendimiento, Supervisión y Exportaciones de registro. Con estas opciones habilitadas, puede probar la idoneidad de características como el backtracking, la monitorización mejorada o la información sobre rendimiento en su carga de trabajo. También puede estudiar fácilmente el rendimiento y la resolución de problemas durante la prueba de concepto.

5. Configure el esquema

En el clúster de Aurora, configure las bases de datos, tablas, índices, claves externas y otros objetos del esquema para su aplicación. Si se está trasladando desde un sistema de base de datos compatible con MySQL o PostgreSQL, esta etapa será probablemente sencilla y fácil. Tendrá que usar la misma línea de comandos y sintaxis de SQL u otras aplicaciones cliente con las que ya está familiarizado con su motor de base de datos.

Para generar instrucciones SQL en el clúster, busque el punto de enlace de clúster y suministre el valor como parámetro de conexión a su aplicación cliente. Encontrará el punto de enlace del clúster en la pestaña Connectivity (Conectividad) de la página de detalles del clúster. El punto de enlace del clúster es el que está etiquetado como Writer (Escritor). El otro punto de enlace, etiquetado como Reader (Lector), representa una conexión de solo lectura que se puede proporcionar a usuarios finales que ejecuten informes u otras consultas de solo lectura. Para obtener ayuda con problemas relativos a la conexión del clúster, consulte Conexión a un clúster de base de datos Amazon Aurora.

Si migra el esquema y los datos desde otro sistema de base de datos, en este punto probablemente tendrá que introducir algún cambio. Los cambios de esquema se efectúan para que coincida con la sintaxis de SQL y las capacidades disponibles en Aurora. En este punto puede excluir algunas columnas, restricciones, desencadenantes u otros objetos del esquema. Esta operación es útil, sobre todo si los objetos requieren un trabajo adicional para que sean compatibles con Aurora y no son especialmente importantes para sus objetivos en la prueba de concepto.

Si realiza la migración desde un sistema de base de datos cuyo motor subyacente no sea el mismo que el de Aurora, considere usar AWS Schema Conversion Tool (AWS SCT) para simplificar el proceso. Para obtener más detalles, consulte la guía del usuario de AWS Schema Conversion Tool. Para obtener detalles generales sobre las actividades de migración y portabilidad, consulte el documento técnico AWS Migrating Your Databases to Amazon Aurora (Migración de bases de datos a Amazon Aurora).

Durante esta etapa, puede detectar si la configuración del esquema es poco eficiente en algunos aspectos como, por ejemplo, en la estrategia de indexación u otras estructuras de tabla como tablas con particiones. Esta baja eficiencia puede incrementarse si implementa la aplicación en un clúster que tenga numerosas instancias de base de datos y una carga de trabajo elevada. Decida si debe ajustar ahora estos aspectos del rendimiento o bien si esperará a actividades posteriores, como la prueba de referencia completa.

6. Importe los datos

Durante la prueba de concepto, tiene que importar los datos, o una muestra representativa de ellos, desde su antiguo sistema de base de datos. Si es práctico, configure como mínimo algunos datos en cada una de las tablas. Esto es útil para probar la compatibilidad de todos los tipos de datos y las características del esquema. Cuando haya probado las características básicas de Aurora, aumente la cantidad de datos. Cuando termine la prueba de concepto, deberá poder probar sus herramientas de ETL, consultas y carga de trabajo en general con un conjunto de datos que sea lo suficientemente grande como para obtener conclusiones precisas.

Para importar los datos de la copia de seguridad lógica o física a Aurora, puede usar varias técnicas. Para obtener información detallada, consulte Migración de datos a un clúster de base de datos de Amazon Aurora MySQL o Migración de datos a Amazon Aurora con compatibilidad con PostgreSQL, según el motor de base de datos que use en la prueba de concepto.

Experimente con las tecnologías y las herramientas ETL que esté pensando utilizar. Vea cuáles son las que mejor se adaptan a sus necesidades. Tenga en cuenta tanto el desempeño como la flexibilidad. Por ejemplo, algunas herramientas ETL realizan una transferencia única, mientras que otras utilizan una replicación en curso desde el sistema antiguo hasta Aurora.

Si realiza la migración desde un sistema compatible con MySQL hasta Aurora MySQL, puede usar las herramientas de transferencia de datos nativas. Es el mismo caso que si migra desde un sistema compatible con PostgreSQL hasta Aurora PostgreSQL. Si migra desde un sistema de base de datos que usa un motor subyacente que no es el mismo que el de Aurora, puede experimentar con AWS Database Migration Service (AWS DMS). Para obtener más detalles sobre AWS DMS, consulte la guía del usuario de AWS Database Migration Service.

Para obtener más detalles sobre las actividades de migración y transferencia, consulte el documento técnico de AWS del manual de migración de Aurora.

7. Transfiera su código SQL

La prueba de SQL y las aplicaciones asociadas es más o menos laboriosa, en función de los diferentes casos. El nivel de trabajo depende, en concreto, de si la migración se efectúa desde un sistema compatible con MySQL o PostgreSQL, o desde otro tipo de sistema.

- Si la migración se efectúa desde RDS for MySQL o RDS for PostgreSQL, los cambios que SQL necesita son tan pequeños que puede probar el código SQL original con Aurora e incorporar manualmente los cambios necesarios.

- Igualmente, si efectúa la migración desde una base de datos local compatible con MySQL o PostgreSQL, puede probar el código SQL original e incorporar manualmente los cambios.
- Si realiza la migración desde otra base de datos comercial, los cambios que debe efectuar en SQL son más extensos. En dicho caso, piense en utilizar AWS SCT.

Durante esta etapa, puede detectar si la configuración del esquema es poco eficiente en algunos aspectos como, por ejemplo, en la estrategia de indexación u otras estructuras de tabla como tablas con particiones. Decida si debe ajustar ahora estos aspectos del rendimiento o bien si esperará a actividades posteriores, como la prueba de referencia completa.

Puede verificar la lógica de conexión de la base de datos en su aplicación. Para aprovechar el procesamiento distribuido de Aurora, es posible que deba usar conexiones diferentes para las operaciones de lectura y escritura, así como tener sesiones relativamente cortas para las operaciones de consulta. Para obtener más información acerca de las conexiones, consulte 9. Conéctese a Aurora.

Tenga en cuenta si ha tenido que transigir o hacer concesiones para solucionar problemas en la base de datos de producción. Integre tiempo en el programa de realización de la prueba de concepto para introducir mejoras en las consultas y el diseño de esquema. Para evaluar si puede obtener fácilmente beneficios en el rendimiento, los costos operativos y la escalabilidad, pruebe las aplicaciones original y modificada, una al lado de la otra, en clústeres de Aurora diferentes.

Para obtener más detalles sobre las actividades de migración y transferencia, consulte el documento técnico de AWS del manual de migración de Aurora.

8. Especifique las opciones de configuración

Al efectuar la prueba de concepto de Aurora, tiene la posibilidad también de revisar los parámetros de configuración de la base de datos. Probablemente la configuración de MySQL o PostgreSQL ya esté ajustada para el rendimiento y la escalabilidad de su entorno actual. El subsistema de almacenamiento de Aurora está adaptado y ajustado a un entorno basado en la nube distribuido con un subsistema de almacenamiento de alta velocidad. En consecuencia, muchas configuraciones de motor de base de datos antiguas no se pueden aplicar. Le recomendamos que realice los experimentos iniciales con los ajustes de configuración de Aurora predeterminados. Vuelva a aplicar la configuración de su entorno actual solo si se producen atascos de rendimiento o de escalabilidad. Si está interesado en este tema, puede profundizar consultando la Introducción del motor de almacenamiento Aurora en el blog de la base de datos de AWS.

Aurora facilita la reutilización de los ajustes de configuración óptimos para una aplicación o un caso de uso determinado. En vez de editar un archivo de configuración independiente por cada instancia de base de datos, administre conjuntos de parámetros que después asignará a clústeres enteros o a instancias de bases de datos específicas. Por ejemplo, la configuración de la zona horaria se aplica a todas las instancias de base de datos del clúster, y puede ajustar el valor del tamaño de la memoria caché de la página para cada instancia de base de datos.

Comience con uno de los conjuntos de parámetros predeterminados y aplique cambios solo a aquellos parámetros que tenga que ajustar. Para obtener información detallada acerca de cómo trabajar con grupos de parámetros, consulte Parámetros del clúster de base de datos de Amazon Aurora y de instancia de base de datos. Para informarse de los ajustes de configuración que se pueden o no se pueden aplicar a clústeres de Aurora, consulte Parámetros de configuración de Aurora MySQL o Parámetros de Amazon Aurora PostgreSQL. según el motor de base de datos.

9. Conéctese a Aurora

Cuando realice la configuración inicial del esquema y los datos, y ejecute las consultas de ejemplo, verá que puede conectarse a diferentes puntos de enlace de un clúster de Aurora. El punto de enlace que use dependerá de si la operación es de lectura, como una instrucción `SELECT`, o de escritura, como una instrucción `CREATE` o `INSERT`. A medida que aumente la carga de trabajo en un clúster de Aurora y experimente con características de Aurora, es importante que la aplicación asigne cada operación al punto de enlace adecuado.

Si usa el punto de enlace del clúster destinado a operaciones de escritura, se conectará siempre a una instancia de base de datos del clúster que tenga capacidad de lectura-escritura. De forma predeterminada, solo una instancia de base de datos de un clúster de Aurora tiene capacidad de lectura-escritura. Esta instancia de la base de datos se conoce como la instancia principal. Si la instancia principal original deja de esta disponible, Aurora activa un mecanismo de conmutación por error y otra instancia de base de datos pasa a ocupar el lugar de la principal.

Igualmente, si dirige instrucciones `SELECT` al punto de enlace del lector, extiende el trabajo de procesamiento de consultas a las instancias de base de datos del clúster. Cada conexión del lector se asigna a una instancia de base de datos diferente mediante una resolución DNS de turno rotativo. Si efectúa la mayor parte del trabajo de consulta en las réplicas de Aurora de base de datos de solo lectura, reducirá la carga de la instancia principal y la liberará para que pueda gestionar las instrucciones DDL y DML.

Al utilizar estos puntos de enlace, reducirá la dependencia de los nombres de host no modificables y, al mismo tiempo, ayudará a la aplicación a recuperarse más rápidamente de los errores de las instancias de base de datos.

> **Note**
>
> Aurora también tiene puntos de enlace personalizados que usted crea. Normalmente, estos puntos de enlace no se necesitan durante una prueba de concepto.

Las réplicas de Aurora están sujetas a un retraso de réplica, aunque por lo general dicho retraso sea de 10 a 20 milisegundos. Puede monitorizar el retraso de replicación y decidir si entra dentro del rango de requisitos de coherencia de sus datos. A veces, las consultas de lectura pueden necesitar que la coherencia de lectura sea sólida (coherencia de lectura después de escritura). En dichos casos, puede seguir utilizando para ellas el punto de enlace del clúster y no el punto de enlace del lector.

Para aprovechar plenamente las capacidades de Aurora para la ejecución en paralelo distribuida, puede que tenga que cambiar la lógica de conexión. El objetivo es evitar enviar todas las solicitudes de lectura a la instancia principal. Las réplicas de Aurora de solo lectura están a la espera, listas para hacerse cargo de las instrucciones `SELECT`. Codifique la lógica de aplicación para que use el punto de enlace adecuado para cada tipo de operación. Siga estas instrucciones generales:

- Evite utilizar una única cadena de conexión no modificable para todas las sesiones de base de datos.
- Si es práctico, escriba operaciones como instrucciones DDL o DML en las funciones, en el código de aplicación cliente. De esta forma, puede hacer que diferentes tipos de operaciones usen conexiones específicas.
- Realice funciones separadas para las operaciones de consulta. Aurora asigna cada nueva conexión que se realice al punto de enlace del lector a una réplica de Aurora diferente a fin de equilibrar la carga de las aplicaciones con un uso elevado de la lectura.
- En el caso de las operaciones que usan conjuntos de consultas, cierre y vuelva a abrir la conexión al punto de enlace del lector cuando acabe cada conjunto de consultas relacionadas. Agrupe las conexiones si esta característica está disponible en su pila de software. Dirigir las consultas a diferentes conexiones ayuda a Aurora a distribuir la carga de trabajo de lectura entre las instancias de bases de datos del clúster.

Para obtener información general acerca de la administración de conexiones y los puntos de enlace de Aurora, consulte Conexión a un clúster de base de datos Amazon Aurora. Para profundizar en el tema, consulte Manual de administrador de bases de datos MySQL de Aurora: administración de conexiones.

10. Ejecute la carga de trabajo

Una vez que haya establecido los valores de esquema, datos y configuración, puede empezar a practicar con el clúster ejecutando la carga de trabajo. La carga de trabajo que use en la prueba de concepto debe reflejar los principales aspectos de su carga de trabajo de producción. Le recomendamos que las decisiones que tome sobre el rendimiento se basen siempre en pruebas y cargas de trabajo reales, y no en referencias sintéticas como sysbench o TPC-C. Cuando sea práctico, recopile medidas que se basen en su esquema, sus patrones de consulta y su volumen de uso.

Además de tener en cuenta el aspecto práctico, replique las condiciones reales en las que se ejecutará la aplicación. Por ejemplo, normalmente se ejecuta el código de aplicación en instancias Amazon EC2, en la misma región de AWS y en la misma Virtual Private Cloud (VPC) que el clúster de Aurora. Si la aplicación de producción se ejecuta en varias instancias EC2 que se extienden por varias zonas de disponibilidad, configure el entorno de la prueba de concepto de la misma manera. Para obtener más información sobre las regiones de AWS, consulte Regiones y zonas de disponibilidad en la guía del usuario de Amazon RDS. Para obtener más información acerca del servicio Amazon VPC, consulte ¿Qué es Amazon VPC? en la Guía del usuario de Amazon VPC.

Una vez que haya comprobado que las características básicas de su aplicación funcionan y que puede obtener acceso a los datos a través de Aurora, puede practicar aspectos del clúster de Aurora. Puede que le interese practicar características como conexiones simultáneas con equilibrio de carga, transacciones simultáneas y replicaciones automáticas.

Cuando llegue a este punto, debería estar ya familiarizado con los mecanismos de transferencia de datos y, por lo tanto, poder ejecutar pruebas con una proporción más grande de datos de muestra.

En esta etapa puede estudiar cómo repercuten los cambios en los ajustes de configuración como, por ejemplo, los límites de memoria o los de conexión. Repase los procedimientos que ha explorado en 8. Especifique las opciones de configuración.

También puede experimentar con mecanismos como la creación y restauración de instantáneas. Por ejemplo, puede crear clústeres con clases de instancias de AWS diferentes, números de réplicas de AWS diferentes, etc. Luego, en cada clúster, puede restaurar la misma instantánea que contiene su

esquema y todos los datos. Para obtener información detallada sobre este ciclo, consulte Creación de una instantánea de clúster de base de datos y Restauración de una instantánea de clúster de base de datos.

11. Mida el rendimiento

Las prácticas recomendadas de esta área están diseñadas para garantizar que se configuren las herramientas y los procesos adecuados para aislar rápidamente los comportamientos anormales durante las operaciones de carga de trabajo. También se configuran para que pueda identificar de forma fiable todas las causes aplicables.

Siempre puede ver el estado actual de su clúster o examinar las tendencias a lo largo del tiempo, consultando la pestaña Monitoring (Monitorización). Esta pestaña está disponible en la página de detalles de la consola de cada clúster o de la instancia de base de datos de Aurora. En ella se muestran las métricas del servicio de monitorización de Amazon CloudWatch en forma de gráficos. Puede filtrar las métricas por nombre, por instancia de base de datos y por período de tiempo.

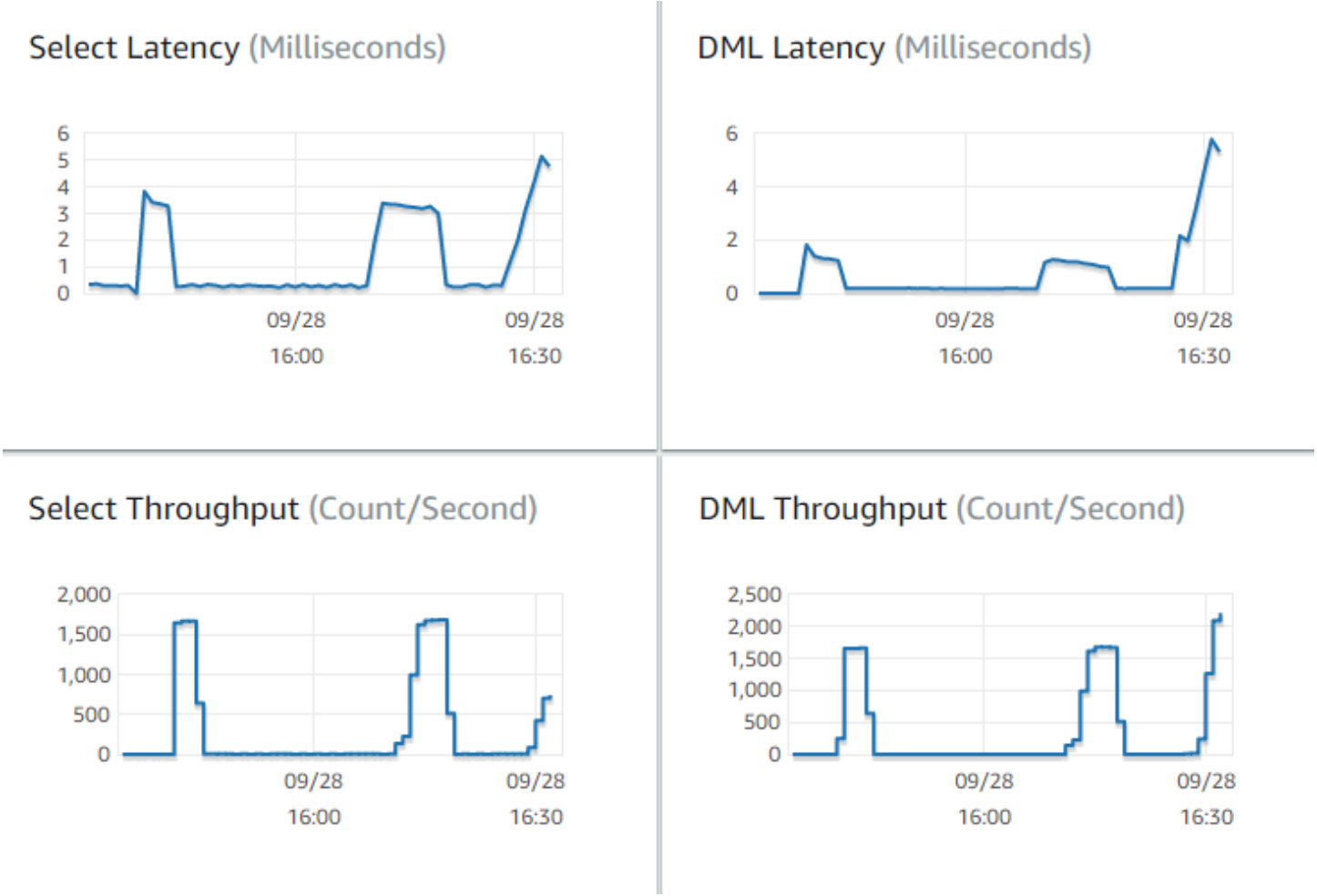
Para tener a su disposición más opciones en la pestaña Monitoring (Monitorización), habilite Enhanced Monitoring (Monitorización mejorada) y Performance Insights (Información sobre rendimiento) en la configuración del clúster. También puede habilitar más adelante estas opciones si no las ha seleccionado al configurar el clúster.

El rendimiento se mide principalmente basándose en los gráficos que muestran la actividad de todo el clúster de Aurora. Puede verificar si las réplicas de Aurora tienen tiempos de carga y respuesta similares. También puede ver cómo se divide el trabajo entre la instancia principal de lectura-escritura y las réplicas de Aurora de solo lectura. Si detecta un desequilibrio en las instancias de base de datos o un problema que afecta a una sola instancia de base de datos, consulte la pestaña Monitoring (Monitorización) de la instancia afectada.

Cuando haya configurado el entorno y la carga de trabajo real para que emulen su aplicación de producción, podrá evaluar el rendimiento de Aurora. A continuación le indicamos las preguntas más importantes que tiene que plantearse:

- ¿Cuántas consultas por segundo procesa Aurora? Para saber la respuesta, consulte las métricas de Throughput (Desempeño) para ver las cifras de diversos tipos de operaciones.
- ¿Cuánto tiempo tarda, en promedio, Aurora en procesar una consulta determinada? Para saber la respuesta, consulte las métricas de Latency (Latencia) para ver las cifras de diversos tipos de operaciones.

Para ver las métricas de rendimiento y latencia, consulte la pestaña Monitorización de un clúster de Aurora en la consola de Amazon RDS. La siguiente captura de pantalla muestra un ejemplo de las métricas Seleccionar latencia, Latencia de DML, Seleccionar rendimiento y Velocidad de DML en la pestaña Monitorización.



Si puede hacerlo, establezca valores de referencia para estas métricas en su entorno actual. Si esta operación no es práctica, cree una base de referencia en el clúster de Aurora ejecutando una carga de trabajo que sea equivalente a su aplicación de producción. Por ejemplo, ejecute su carga de trabajo de Aurora con un número similar de usuarios y consultas simultáneos. Luego observe cómo cambian los valores a medida que va experimentando con diferentes clases de instancias, tamaños de clúster, ajustes de configuración, etc.

Si los resultados del desempeño no están a la altura de lo que esperaba, profundice en la investigación de los factores que repercuten en el rendimiento de la base de datos para su carga de trabajo. Igualmente, si las cifras de la latencia son superiores a lo que esperaba, profundice en las causas. Para ello, monitorice las métricas secundarias del servidor de base de datos (CPU,

memoria, etc.). De esta forma podrá ver si las instancias de base de datos están próximas a sus límites. También podrá saber cuánta capacidad adicional les queda a las instancias de bases de datos para gestionar más consultas simultáneas, consultas para tablas más grandes, etc.

> **Tip**
>
> Para detectar valores de métricas que se salen de los rangos previstos, configure alarmas de CloudWatch.

Cuando evalúe la capacidad y el tamaño de clúster ideales de Aurora, puede encontrar la configuración que obtiene el rendimiento máximo de la aplicación sin aprovisionar recursos en exceso. En este sentido, un factor importante es encontrar el tamaño adecuado de las instancias de base de datos de clúster de Aurora. Comience seleccionando un tamaño de instancia cuya capacidad de memoria y CPU sea similar a la de su entorno de producción actual. Recopile las cifras de desempeño y latencia de la carga de trabajo con ese tamaño de instancia. Luego, escale el tamaño al siguiente tamaño más grande. Compruebe si los resultados del desempeño y la latencia mejoran. Reduzca también el tamaño de la instancia para ver si los resultados de la latencia y el desempeño siguen siendo los mismos. Su objetivo es obtener el máximo desempeño, con el nivel de latencia más bajo, en la instancia más pequeña posible.

> **Tip**
>
> Dé a los clústeres de Aurora y a las instancias de base de datos la capacidad suficiente como para poder gestionar los picos de tráfico repentinos e impredecibles. Para las bases de datos críticas para el trabajo, deje como mínimo un 20 % de espacio de CPU y de capacidad de memoria libre.

Ejecute pruebas de rendimiento el tiempo suficiente como para medir el rendimiento de la base de datos en un estado constante y flexible. Puede que tenga que ejecutar la carga de trabajo durante varios minutos o incluso unas cuantas horas para llegar a este estado constante. Es normal que, al principio de una ejecución, haya algunas variaciones. Estas variaciones se deben a que cada réplica de Aurora activa sus cachés en función de las consultas `SELECT` que gestiona.

El mejor rendimiento de Aurora se obtiene con cargas de trabajo transaccionales en las que hay numerosos usuarios y consultas simultáneos. Para asegurarse de que la carga sea suficiente para un rendimiento óptimo, ejecute referencias con multiprocesos, o bien ejecute varias instancias de las

pruebas de rendimiento a la vez. Mida el rendimiento con centenares o incluso miles de subprocesos cliente a la vez. Simule el número de subprocesos simultáneos que prevé tener en su entorno de producción. También puede ejecutar pruebas de estrés adicionales con más subprocesos para medir la escalabilidad de Aurora.

12. Pruebe la alta disponibilidad de Aurora

Muchas de las principales características de Aurora tienen una alta disponibilidad. Se trata de las características de replicación automática, conmutación por error automática, copia de seguridad automática con restauración a un momento dado y capacidad para añadir instancias de base de datos al clúster. La seguridad y la fiabilidad de dichas características es importante para las aplicaciones críticas.

La evaluación de estas características requiere adoptar una visión específica. En las actividades anteriores, como la medición del rendimiento, se observaba el rendimiento del sistema cuando todo funcionaba correctamente. Sin embargo, para probar la alta disponibilidad, es preciso enfocar la observación desde el estudio de las peores situaciones que puedan producirse. Ha de tener en cuenta distintos tipos de errores, incluso si tales casos son excepcionales. Por ejemplo, puede introducir problemas a propósito para asegurarse de que el sistema se recupere deprisa y correctamente.

> **Tip**
>
> Para una prueba de concepto, configure todas las instancias de base de datos de un clúster de Aurora con la misma clase de instancia de AWS. Con ello, será posible probar las características de disponibilidad de Aurora sin grandes cambios en el rendimiento ni la escalabilidad, ya que saca fuera de línea las instancias de base de datos para simular los errores.

Le recomendamos que use como mínimo dos instancias en cada clúster de Aurora. Las instancias de base de datos de un clúster de Aurora pueden extenderse hasta un máximo de tres zonas de disponibilidad (AZ). Ubique las dos o tres primeras instancias de base de datos de cada AZ diferente. Cuando comience a utilizar clústeres más grandes, extienda sus instancias de base de datos a todas las zonas de disponibilidad de la región de AWS. Al hacerlo aumentará su capacidad de tolerancia a errores. Aunque un problema afecte a una AZ entera, Aurora puede realizar una conmutación por error a una instancia de base de datos de otra AZ. Si tiene un clúster con más de tres instancias, distribuya las instancias de base de datos de la forma más homogénea posible entre las tres AZ.

> **Tip**
>
> El almacenamiento de un clúster de Aurora no depende de las instancias de base de datos. El almacenamiento de cada clúster de Aurora se extiende siempre a tres AZ.
> Cuando pruebe las características de alta disponibilidad, use siempre instancias de base de datos con capacidad idéntica en el clúster de prueba. Con esto evitará cambios imprevistos en el rendimiento, la latencia, etc. siempre que una instancia de base de datos sustituya a otra.

Para aprender a simular condiciones de error para probar las características de alta disponibilidad, consulte Pruebas de Amazon Aurora MySQL por medio de consultas de inserción de errores.

En la ejecución de la prueba de concepto, uno de los objetivos es encontrar el número ideal de instancias de bases de datos y la clase de instancia óptima para dichas instancias de base de datos. Para ello, tiene que equilibrar los requisitos de alta disponibilidad y rendimiento.

En Aurora, cuantas más instancias de bases de datos tenga en un clúster, más beneficiada saldrá la alta disponibilidad. Si tiene más instancias de base de datos, mejorará también la escalabilidad de las aplicaciones que realicen una lectura intensiva. Aurora puede distribuir múltiples conexiones para consultas `SELECT` entre las réplicas de Aurora de solo lectura.

Por otra parte, si limita el número de instancias de base de datos reducirá el tráfico de replicación desde el nodo principal. El tráfico de replicación consume ancho de banda de red, que es otro aspecto del rendimiento y la escalabilidad generales. Por lo tanto, para las aplicaciones OLTP de uso intensivo de escritura, es mejor que se incline por un número más pequeño de instancias de base de datos grandes que por muchas instancias de base de datos pequeñas.

En un clúster de Aurora típico, una instancia de base de datos (la instancia principal) gestiona todas las instrucciones DDL y DML. Las demás instancias de base de datos (las réplicas de Aurora) solo se encargan de las instrucciones `SELECT`. Aunque las instancias de base de datos no realizan exactamente la misma cantidad de trabajo, recomendamos que use la misma clase de instancia para todas las instancias de base de datos del clúster. Así, si se produce un error y Aurora promociona una instancia de base de datos de solo lectura como la nueva instancia principal, dicha instancia principal tendrá la misma capacidad que antes.

Si tiene que utilizar instancias de base de datos con capacidades diferentes en el mismo clúster, configure niveles de conmutación por error para las instancias de base de datos. Estos niveles

determinarán el orden de promoción de las réplicas de Aurora mediante el mecanismo de conmutación por error. Ponga las instancias de base de datos que sean mucho más grandes o pequeñas que las demás en un nivel de conmutación por error más bajo. Así se asegurará de que se sean las últimas en ser elegidas para una promoción.

Practique las características de recuperación de datos de Aurora, como la restauración automática a un momento dado, las instantáneas manuales y su restauración, y el backtracking de clústeres. Si es pertinente, copie instantáneas en otras regiones de AWS y restáurelas en otras regiones de AWS para imitar escenarios de recuperación de desastres (DR).

Investigue los requisitos de su organización para los objetivos de tiempo de recuperación (RTO), los objetivos de punto de recuperación (RPO) y la redundancia geográfica. La mayoría de las organizaciones agrupan estos elementos en la categoría más general de recuperación de desastres. Evalúe las características de alta disponibilidad de Aurora descritas en esta sección en el contexto de su proceso de recuperación de desastres, a fin de asegurarse de que los requisitos de RTO y RPO se cumplan.

13. Qué hacer a continuación

Al finalizar un proceso de prueba de concepto realizado correctamente, confirmará que Aurora es una solución adecuada para usted basándose en la carga de trabajo prevista. A lo largo del proceso anterior, ha comprobado cómo funciona Aurora en un entorno operativo realista y ha medido su funcionamiento de acuerdo con sus criterios de éxito.

Después de configurar su entorno de base de datos y ejecutarlo con Aurora, puede pasar a los pasos de evaluación más detallados que lo conducirán a la migración e implementación de producción finales. Según cuál sea su situación, dichos pasos adicionales se incluirán o no en el proceso de prueba de concepto. Para obtener más detalles sobre las actividades de migración y transferencia, consulte el documento técnico de AWS del manual de migración de Aurora.

En otro paso adicional, estudie las configuraciones de seguridad adecuadas para su carga de trabajo y diseñadas para cumplir los requisitos de seguridad de un entorno de producción. Planifique los controles que deben implantarse para proteger el acceso a las credenciales de usuario maestro del clúster de Aurora. Defina los roles y las responsabilidades de los usuarios de la base de datos para controlar el acceso a los datos almacenados en el clúster de Aurora. Tenga en cuenta los requisitos de acceso a la base de datos de las aplicaciones, scripts y herramientas o servicios de terceros. Explore servicios y características de AWS como AWS Secrets Manager y autenticación de IAM de AWS Identity and Access Management.

Cuando llegue a este punto, debería conocer los procedimientos y las prácticas recomendadas para ejecutar pruebas de referencia con Aurora. Puede darse el caso de que necesite realizar un ajuste adicional del rendimiento. Para obtener información detallada, consulteAdministración del rendimiento y el escalado para clústeres de base de datos Aurora, Mejoras del rendimiento de Amazon Aurora MySQL, Rendimiento y escalado para Amazon Aurora PostgreSQL y Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora. Si realiza un ajuste adicional, tiene que estar familiarizado con las métricas que ha recopilado durante la prueba de concepto. En un paso siguiente, podría tener que crear clústeres nuevos con opciones diferentes para los ajustes de configuración, el motor de base de datos y la versión de la base de datos. O bien, podría tener que crear tipos especializados de clústeres de Aurora para adaptarse a las necesidades de casos de uso específicos.

Por ejemplo, puede explorar clústeres de consulta en paralelo de Aurora para aplicaciones de procesamiento de transacciones híbridas o procesamiento analítico (HTAP). Si una distribución geográfica extensa es fundamental para la recuperación de desastres o para minimizar la latencia, puede explorar bases de datos globales de Aurora. Si su carga de trabajo es intermitente o utiliza Aurora en un escenario de desarrollo y prueba, puede explorar clústeres de Aurora Serverless.

Sus clústeres de producción también podrán necesitar grandes volúmenes de conexiones de entrada. Para aprender dichas técnicas, consulte el documento técnico de AWS del manual de administrador de base de datos de Aurora MySQL: administración de conexiones.

Si, después de la prueba de concepto, decide que su caso de uso no es adecuado para Aurora, considere los siguientes servicios de:AWS

- Para casos de uso puramente analíticos, las cargas de trabajo se benefician de un formato de almacenamiento en columna y otras características más adecuadas para las cargas de trabajo OLAP. Los servicios de AWS que abordan tales casos de uso incluyen los siguientes:
  - Amazon Redshift
  - Amazon EMR
  - Amazon Athena
- Muchas cargas de trabajo disponen de una combinación de Aurora con uno o varios de estos servicios. Puede mover datos entre estos servicios con los siguientes productos:
  - AWS Glue
  - AWS DMS
  - Importación desde Amazon S3, tal y como se describe en la Guía del usuario de Amazon Aurora .

- Exportación a Amazon S3, tal y como se describe en la Guía del usuario de Amazon Aurora.
- Numerosas herramientas ETL conocidas más.

Seguridad en Amazon Aurora

La seguridad en la nube de AWS es la mayor prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y un centro de datos que están diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Auditores independientes prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener información acerca de los programas de conformidad que se aplican a Amazon Aurora (Aurora), consulte los servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. Usted también es responsable de otros factores incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon Aurora. En los siguientes temas, se le mostrará cómo configurar Amazon Aurora para satisfacer los objetivos de seguridad y conformidad. También puede aprender a utilizar otros servicios de AWS que ayuden a supervisar y proteger los recursos de Amazon Aurora.

Puede administrar el acceso a los recursos y las bases de datos de Amazon Aurora en un clúster de base de datos. El método que se utiliza para controlar el acceso depende del tipo de tarea que el usuario necesite realizar con Amazon Aurora:

- Ejecute el clúster de base de datos en una nube privada virtual (VPC) en función del servicio de Amazon VPC para el posible control de acceso de red más grande. Para obtener más información acerca de la creación de un clúster de base de datos en una VPC, consulte VPC de Amazon y Amazon Aurora.
- Utilice políticas de AWS Identity and Access Management (IAM) para asignar permisos que determinen quién puede administrar los recursos de Amazon Aurora. Por ejemplo, puede utilizar IAM para determinar quién tiene permiso para crear, describir, modificar y eliminar clústeres de bases de datos, etiquetar recursos o modificar grupos de seguridad.

Para revisar ejemplos de política de IAM, consulte Ejemplos de políticas basadas en identidad para Amazon Aurora.

- Utilice grupos de seguridad para controlar las direcciones IP o instancias de Amazon EC2 que pueden conectarse a las bases de datos de un clúster de bases de datos. Cuando se crea un clúster de bases de datos por primera vez, su firewall impide cualquier acceso a las bases de datos, salvo si se cumplen las reglas especificadas por un grupo de seguridad asociado.
- Utilice la capa de conexión segura (SSL) o seguridad de la capa de transporte (TLS) con clústeres de base de datos que ejecuten Aurora MySQL o Aurora PostgreSQL. Para obtener más información acerca del uso de SSL/TLS con un clúster de bases de datos, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.
- Utilice el cifrado de Amazon Aurora para proteger los clústeres de base de datos y las instantáneas en reposo. El cifrado de Amazon Aurora utiliza el algoritmo de cifrado AES-256 estándar del sector para cifrar los datos en el servidor que aloja el clúster de bases de datos. Para obtener más información, consulte Cifrado de recursos de Amazon Aurora .
- Utilice las características de seguridad del motor de base de datos para controlar quién puede iniciar sesión en las bases de datos en un clúster de bases de datos. Estas características funcionas de igual forma que si la base de datos estuviera en su red local.

  Para obtener información sobre seguridad con Aurora MySQL, consulte Seguridad con Amazon Aurora MySQL. Para obtener información sobre seguridad con Aurora PostgreSQL, consulte Seguridad con Amazon Aurora PostgreSQL.

Aurora forma parte del servicio de base de datos administrada de Amazon Relational Database Service (Amazon RDS). Amazon RDS es un servicio web que facilita la configuración, el funcionamiento y el escalado de una base de datos relacional en la nube. Si no está familiarizado con Amazon RDS, consulte la Guía del usuario de Amazon RDS.

Aurora incluye un subsistema de almacenamiento de alto rendimiento. Sus motores de base de datos compatibles con MySQL y PostgreSQL están personalizados para aprovechar su almacenamiento de rápida distribución. Aurora también automatiza y estandariza la agrupación en clústeres y la reproducción de base de datos, que suelen ser algunos de los aspectos más problemáticos de la configuración y administración de las bases de datos.

En Amazon RDS y Aurora, puede acceder mediante programación de la API de RDS y puede utilizar la AWS CLI para acceder de forma interactiva a la API de RDS. Algunas operaciones de la API de RDS y los comandos de AWS CLI se aplican a Amazon RDS y a Aurora, mientras que otros se

aplican a Amazon RDS o a Aurora. Para obtener información sobre operaciones de API de RDS, consulte Referencia de API de Amazon RDS. Para obtener más información acerca de AWS CLI, consulte la referencia de AWS Command Line Interface para Amazon RDS.

> **Note**
>
> Solo tiene que configurar la seguridad para sus casos de uso. No tiene que configurar el acceso de seguridad para procesos que Amazon Aurora administra. Entre ellos, se incluye la creación de copias de seguridad, la conmutación por error automática y otros procesos.

Para obtener más información acerca de la administración del acceso a los recursos de Amazon Aurora y las bases de datos de un clúster de bases de datos, consulte los siguientes temas.

Temas

- Autenticación de bases de datos con Amazon Aurora
- Administración de contraseñas con Amazon Aurora y AWS Secrets Manager
- Protección de datos en Amazon RDS
- Administración de la identidad y el acceso en Amazon Aurora
- Registro y monitoreo en Amazon Aurora
- Validación de la conformidad en Amazon Aurora
- Resiliencia en Amazon Aurora
- Seguridad de la infraestructura en Amazon Aurora
- La API de Amazon RDS y los puntos de enlace de la VPC de tipo interfaz (AWS PrivateLink)
- Prácticas recomendadas de seguridad para Amazon Aurora
- Control de acceso con grupos de seguridad
- Privilegios de la cuenta de usuario maestro
- Uso de roles vinculados a servicios de Amazon Aurora
- VPC de Amazon y Amazon Aurora

Autenticación de bases de datos con Amazon Aurora

Amazon Aurora admite varias formas de autenticar usuarios de bases de datos.

La autenticación de contraseña está disponible de forma predeterminada para todos los clústeres de base de datos. Para Aurora MySQL y Aurora PostgreSQL, también puede añadir una o ambas autenticaciones de base de datos de IAM y Kerberos para el mismo clúster de base de datos.

La autenticación con contraseña, de Kerberos y de base de datos de IAM utilizan diferentes métodos de autenticación en la base de datos. Por lo tanto, un usuario específico puede iniciar sesión en una base de datos mediante un solo método de autenticación.

Para PostgreSQL, utilice solo una de las siguientes configuraciones de rol para un usuario de una base de datos específica:

- Para utilizar la autenticación de base de datos de IAM, asigne el rol `rds_iam` al usuario.
- Para utilizar la autenticación de Kerberos, asigne el rol `rds_ad` al usuario.
- Para utilizar la autenticación con contraseña, no asigne ninguno de los roles `rds_iam` o `rds_ad` al usuario.

No asigne el rol `rds_iam` ni el rol `rds_ad` a un usuario de una base de datos de PostgreSQL, de forma directa o indirecta, mediante el acceso de concesión anidada. Si el rol `rds_iam` se agrega al usuario maestro, la autenticación de IAM tiene prioridad sobre la autenticación con contraseña, por lo que el usuario maestro tiene que iniciar sesión como usuario de IAM.

> **Important**
>
> Le recomendamos encarecidamente que no utilice el usuario maestro directamente en sus aplicaciones. En lugar de ello, es mejor ceñirse a la práctica recomendada de utilizar un usuario de base de datos creado con los privilegios mínimos necesarios para su aplicación.

Temas

- Autenticación de contraseña
- Autenticación de bases de datos de IAM
- Autenticación Kerberos

Autenticación de contraseña

Con la autenticación de contraseña, la base de datos realiza toda la administración de cuentas de usuario. Puede crear usuarios con instrucciones SQL como `CREATE USER`, con la cláusula

adecuada que requiere el motor de base de datos para especificar contraseñas. Por ejemplo, en MySQL la instrucción es `CREATE USER nombre IDENTIFIED BY contraseña`, mientras que, en PostgreSQL, la instrucción es `CREATE USER nombre WITH PASSWORD contraseña`.

Con la autenticación por contraseña, la base de datos controla y autentica las cuentas de usuario. Si un motor de base de datos tiene características sólidas de administración de contraseñas, puede mejorar la seguridad. La autenticación de bases de datos puede ser más fácil de administrar mediante la autenticación de contraseña cuando tiene comunidades de usuarios pequeñas. Debido a que en este caso se generan contraseñas de texto sin formato, la integración con AWS Secrets Manager puede mejorar la seguridad.

Para obtener información sobre el uso de Secrets Manager con Amazon Aurora, consulte Creación de un secreto básico y Rotación de secretos para bases de datos de Amazon RDS admitidas en la Guía del usuario de AWS Secrets Manager. Si quiere obtener información para recuperar los secretos mediante programación en las aplicaciones personalizadas, consulte Recuperar el valor secreto en la guía del usuario de AWS Secrets Manager.

Autenticación de bases de datos de IAM

Puede autenticar en el clúster de bases de datos mediante la autenticación de base de datos de AWS Identity and Access Management (IAM). Con este método de autenticación, no es necesario usar una contraseña al conectarse a un clúster de bases de datos. En su lugar, puede usar un token de autenticación.

Para obtener más información acerca de la autenticación de bases de datos de IAM, incluida información sobre la disponibilidad de motores de base de datos específicos, consulte Autenticación de bases de datos de IAM .

Autenticación Kerberos

Amazon Aurora admite la autenticación externa de usuarios de bases de datos que usan Kerberos y Microsoft Active Directory. Kerberos es un protocolo de autenticación de red que usa tickets y criptografía de clave simétrica para eliminar la necesidad de transmitir contraseñas a través de la red. Kerberos ha sido creado en Active Directory y está diseñado para autenticar usuarios para recursos de redes, como bases de datos.

La compatibilidad de Amazon Aurora con Kerberos y Active Directory ofrece beneficios de inicio de sesión único y autenticación centralizada de usuarios de bases de datos. Puede mantener sus

credenciales de usuario en Active Directory. Active Directory ofrece un lugar centralizado para almacenar y administrar credenciales para varios clústeres de bases de datos.

Para usar las credenciales del Active Directory autoadministrado, debe configurar una relación de confianza con Directory Service para el Microsoft Active Directory al que se haya unido el clúster de base de datos.

Aurora PostgreSQL y Aurora MySQL admiten relaciones de confianza unidireccionales y bidireccionales entre bosques con autenticación selectiva o en todo el bosque.

En algunos casos, puede configurar la autenticación Kerberos a través de una relación de confianza externa. Esto requiere que el directorio de Active Directory autoadministrado tenga una configuración adicional. Esto incluye, entre otras cosas, Kerberos Forest Search Order.

Aurora admite la autenticación Kerberos para clústeres de base de datos de Aurora MySQL y Aurora PostgreSQL. Para obtener más información, consulte Uso de la autenticación Kerberos para Aurora MySQL y Uso de la autenticación Kerberos con Aurora PostgreSQL.

Administración de contraseñas con Amazon Aurora y AWS Secrets Manager

Amazon Aurora se integra con Secrets Manager para administrar las contraseñas de usuario maestras de sus .

Temas

- Disponibilidad en regiones y versiones
- Limitaciones de la integración de Secrets Manager con Amazon Aurora
- Descripción general de la administración de contraseñas de usuarios maestros con AWS Secrets Manager
- Ventajas de administrar las contraseñas de usuarios maestros con Secrets Manager
- Permisos necesarios para la integración de Secrets Manager
- Cumplimiento de la administración de la contraseña del usuario maestro por parte de Aurora en AWS Secrets Manager
- Administración de la contraseña de usuario maestra para un clúster de base de datos con Secrets Manager
- Rotación del secreto de contraseña de usuario maestra para un clúster de base de datos
- Visualización de los detalles de un secreto para un clúster de base de datos

Disponibilidad en regiones y versiones

La disponibilidad y el soporte de las características varía según las versiones específicas de cada motor de base de datos y entre Regiones de AWS. Para obtener más información sobre la disponibilidad de versiones y regiones con la integración de Secrets Manager con Amazon Aurora, consulte Regiones y motores de bases de datos Aurora admitidos para la integración de Secrets Manager.

Limitaciones de la integración de Secrets Manager con Amazon Aurora

Las siguientes funciones no admiten la administración de contraseñas de usuario maestro con Secrets Manager:

- Implementaciones azules/verdes de Amazon RDS
- Clústeres de base de datos que forman parte de una base de datos de Aurora global

- Aurora Serverless v1Clústeres de base de datos de
- Réplicas de lectura entre regiones
- Replicación externa de registros binarios

Descripción general de la administración de contraseñas de usuarios maestros con AWS Secrets Manager

Con AWS Secrets Manager, puede reemplazar las credenciales con codificación rígida (incluidas las contraseñas de bases de datos), con una llamada a la API de Secrets Manager para recuperar el secreto mediante programación. Para obtener más información acerca de Secrets Manager, consulte la Guía del usuario de AWS Secrets Manager.

Cuando guarda secretos de base de datos en Secrets Manager, su Cuenta de AWS incurre en cargos. Para obtener más información acerca de los precios, consulte Precios de AWS Secrets Manager.

Puede especificar que Aurora administre la contraseña de usuario principal en Secrets Manager para un clúster de base de datos de Amazon Aurora al realizar una de las siguientes operaciones:

- Creación de un clúster de base de datos
- Modificación de un clúster de base de datos
- Restauración de un clúster de base de datos desde Amazon S3 (solo Aurora MySQL)

Al especificar que Aurora administre la contraseña de usuario principal en Secrets Manager, Aurora genera la contraseña y la almacena en Secrets Manager. Puede interactuar directamente con el secreto para recuperar las credenciales del usuario maestro. También puede especificar una clave gestionada por el cliente para cifrar el secreto o utilizar la clave de KMS que proporciona Secrets Manager.

Aurora administra la configuración del secreto y, de forma predeterminada, lo rota cada siete días. Puede modificar algunos de los ajustes, como el programa de rotación. Si elimina un clúster de base de datos que administra un secreto en Secrets Manager, también se eliminarán el secreto y los metadatos asociados.

Para conectarse a con las credenciales en un secreto, puede recuperar el secreto en Secrets Manager. Para obtener más información, consulte Recuperar secretos de AWS Secrets Manager y

Conexión a una base de datos SQL con credenciales en un secreto de AWS Secrets Manager en la Guía del usuario de AWS Secrets Manager.

Ventajas de administrar las contraseñas de usuarios maestros con Secrets Manager

La administración de las contraseñas de usuarios maestros de Aurora con Secrets Manager ofrece las siguientes ventajas:

- Aurora genera automáticamente las credenciales de la base de datos.
- Aurora almacena y administra automáticamente las credenciales de la base de datos en AWS Secrets Manager.
- Aurora rota las credenciales de la base de datos con regularidad, sin necesidad de realizar cambios en la aplicación.
- Secrets Manager protege las credenciales de la base de datos del acceso humano y de la visualización en texto plano.
- Secrets Manager permite recuperar las credenciales de la base de datos en secretos para las conexiones a bases de datos.
- Secrets Manager permite un control detallado del acceso a las credenciales de la base de datos en secretos mediante IAM.
- Si lo desea, puede separar el cifrado de bases de datos del cifrado de credenciales con diferentes claves de KMS.
- Puede eliminar la administración manual y la rotación de las credenciales de la base de datos.
- Puede monitorear fácilmente las credenciales de la base de datos con AWS CloudTrail y Amazon CloudWatch.

Para obtener más información acerca de los beneficios de Secrets Manager, consulte la Guía del usuario de AWS Secrets Manager.

Permisos necesarios para la integración de Secrets Manager

Los usuarios deben tener los permisos necesarios para realizar las operaciones relacionadas con la integración de Secrets Manager. Puede crear políticas de IAM que concedan permisos para realizar operaciones de la API concretas en los recursos especificados que necesiten. A continuación, puede asociar esas políticas a los roles o conjuntos de permisos de IAM que necesiten esos permisos. Para obtener más información, consulte Administración de la identidad y el acceso en Amazon Aurora.

Para las operaciones de creación, modificación o restauración, el usuario que especifique que Aurora administra la contraseña de usuario maestro en Secrets Manager debe tener permisos para realizar las siguientes operaciones:

- `kms:DescribeKey`
- `secretsmanager:CreateSecret`
- `secretsmanager:TagResource`

El permiso `kms:DescribeKey` es necesario para acceder a la clave administrada por el cliente para `MasterUserSecretKmsKeyId` y para describir `aws/secretsmanager`.

Para las operaciones de creación, modificación o restauración, el usuario que especifique la contraseña de usuario maestro en Secrets Manager debe tener permisos para realizar las siguientes operaciones:

- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:CreateGrant`

Para las operaciones de modificación, el usuario que rote la contraseña de usuario maestro en Secrets Manager debe tener permisos para realizar la siguiente operación:

- `secretsmanager:RotateSecret`

Cumplimiento de la administración de la contraseña del usuario maestro por parte de Aurora en AWS Secrets Manager

Puede utilizar las claves de condición de IAM para hacer que Aurora administre la contraseña del usuario maestro en AWS Secrets Manager. La siguiente política no permite a los usuarios crear ni restaurar instancias de base de datos o clústeres de bases de datos a menos que Aurora administre la contraseña del usuario principal en Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

> **Note**
>
> Esta política aplica la administración de contraseñas en AWS Secrets Manager cuando se crea. Sin embargo, sigue pudiendo deshabilitar la integración de Secrets Manager y establecer manualmente una contraseña maestra si modifica el clúster.
> Para evitarlo, incluya `rds:ModifyDBInstance`, `rds:ModifyDBCluster` en el bloque de acciones de la política. Tenga en cuenta que esto impide que el usuario aplique más modificaciones a los clústeres existentes que no tengan habilitada la integración de Secrets Manager.

Para obtener más información sobre el uso de las claves de condición en las políticas de IAM, consulte Claves de condición de políticas para Aurora y Políticas de ejemplo: uso de claves de condición.

Administración de la contraseña de usuario maestra para un clúster de base de datos con Secrets Manager

Puede configurar la administración de Aurora de la contraseña del usuario maestro en Secrets Manager cuando realice las siguientes acciones:

- Creación de un clúster de base de datos de Amazon Aurora
- Modificación de un clúster de base de datos de Amazon Aurora

- Migración de datos desde una base de datos MySQL externa a un clúster de base de datos de Amazon Aurora MySQL

Puede utilizar la consola de RDS, la AWS CLI o la API de RSD para realizar estas acciones.

Consola

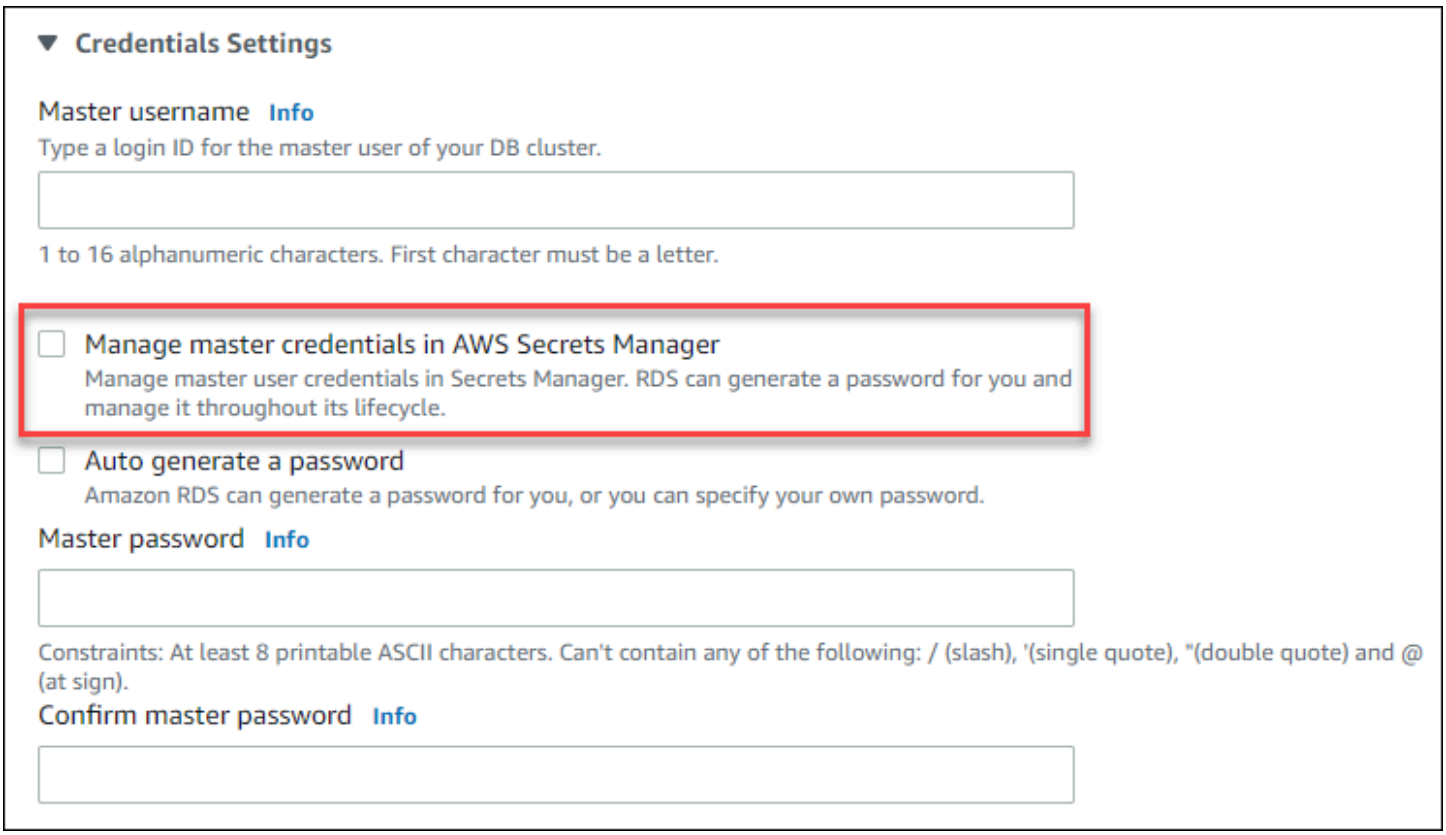
Siga las instrucciones para crear o modificar un clúster de base de datos con la consola de RDS:

- Creación de un clúster de base de datos
- Modificación de una instancia de base de datos en un clúster de base de datos

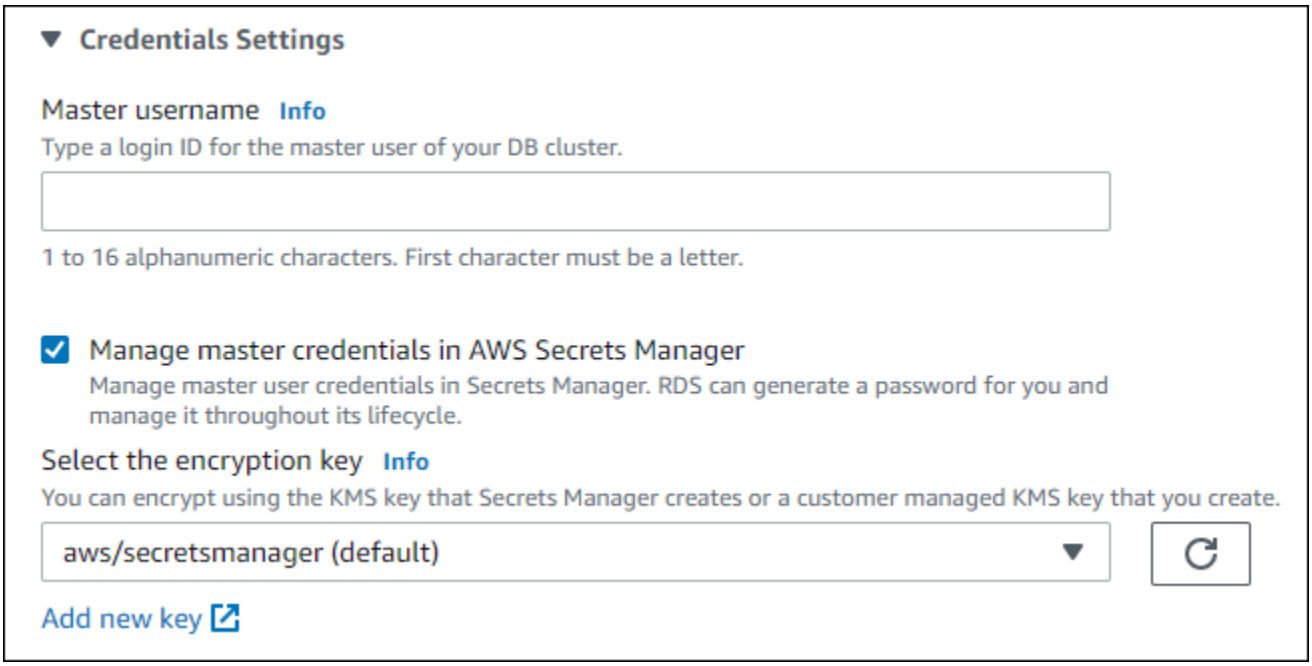
  En la consola de RDS, puede modificar cualquier instancia de base de datos para especificar la configuración de administración de contraseñas del usuario maestro para todo el clúster de base de datos.
- Restauración de un clúster de base de datos de Amazon Aurora MySQL desde un bucket de Amazon S3

Al utilizar la consola de RDS para realizar una de estas operaciones, puede especificar que Aurora administre la contraseña del usuario maestro en Secrets Manager. Para hacerlo al crear o restaurar un clúster de base de datos, seleccione Administrar las credenciales principales en AWS Secrets Manager en Configuración de credenciales. Cuando modifique un clúster de base de datos, seleccione Administrar las credenciales principales en AWS Secrets Manager en Configuración.

La siguiente imagen es un ejemplo de la configuración Administrar las credenciales principales en AWS Secrets Manager que se utiliza al crear o restaurar un clúster de base de datos.



Al seleccionar esta opción, Aurora genera la contraseña de usuario maestra y la administra durante todo su ciclo de vida en Secrets Manager.



Puede optar por cifrar el secreto con una clave de KMS que proporcione Secrets Manager o con la clave gestionada por el cliente que cree usted. Una vez que Aurora administre las credenciales de

la base de datos de un clúster de base de datos, no podrá cambiar la clave de KMS que se usa para cifrar el secreto.

Puede elegir otros ajustes para cumplir con sus requisitos.

Para obtener más información sobre la configuración disponible al crear un clúster de base de datos, consulte Configuración de clústeres de bases de datos de Aurora. Para obtener más información sobre la configuración disponible al modificar un clúster de base de datos, consulte Configuración para Amazon Aurora.

AWS CLI

Para administrar la contraseña del usuario maestro con Aurora en Secrets Manager, especifique la opción `--manage-master-user-password` en uno de los siguientes comandos:

- create-db-cluster
- modify-db-cluster
- restore-db-cluster-from-s3

Al especificar la opción `--manage-master-user-password` en estos comandos, Aurora genera la contraseña de usuario maestro y la administra durante todo su ciclo de vida en Secrets Manager.

Para cifrar el secreto, también puede especificar una clave gestionada por el cliente o utilizar la clave de KMS que proporciona Secrets Manager. Use la opción `--master-user-secret-kms-key-id` para especificar una clave administrada por el cliente. El identificador de la clave de AWS KMS es el ARN de la clave, el identificador de clave, el ARN de alias o el nombre de alias de la clave de KMS. Para especificar una clave en una Cuenta de AWS diferente, debe utilizar la clave de ARN o el alias de ARN. Una vez que Aurora administre las credenciales de la base de datos de un clúster de base de datos, no podrá cambiar la clave de KMS que se usa para cifrar el secreto.

Puede elegir otros ajustes para cumplir con sus requisitos.

Para obtener más información sobre la configuración disponible al crear un clúster de base de datos, consulte Configuración de clústeres de bases de datos de Aurora. Para obtener más información sobre la configuración disponible al modificar un clúster de base de datos, consulte Configuración para Amazon Aurora.

En este ejemplo se crea un clúster de base de datos y se especifica que Aurora administre la contraseña en Secrets Manager. El secreto se cifra mediante la clave de KMS que proporciona Secrets Manager.

Example

Para Linux, macOS o Unix:

```
aws rds create-db-cluster \
     --db-cluster-identifier sample-cluster \
     --engine aurora-mysql \
     --engine-version 8.0 \
     --master-username admin \
     --manage-master-user-password
```

Para Windows:

```
aws rds create-db-cluster ^
     --db-cluster-identifier sample-cluster ^
     --engine aurora-mysql ^
     --engine-version 8.0 ^
     --master-username admin ^
     --manage-master-user-password
```

API de RDS

Para especificar que Aurora administre la contraseña del usuario maestro en Secrets Manager, defina el parámetro `ManageMasterUserPassword` en `true` con alguna de las siguientes operaciones:

- CreateDBCluster
- ModifyDBCluster
- RestoreDBClusterFromS3

Al seleccionar el parámetro `ManageMasterUserPassword` en `true` en una de estas operaciones, Aurora genera la contraseña de usuario maestro y la administra durante todo su ciclo de vida en Secrets Manager.

Para cifrar el secreto, también puede especificar una clave gestionada por el cliente o utilizar la clave de KMS que proporciona Secrets Manager. Utilice el parámetro `MasterUserSecretKmsKeyId` para especificar una clave administrada por el cliente. El identificador de la clave de AWS KMS es el ARN de la clave, el identificador de clave, el ARN de alias o el nombre de alias de la clave de KMS. Para especificar una clave en una Cuenta de AWS diferente, debe utilizar la clave de ARN o el alias

de ARN. Una vez que Aurora administre las credenciales de la base de datos de un clúster de base de datos, no podrá cambiar la clave de KMS que se usa para cifrar el secreto.

Rotación del secreto de contraseña de usuario maestra para un clúster de base de datos

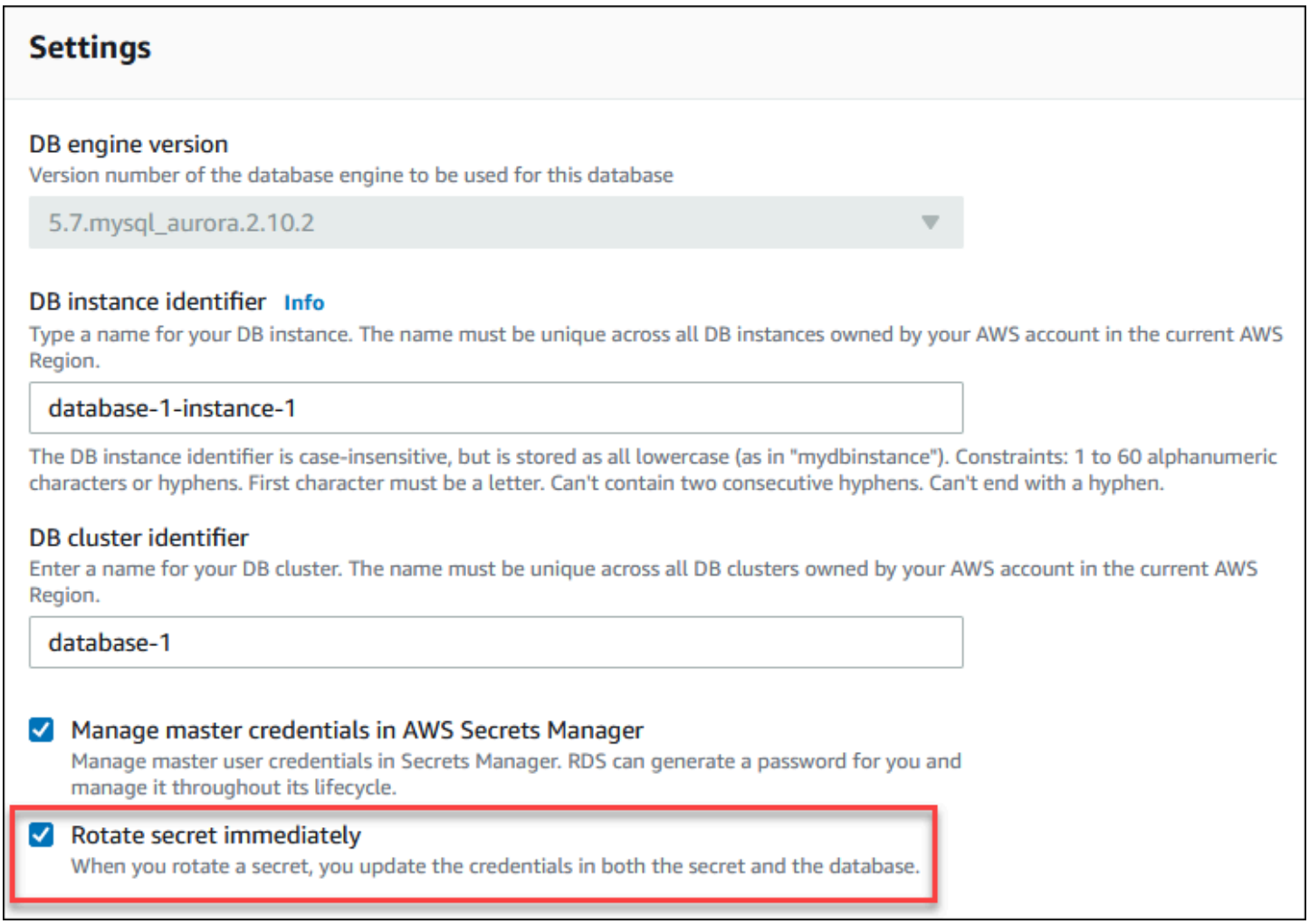
Cuando Aurora rota un secreto de contraseña de usuario maestro, Secrets Manager genera una nueva versión de secreto para el secreto existente. La nueva versión del secreto contiene la nueva contraseña de usuario maestra. Aurora cambia la contraseña de usuario maestro para el clúster de base de datos para que coincida con la contraseña de la nueva versión de secreto.

Puede rotar un secreto inmediatamente en lugar de esperar a que se programe una rotación. Para rotar un secreto de contraseña de usuario maestra en Secrets Manager, modifique el clúster de base de datos . Para obtener más información sobre la modificación de un clúster de bases de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Puede cambiar un secreto de contraseña de usuario maestro inmediatamente con la consola de RDS, la AWS CLI de RDS o la API de RDS. La nueva contraseña siempre tiene 28 caracteres y contiene al menos una mayúscula y una minúscula, un número y un signo de puntuación.

Consola

Para rotar un secreto de contraseña de usuario maestro mediante la consola de RDS, modifique el clúster de base de datos y seleccione Rotate secret immediately (Rotar el secreto inmediatamente) en Settings (Configuración).



Siga las instrucciones para modificar un clúster de base de datos con la consola RDS en Modificación del clúster de base de datos con la consola, CLI y API. Debe elegir Apply immediately (Aplicar inmediatamente) en la página de confirmación.

AWS CLI

Para rotar un secreto de contraseña de usuario maestro mediante la AWS CLI, utilice el comando modify-db-cluster y especifique la opción `--rotate-master-user-password`. Debe especificar la opción `--apply-immediately` al rotar la contraseña maestra.

En este ejemplo, se rota un secreto de contraseña de usuario maestro.

Example

Para Linux, macOS o Unix:

```
aws rds modify-db-cluster \
```

```
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Para Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

API de RDS

Puede rotar un secreto de contraseña de usuario maestro mediante la operación ModifyDBCluster y estableciendo el parámetro `RotateMasterUserPassword` en `true`. Debe establecer el parámetro `ApplyImmediately` en `true` al rotar la contraseña maestra.

Visualización de los detalles de un secreto para un clúster de base de datos

Puede recuperar sus secretos mediante la consola (https://console.aws.amazon.com/secretsmanager/) o la AWS CLI (comando get-secret-value de Secrets Manager).

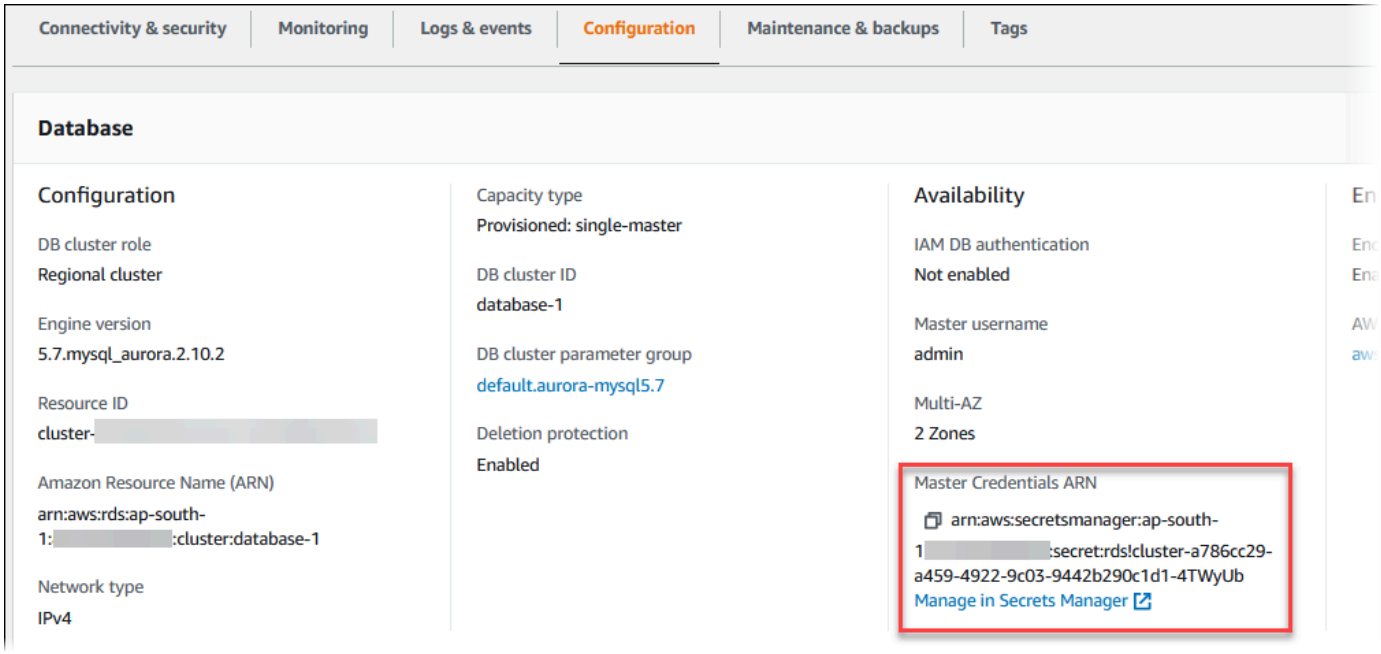
Puede encontrar el nombre de recurso de Amazon (ARN) de un secreto administrado por Aurora en Secrets Manager con la consola de RDS, la AWS CLI de RDS o la API de RDS.

Consola

Para ver los detalles de un secreto administrado por Aurora en Secrets Manager

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el nombre del clúster de base de datos para mostrar sus detalles.
4. Elija la pestaña Configuración.

   En el Master Credentials ARN (ARN de credenciales maestras), puede ver el ARN secreto.



Puede seguir el enlace Manage in Secrets Manager (Administrar en Secrets Manager) para ver y administrar el secreto en la consola de Secrets Manager.

AWS CLI

Puede utilizar el comando describe-db-clusters de la AWS CLI de RDS para buscar la siguiente información sobre un secreto administrado por Aurora en Secrets Manager:

- `SecretArn`: ARN del secreto
- `SecretStatus`: estado del secreto

  Otros valores de estado posibles son:

  - `creating`: se está creando el secreto.
  - `active`: el secreto está disponible para su uso y rotación normales.
  - `rotating`: se está rotando el secreto.
  - `impaired`: el secreto se puede usar para acceder a las credenciales de la base de datos, pero no se puede rotar. Un secreto puede tener este estado si, por ejemplo, se cambian los permisos para que RDS ya no pueda acceder al secreto ni a la clave de KMS del secreto.

    Cuando un secreto tiene este estado, puede corregir la condición que provocó el estado. Si corrige la condición que causó el estado, el estado sigue siendo `impaired` hasta la siguiente rotación. De forma alternativa, puede modificar el clúster de base de datos para desactivar

la administración automática de las credenciales de la base de datos y, a continuación, volver a modificar el clúster de base de datos para activar la administración automática de las credenciales de la base de datos. Para modificar un clúster de base de datos, use la opción `--manage-master-user-password` en el comando modify-db-cluster.

- `KmsKeyId`: ARN de la clave de KMS que se utiliza para cifrar el secreto.

Especifique la opción `--db-cluster-identifier` para mostrar el resultado de un clúster de base de datos específica. En este ejemplo se muestra el resultado de un secreto que utiliza un clúster de base de datos.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

A continuación, se muestra un ejemplo de resultado de un secreto:

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

Cuando tenga el ARN secreto, podrá ver los detalles del secreto con el comando get-secret-value de la CLI de Secrets Manager.

En este ejemplo se muestran los detalles del secreto del resultado del ejemplo anterior.

Example

Para Linux, macOS o Unix:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Para Windows:

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

API de RDS

Puede ver el ARN, el estado y la clave de KMS de un secreto administrado por Aurora en Secrets Manager mediante la operación RDS DescribeDBClusters y estableciendo el parámetro `DBClusterIdentifier` en un identificador de clúster de base de datos. En el resultado se incluyen detalles sobre el secreto.

Cuando tenga el ARN secreto, podrá ver los detalles del secreto con la operación GetSecretValue de Secrets Manager.

Protección de datos en Amazon RDS

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en Amazon Relational Database Service. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulta las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulta la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS.

Con fines de protección de datos, recomendamos proteger las credenciales de la Cuenta de AWS y configurar cuentas de usuario individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utiliza SSL/TLS para comunicarse con los recursos de AWS. Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure los registros de API y de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre cómo utilizar registros de seguimiento de CloudTrail para capturar actividades de AWS, consulta Working with CloudTrail trails en la Guía del usuario de AWS CloudTrail.

- Utiliza las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-3 al acceder a AWS a través de una interfaz de línea de comandos o una API, utiliza un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulta Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja con Amazon RDS u otros Servicios de AWS mediante la consola, la API, AWS CLI o los SDK de AWS. Cualquier dato que ingrese en etiquetas o campos de texto de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Temas

- Protección de datos mediante cifrado
- Privacidad del tráfico entre redes

Protección de datos mediante cifrado

Puede habilitar el cifrado para recursos de bases de datos. También puede cifrar conexiones a clústeres de bases de datos.

Temas

- Cifrado de recursos de Amazon Aurora
- Administración de AWS KMS key
- Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos
- Rotar certificados SSL/TLS

Cifrado de recursos de Amazon Aurora

Amazon Aurora puede cifrar sus Amazon Aurora clústeres de base de datos. Los datos cifrados en reposo incluyen el almacenamiento subyacente de clústeres de bases de datos, sus copias de seguridad automatizadas, sus réplicas de lectura y sus instantáneas.

En los clústeres de bases de datos de Amazon Aurora con cifrado se utiliza el algoritmo de cifrado AES-256 estándar del sector para cifrar los datos en el servidor que aloja clústeres de bases de datos de Amazon Aurora. Una vez cifrados los datos, Amazon Aurora se encarga de la autenticación de acceso y del descifrado de los datos de forma transparente, con un impacto mínimo en el desempeño. No es necesario modificar las aplicaciones cliente de base de datos para utilizar el cifrado.

> **Note**
>
> Para los clústeres de de base de datos cifradas y no cifradas, los datos en tránsito entre el origen y las réplicas de lectura están cifrados, incluso al replicar entre regiones de AWS.

Temas

- Información general del cifrado de los recursos de Amazon Aurora
- Cifrar un clúster de bases de datos de Amazon Aurora
- Determinar si el cifrado está activado para un clúster de bases de datos
- Disponibilidad del cifrado de Amazon Aurora
- Cifrado en tránsito
- Limitaciones de los clústeres de base de datos cifrados de Amazon Aurora

Información general del cifrado de los recursos de Amazon Aurora

Los clústeres de bases de datos cifrados de Amazon Aurora proporcionan una capa adicional de protección de datos al proteger los datos del acceso no autorizado al almacenamiento subyacente. Puede utilizar el cifrado de Amazon Aurora para aumentar la protección de datos de las aplicaciones implementadas en la nube y para cumplir con los requisitos de conformidad para el cifrado en reposo. Para un clúster de bases de datos cifrado de Amazon Aurora, todas las instancias de base de datos, los registros, las copias de seguridad y las instantáneas están cifrados. Para obtener más información sobre la disponibilidad y las limitaciones del cifrado, consulte Disponibilidad del cifrado de Amazon Aurora y Limitaciones de los clústeres de base de datos cifrados de Amazon Aurora.

Amazon Aurora usa una clave de AWS Key Management Service para cifrar estos recursos. AWS KMS combina hardware y software seguros y altamente disponibles para ofrecer un sistema de administración de claves escalado para la nube. Puede utilizar una Clave administrada de AWS, o bien puede o crear claves administradas por el cliente.

Cuando crea un clúster de bases de datos cifrado, puede elegir una clave administrada por el cliente o la Clave administrada de AWS para Amazon Aurora para cifrar el clúster de bases de datos. Si no especifica el identificador de clave para una clave administrada por el cliente, Amazon Aurora utiliza la Clave administrada de AWS para el nuevo clúster de bases de datos. Amazon Aurora crea una Clave administrada de AWS para Amazon Aurora para su cuenta de AWS. Su cuenta de AWS tiene una Clave administrada de AWS diferente para Amazon Aurora para cada región de AWS.

Para administrar las claves administradas por el cliente que se usan para cifrar y descifrar los recursos de Amazon Aurora, se utiliza AWS Key Management Service (AWS KMS).

Si utiliza AWS KMS, podrá crear claves administradas por el cliente y definir las políticas que controlan cómo se pueden utilizar las claves administradas por el cliente. AWS KMS es compatible con CloudTrail, lo que permite auditar la utilización de claves de KMS para comprobar que las claves administradas por el cliente se utilizan de forma adecuada. Puede utilizar las claves administradas por el cliente con Amazon Aurora y los servicios de AWS admitidos, como, por ejemplo, Amazon S3, Amazon EBS y Amazon Redshift. Para ver una lista de los servicios integrados con AWS KMS, consulte Integración con los servicios de AWS. Algunas consideraciones sobre el uso de las claves de KMS:

- Una vez que se crea una instancia de base de datos cifrada, no se puede cambiar la clave de KMS que dicha instancia de base de datos utiliza. Por tanto, asegúrese de determinar los requisitos de su clave de KMS antes de crear la instancia de base de datos cifrada.

  Si debe cambiar la clave de cifrado del clúster de base de datos, cree una instantánea manual del clúster y habilite el cifrado mientras se copia la instantánea. Para obtener más información, consulte artículo de información de re:Post.
- Si copia una instantánea cifrada, puede utilizar una clave de KMS para cifrar la instantánea de destino diferente de la que se utilizó para cifrar la instantánea de origen.
- No se puede compartir una instantánea que se haya cifrado con la Clave administrada de AWS de la cuenta de AWS que compartió la instantánea.
- Cada instancia de base de datos del clúster de bases de datos se cifra utilizando la misma clave de KMS que el clúster de bases de datos.
- También puede cifrar una réplica de lectura de un clúster cifrado de Amazon Aurora.

Important

Amazon Aurora puede perder el acceso a la clave KMS para un clúster de base de datos al deshabilitar la clave KMS. En estos casos, el clúster de bases de datos cifrado entra en estado `inaccessible-encryption-credentials-recoverable`. El clúster de base de datos permanece en este estado durante siete días, durante los cuales la instancia se detiene. Es posible que las llamadas a la API realizadas al clúster de base de datos durante este tiempo no se realicen correctamente. Para recuperar el clúster de base de datos, habilite la clave KMS y reinicie este clúster de base de datos. Habilite la clave de KMS desde la Consola de administración de AWS, la AWS CLI o la API de RDS. Reinicie el clúster de base de datos con el comando de la AWS CLI start-db-cluster o la Consola de administración de AWS.

El estado `inaccessible-encryption-credentials-recoverable` solo se aplica a clústeres de base de datos que pueden detenerse. Este estado recuperable no se aplica a las instancias que no se pueden detener, como los clústeres con réplicas de lectura entre regiones. Para obtener más información, consulte the section called “Limitaciones”.

Si el clúster de bases de datos no se recupera en sitie días, pasa al estado de terminal `inaccessible-encryption-credentials`. En este estado, el clúster de base de datos ya no se puede usar y solo puede restaurarlo desde una copia de seguridad. Recomendamos que siempre habilite las copias de seguridad para los clústeres de bases de datos cifrados con el fin de protegerse contra la pérdida de los datos cifrados de dichas bases de datos.

Durante la creación de un clúster de base de datos, Aurora comprueba si la entidad principal que realiza la llamada tiene acceso a la clave KMS y genera una concesión a partir de la clave KMS que utiliza durante toda la vida útil del clúster de base de datos. La revocación del acceso de la entidad principal que realiza la llamada a la clave KMS no afecta a una base de datos en ejecución. Cuando se utilizan claves KMS en situaciones de varias cuentas, como copiar una instantánea a otra cuenta, la clave KMS debe compartirse con la otra cuenta.

Si crea un clúster de base de datos a partir de la instantánea sin especificar una clave de KMS diferente, el nuevo clúster utiliza la clave de KMS de la cuenta de origen. La revocación del acceso a la clave después de crear el clúster de base de datos no afecta al clúster. Sin embargo, la desactivación de la clave afecta a todos los clústeres de base de datos cifrados con esa clave. Para evitarlo, especifique una clave diferente durante la operación de copia de la instantánea.

Para obtener más información acerca de claves de KMS, consulte AWS KMS keys en la Guía para desarrolladores de AWS Key Management Service y Administración de AWS KMS key.

Cifrar un clúster de bases de datos de Amazon Aurora

Para cifrar un clúster de bases de datos nuevo, elija Habilitar cifrado en la consola. Para obtener más información acerca de la creación de un clúster de bases de datos, consulte Creación de un clúster de base de datos de Amazon Aurora.

Si utiliza el comando create-db-cluster de la AWS CLI para crear un clúster de bases de datos cifrado, establezca el parámetro `--storage-encrypted`. Si utiliza la operación CreateDBCluster de la API, establezca el parámetro `StorageEncrypted` en true.

Una vez que se crea un clúster de bases de datos cifrado, no se puede cambiar la clave de KMS que dicho clúster de bases de datos utiliza. Por tanto, no olvide especificar los requisitos de la clave de KMS antes de crear el clúster de bases de datos cifrado.

Si utiliza el comando AWS CLI de la `create-db-cluster` para crear un clúster de bases de datos cifrado con una clave administrada por el cliente, establezca el parámetro `--kms-key-id` en cualquier identificador de clave para la clave de KMS. Si utiliza la operación `CreateDBInstance` de la API de Amazon RDS, establezca el parámetro `KmsKeyId` en cualquier identificador de clave para la clave de KMS. Para utilizar una clave administrada por el cliente en una cuenta de AWS diferente, especifique el ARN de la clave o el ARN del alias.

Determinar si el cifrado está activado para un clúster de bases de datos

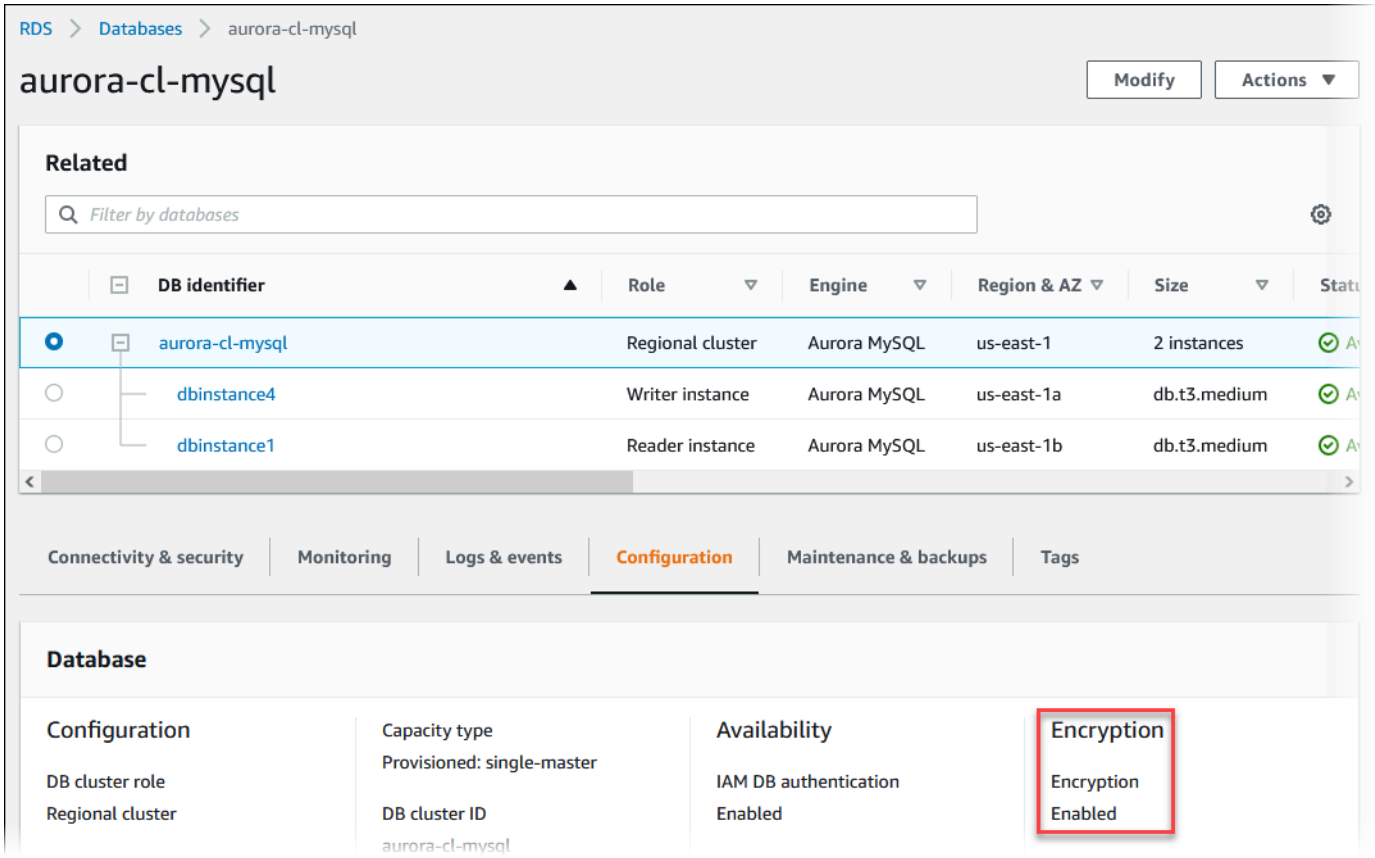
Puede utilizar Consola de administración de AWS, AWS CLI o la API de RDS para determinar si el cifrado en reposo está activado para un clúster de bases de datos.

Consola

Para determinar si el cifrado en reposo está activado para un clúster de bases de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos).
3. Elija el nombre del clúster de bases de datos que desea verificar para ver los detalles.
4. Elija la pestaña Configuration y verifique el valor Cifrado.

   Muestra Enabled (Habilitado) o Not enabled (No habilitado).



AWS CLI

Para determinar si el cifrado en reposo está activado para un clúster de bases de datos mediante AWS CLI, llame al comando describe-db-clusters con la siguiente opción:

- `--db-cluster-identifier`: el nombre del clúster de bases de datos.

En el siguiente ejemplo se utiliza una consulta para devolver ya sea `TRUE` o `FALSE` en relación con el cifrado en reposo del clúster de bases de datos `mydb`.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

API de RDS

Para determinar si el cifrado en reposo está activado para un clúster de bases de datos mediante la API de Amazon RDS, llame a la operación DescribeDBClusters con el siguiente parámetro:

- `DBClusterIdentifier`: el nombre del clúster de bases de datos.

Disponibilidad del cifrado de Amazon Aurora

Actualmente, el cifrado de Amazon Aurora; están disponibles para todos los motores de bases de datos y tipos de almacenamiento.

> **Note**
>
> El cifrado de Amazon Aurora no está disponible para la clase de instancia de base de datos sdb.t2.micro.

Cifrado en tránsito

Cifrado en la capa física

Todos los datos que fluyen en las Regiones de AWS a través de la red global de AWS se cifran automáticamente en la capa física antes de salir de las instalaciones seguras de AWS. Todo el tráfico entre las zonas de disponibiliad está cifrado. Las capas adicionales de cifrado, incluidas las que aparecen en esta sección, pueden proporcionar una protección adicional.

Cifrado proporcionado por el emparejamiento de Amazon VPC y el emparejamiento entre regiones de puerta de enlace de tránsito

Todo el tráfico entre regiones que utiliza la interconexión de Amazon VPC y puerta de enlace de tránsito se cifra de forma masiva automáticamente cuando sale de una región. De manera automática, se proporciona una capa adicional de cifrado en la capa física para todo el tráfico antes de dejar las instalaciones seguras de AWS.

Cifrado entre instancias

AWS proporciona conectividad privada y segura entre instancias de bases de datos de todo tipo. Además, en algunos tipos de instancia, se utilizan las capacidades de descarga del hardware Nitro System subyacente para cifrar de manera automática el tráfico en tránsito entre instancias. Este cifrado utiliza algoritmos de encriptación autenticada con datos asociados (AEAD), con cifrado de 256 bits. No hay impacto en el rendimiento de la red. Para admitir este cifrado adicional del tráfico en tránsito entre instancias, se deben cumplir los siguientes requisitos:

- Las instancias utilizan los siguientes tipos de instancias:

    - De uso general: M6i, M6id, M6in, M6idn, M7g
    - Optimizada para memoria: R6i, R6id, R6in, R6idn, R7g, X2idn, X2iedn, X2iezn
  - Las instancias se encuentran en la misma Región de AWS.
  - Las instancias están en la misma VPC o VPC interconectadas, y el tráfico no pasa a través de un dispositivo o servicio de red virtual, como un equilibrador de carga o una puerta de enlace de tránsito.

Limitaciones de los clústeres de base de datos cifrados de Amazon Aurora

Existen las siguientes limitaciones para los clústeres de Amazon Aurora con cifrado de bases de datos:

- No puede desactivar el cifrado en un clúster de bases de datos cifrado.
- No puede crear una instantánea cifrada de una clúster de bases de datos sin cifrar.
- Una instantánea de una clúster de bases de datos cifrado debe cifrarse utilizando la misma clave de KMS que la clúster de bases de datos.
- No puede convertir un clúster de bases de datos cifrado en uno sin cifrar. Sin embargo, sí es posible restaurar una instantánea sin cifrar en un clúster de bases de datos cifrado de Aurora. Para ello, especifique una clave de KMS cuando realice la restauración partiendo de una instantánea sin cifrar.
- No puede crear una réplica de Aurora cifrada a partir de un clúster de bases de datos de Aurora sin cifrar. No puede crear una réplica de Aurora sin cifrar a partir de un clúster de bases de datos de Aurora cifrado.
- Para copiar una instantánea cifrada de una región de AWS en otra, debe especificar la clave de KMS de la región de AWS de destino. Esto se debe a que las claves de KMS son específicas de la región de AWS en la que se crean.

  La instantánea de origen permanece cifrada durante todo el proceso de copia. Amazon Aurora utiliza el cifrado de sobre para proteger los datos durante el proceso de copia. Para obtener más información acerca del cifrado de sobre, consulte Cifrado de sobre en la guía para desarrolladores de AWS Key Management Service.
- No se puede descifrar una clúster de bases de datos cifrado. Sin embargo, puede exportar datos de una clúster de bases de datos cifrado e importar datos a una clúster de bases de datos sin cifrar.

Administración de AWS KMS key

Amazon Aurora se integra automáticamente con AWS Key Management Service (AWS KMS) para la administración de claves. Amazon Aurora utiliza el cifrado de sobre. Para obtener más información acerca del cifrado de sobre, consulte Cifrado de sobre en la Guía para desarrolladores de AWS Key Management Service.

Puede utilizar dos tipos de claves de AWS KMS para cifrar clústeres de bases de datos.

- Si desea tener un control total sobre una clave de KMS, debe crear una clave administrada por el cliente. Para obtener más información acerca de las claves administradas por el cliente, consulte Claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service.
- Las Claves administradas por AWS son las claves de KMS de la cuenta que se crean, administran y utilizan en su nombre por un servicio de AWS integrado con AWS KMS. De forma predeterminada, se utiliza el RDS Clave administrada de AWS (`aws/rds`) para el cifrado. No puede administrar, rotar ni eliminar el RDS Clave administrada de AWS. Para obtener más información acerca de Claves administradas por AWS, consulte Claves administradas por AWS en la Guía para desarrolladores de AWS Key Management Service.

Para administrar las claves KMS usadas para clústeres de bases de datos cifradas de Amazon Aurora, use AWS Key Management Service (AWS KMS) en la consola de AWS KMS, la AWS CLI o la API de AWS KMS. Para ver los registros de auditoría de cada acción realizada con una clave administrada por AWS o por el cliente, utilice AWS CloudTrail. Para obtener más información sobre la rotación de claves, consulte Rotación de claves de AWS KMS.

Autorización del uso de una clave administrada por el cliente

Cuando Aurora utiliza una clave administrada por el cliente en las operaciones criptográficas, actúa en nombre del usuario que está creando o modificando el recurso de Aurora.

Para crear un recurso de Aurora con una clave administrada por el cliente, un usuario debe tener permisos para llamar a las siguientes operaciones en la clave administrada por el cliente:

- `kms:CreateGrant`
- `kms:DescribeKey`

Puede especificar estos permisos necesarios en una política de claves o en una política de IAM si lo permite la política de claves.

> **Important**
>
> Cuando utiliza instrucciones de denegación explícitas para todos los recursos (*) en políticas de claves de AWS KMS con servicios administrados como Amazon RDS, debe especificar una condición para permitir que la cuenta propietaria del recurso acceda. Es posible que las operaciones produzcan un error sin esta condición, incluso si la regla de denegación incluye excepciones para el usuario de IAM.

> **Tip**
>
> Para seguir el principio de privilegios mínimos, no permita el acceso completo a `kms:CreateGrant`. En su lugar, use la clave de condición kms:ViaService para permitir al usuario crear concesiones en la clave de KMS solo cuando un servicio de AWS haya creado la concesión en nombre del usuario.

Puede hacer que la política de IAM sea más estricta de varias maneras. Por ejemplo, si desea permitir que la clave administrada por el cliente se utilice solo para solicitudes que se originen en Aurora, utilice la clave de condición kms:ViaService con el valor `rds.<region>.amazonaws.com`. Además, puede utilizar las claves o valores de Contexto de cifrado de Amazon RDS como condición para utilizar la clave administrada por el cliente para el cifrado.

Para obtener más información, consulte Permitir a los usuarios de otras cuentas utilizar una clave KMS en la Guía para desarrolladores de AWS Key Management Service y Políticas de claves en AWS KMS.

Contexto de cifrado de Amazon RDS

Cuando Aurora utiliza la clave KMS o cuando Amazon EBS utiliza la clave KMS en nombre de Aurora, el servicio especifica un contexto de cifrado. El contexto de cifrado es la información autenticada adicional (ADD) que AWS KMS usa para garantizar la integridad de los datos. Cuando se especifica un contexto de cifrado para una operación de cifrado, el servicio debe especificar el mismo contexto de cifrado para la operación de descifrado. De lo contrario, el descifrado produce un error. El contexto de cifrado también se escribe en los registros de AWS CloudTrail para ayudarle a

entender por qué se usó una determinada clave KMS. Sus registros de CloudTrail pueden contener numerosas entradas que describen el uso de una clave KMS, pero el contexto de cifrado de cada entrada de registro puede ayudarle a determinar el motivo de ese uso concreto.

Como mínimo, Aurora siempre utiliza el ID de clúster de base de datos para el contexto de cifrado, como en el siguiente ejemplo con formato JSON:

```
{ "aws:rds:dbc-id": "cluster-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

Este contexto de cifrado puede ayudarle a identificar el clúster de base de datos para el que se ha utilizado la clave KMS.

Cuando la clave de KMS se usa para un clúster de base de datos y un volumen de Amazon EBS específico, el ID de clúster de base de datos y el ID de volumen de Amazon EBS se usan para el contexto de cifrado, como en el siguiente ejemplo con formato JSON:

```
{
  "aws:rds:dbc-id": "cluster-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos

Puede utilizar SSL (Capa de conexión segura) o TLS (Transport Layer Security) desde una aplicación para cifrar una conexión a un clúster de bases de datos que ejecuta Aurora MySQL o Aurora PostgreSQL.

Las conexiones SSL/TLS proporcionan una capa de seguridad al cifrar los datos que circulan entre el cliente y el clúster de la base de datos. Si lo desea, su conexión SSL/TLS puede realizar una verificación de identidad del servidor validando el certificado del servidor instalado en su base de datos. Para solicitar la verificación de la identidad del servidor, siga este proceso general:

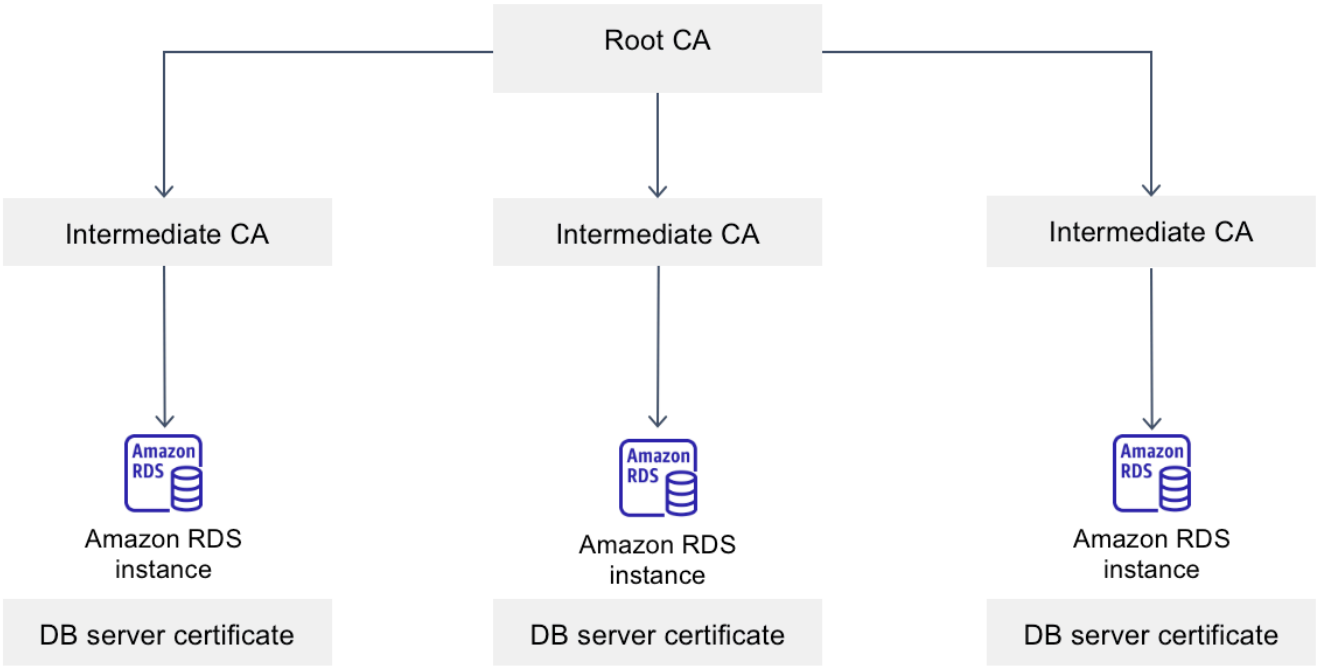
1. Elija la entidad de certificación (CA) que firma el certificado del servidor de base de datos, para su base de datos. Para obtener más información acerca de las entidades de certificación, consulte Entidades de certificación.
2. Descargue un paquete de certificados para usarlo cuando se conecte a la base de datos. Para descargar un paquete de certificados, consulte Agrupaciones de certificados por Región de AWS.

> **Note**
>
> Todos los certificados están disponibles solo para descarga con conexiones SSL/TLS.

3. Conéctese a la base de datos mediante el proceso del motor de base de datos para la implementación de las conexiones SSL/TLS. Cada motor base de datos tiene su propio proceso para implementar SSL/TLS. Para obtener información sobre cómo implementar SSL/TLS para su base de datos, siga el enlace que corresponda a su motor de base de datos:
   - Seguridad con Amazon Aurora MySQL
   - Seguridad con Amazon Aurora PostgreSQL

Entidades de certificación

La entidad de certificación (CA) es el certificado que identifica la CA raíz en la parte superior de la cadena de certificados. La CA firma el certificado del servidor de base de datos, que está instalado en cada instancia de base de datos. El certificado del servidor de base de datos identifica la instancia de base de datos como un servidor de confianza.



Amazon RDS proporciona las siguientes CA para firmar el certificado del servidor de base de datos de una base de datos.

Entidad de certificación (CA)	Descripción	Nombre común (CN)
rds-ca-rsa2048-g1	Utiliza una entidad de certificación con el algoritmo de clave privada RSA 2048 y el algoritmo de firma SHA256 en la mayoría de las Regiones de AWS. En las AWS GovCloud (US) Regions, este certificado utiliza una entidad de certificación con el algoritmo de clave privada RSA 2048 y el algoritmo de firma SHA384. Esta CA admite la rotación automática de certificados de servidor.	Amazon RDS *region-id entifier* Root CA RSA2048 G1
rds-ca-rsa4096-g1	Utiliza una entidad de certificación con el algoritmo de clave privada RSA 4096 y el algoritmo de firma SHA384. Esta CA admite la rotación automática de certificados de servidor.	Amazon RDS *region-id entifier* Root CA RSA4096 G1
rds-ca-ecc384-g1	Utiliza una entidad de certificación con el algoritmo de clave privada ECC 384 y el algoritmo de firma SHA384. Esta CA admite la rotación automática de certificados de servidor.	Amazon RDS *region-id entifier* Root CA ECC384 G1

> **Note**
>
> Si utiliza la AWS CLI, puede ver la validez de las entidades de certificación enumeradas anteriormente mediante describe-certificates.

Estos certificados de CA se incluyen en el paquete de certificados regionales y globales. Al utilizar la CA rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 o rds-ca-ecc384-g1 con una instancia de base de

datos, RDS administra el certificado del servidor de base de datos en la base de datos. RDS rota el certificado del servidor de base de datos de forma automática antes de que caduque.

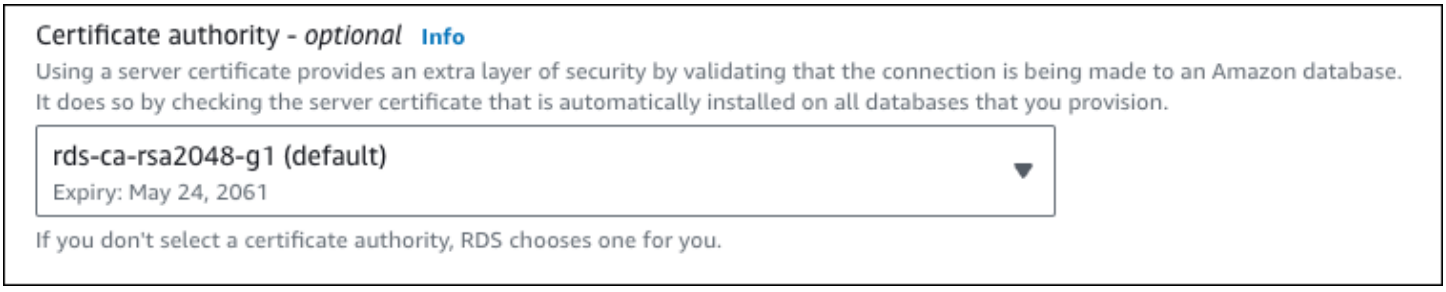
Configuración de la CA para su base de datos

Puede definir la CA para una base de datos con las tareas siguientes:

- Crear un clúster de base de datos de Aurora: puede configurar la CA para una instancia de base de datos en un clúster de Aurora al crear la primera instancia de base de datos del clúster de base de datos mediante la AWS CLI o la API de RDS. Actualmente, no puede definir la CA para las instancias de base de datos en un clúster de base de datos al crear el clúster de base de datos mediante la consola de RDS. Para obtener instrucciones, consulte Creación de un clúster de base de datos de Amazon Aurora .
- Modificar una instancia de base de datos: puede configurar la CA de una instancia de base de datos en un clúster de base de datos modificándola. Para obtener instrucciones, consulte Modificación de una instancia de base de datos en un clúster de base de datos .

> **Note**
>
> La CA predeterminada está establecida en rds-ca-rsa2048-g1. Puede anular la CA predeterminada para su Cuenta de AWS mediante el comando modify-certificates.

Las CA disponibles dependen del motor de base de datos y de la versión del motor de base de datos. Al utilizar la Consola de administración de AWS, puede elegir la CA mediante la configuración de la Entidad de certificación, tal como se muestra en la siguiente imagen.



La consola solo muestra las CA que están disponibles para le motor de base de datos y la versión del motor de base de datos. Si utiliza la AWS CLI, puede configurar la CA para una instancia de base de datos mediante los comandos create-db-instance o modify-db-instance.

Si utiliza la AWS CLI, puede ver las CA disponibles para su cuenta mediante el comando describe-certificates. Este comando también muestra la fecha de caducidad de cada CA en `ValidTill` en la salida. Puede buscar las CA que están disponibles para un motor de base de datos y una versión de motor de base de datos específicos mediante el comando describe-db-engine-versions.

El siguiente ejemplo muestra las CA disponibles para la versión predeterminada del motor de base de datos de RDS para PostgreSQL.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

Su resultado es similar al siguiente. Las CA disponibles se enumeran en `SupportedCACertificateIdentifiers`. El resultado también muestra si la versión del motor de base de datos admite la rotación del certificado sin reiniciarlo en `SupportsCertificateRotationWithoutRestart`.

```
{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
            "SupportedCACertificateIdentifiers": [
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
```

```
}
```

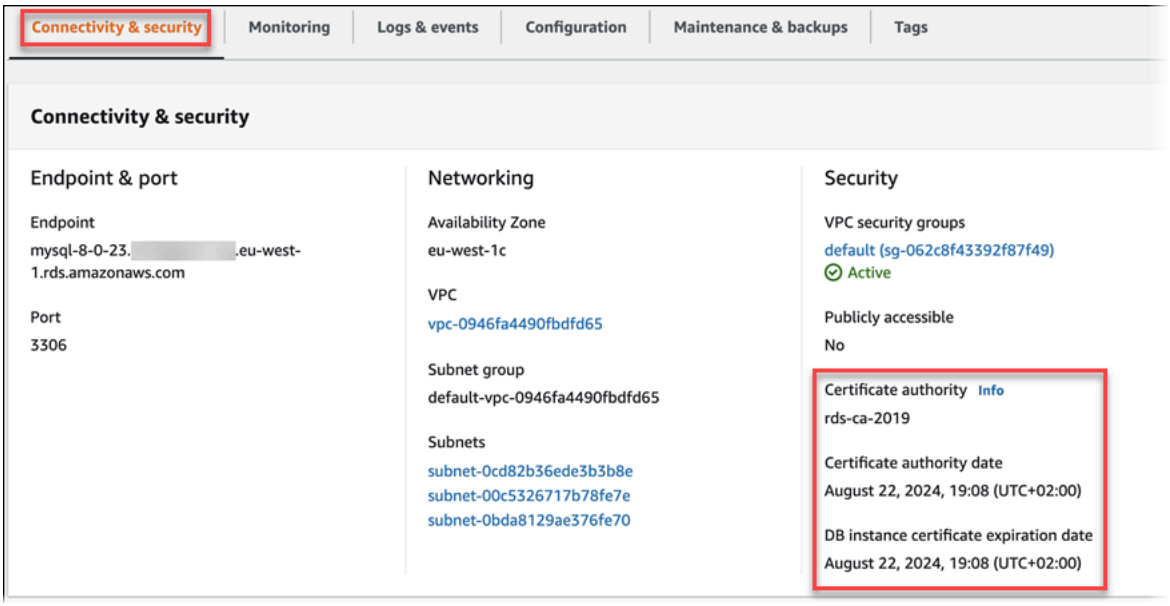
Validez del certificado del servidor de base de datos

La validez del certificado del servidor de base de datos depende del motor de base de datos y de la versión del motor de base de datos. Si la versión del motor de base de datos admite la rotación de certificados sin reinicio, el certificado del servidor de base de datos tiene una validez de 1 año. De no ser así, la validez es de 3 años.

Para obtener más información acerca de la rotación de certificados del servidor de base de datos, consulte Rotación automática de certificados del servidor.

Visualización de la CA de su instancia de base de datos

Puede ver los detalles sobre la CA de una base de datos en la pestaña Conectividad y seguridad de la consola, como se muestra en la siguiente imagen.



Si utiliza la AWS CLI, puede ver los detalles de la CA de una instancia de base de datos mediante el comando describe-db-instances.

Descarga de agrupaciones de certificados para Aurora

Cuando se conecta a la base de datos con SSL o TLS, la instancia de base de datos requiere un certificado de confianza de Amazon RDS. Seleccione el enlace correspondiente en la siguiente tabla para descargar la agrupación correspondiente a la Región de AWS donde se aloja la base de datos.

Agrupaciones de certificados por Región de AWS

Los paquetes de certificados para todas las Regiones de AWS y para regiones de GovCloud (EE. UU.) incluyen los siguientes certificados:

- `rds-ca-rsa2048-g1`
- `rds-ca-rsa4096-g1`
- `rds-ca-ecc384-g1`

Los certificados `rds-ca-rsa4096-g1` y `rds-ca-ecc384-g1` no están disponibles en las siguientes regiones:

- Asia-Pacífico (Mumbai)
- Asia-Pacífico (Melbourne)
- Oeste de Canadá (Calgary)
- Europa (Zúrich)
- Europa (España)
- Israel (Tel Aviv)

El almacén de confianza de la aplicación solo necesita registrar el certificado CA raíz. No registre los certificados CA intermedios en el almacén de confianza, ya que esto podría provocar problemas de conexión cuando RDS rote automáticamente el certificado del servidor de base de datos.

> **Note**
>
> Amazon RDS Proxy y Aurora Serverless v1 usen certificados de AWS Certificate Manager (ACM). Si está utilizando RDS Proxy, no es necesario descargar certificados de Amazon RDS ni actualizar aplicaciones que usen conexiones RDS Proxy. Para obtener más información, consulte Uso de TLS/SSL con RDS Proxy .
> Si usa Aurora Serverless v1, no es necesario descargar certificados de Amazon RDS. Para obtener más información, consulte Uso de TLS/SSL con Aurora Serverless v1 .

Para descargar una agrupación de certificados de una Región de AWS, seleccione el enlace de la Región de AWS en la que se aloja la base de datos en la tabla siguiente.

AWS Región de	Paquete de certificados (PEM)	Paquete de certificados (PKCS7)
Cualquier Región de AWS comercial	global-bundle.pem	global-bundle.p7b
EE.UU. Este (Norte de Virginia)	us-east-1-bundle.pem	us-east-1-bundle.p7b
US East (Ohio)	us-east-2-bundle.pem	us-east-2-bundle.p7b
EE.UU. Oeste (Norte de California)	us-west-1-bundle.pem	us-west-1-bundle.p7b
EE.UU. Oeste (Oregón)	us-west-2-bundle.pem	us-west-2-bundle.p7b
Africa (Cape Town)	af-south-1-bundle.pem	af-south-1-bundle.p7b
Asia Pacific (Hong Kong)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b
Asia-Pacífico (Hyderabad)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b
Asia-Pacífico (Yakarta)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
Asia-Pacífico (Malasia)	ap-southeast-5-bundle.pem	ap-southeast-5-bundle.p7b
Asia-Pacífico (Melbourne)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
Asia Pacific (Bombay)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
Asia Pacific (Osaka)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b
Asia-Pacífico (Tailandia)	ap-southeast-7-bundle.pem	ap-southeast-7-bundle.p7b
Asia-Pacífico (Tokio)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
Asia Pacific (Seoul)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
Asia Pacífico (Singapur)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
Asia Pacífico (Sídney)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b

AWS Región de	Paquete de certificados (PEM)	Paquete de certificados (PKCS7)
Canada (Central)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
Oeste de Canadá (Calgary)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
Europa (Fráncfort)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
Europe (Irlanda)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b
Europe (Londres)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
Europe (Milan)	eu-south-1-bundle.pem	eu-south-1-bundle.p7b
Europe (Paris)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b
Europa (España)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
Europa (Estocolmo)	eu-north-1-bundle.pem	eu-north-1-bundle.p7b
Europa (Zúrich)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
Israel (Tel Aviv)	il-central-1-bundle.pem	il-central-1-bundle.p7b
México (central)	mx-central-1-bundle.pem	mx-central-1-bundle.p7b
Medio Oriente (Baréin)	me-south-1-bundle.pem	me-south-1-bundle.p7b
Medio Oriente (EAU)	me-central-1-bundle.pem	me-central-1-bundle.p7b
América del Sur (São Paulo)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b
Cualquier AWS GovCloud (US) Region	global-bundle.pem	global-bundle.p7b
AWS GovCloud (EE. UU. Este)	us-gov-east-1-bundle.pem	us-gov-east-1-bundle.p7b
AWS GovCloud (EE. UU. Oeste)	us-gov-west-1-bundle.pem	us-gov-west-1-bundle.p7b

Visualización del contenido de su certificado de CA

Para comprobar el contenido del paquete de certificados de la CA, utilice el siguiente comando:

```
keytool -printcert -v -file global-bundle.pem
```

Rotar certificados SSL/TLS

Los certificados rds-ca-2019 de la entidad de certificación de Amazon RDS caducaron en agosto de 2024. Si usa o planea usar la capa de conexión segura (SSL) o la seguridad de la capa de transporte (TLS) con la verificación de certificados para conectarse a las instancias de base de datos de RDS , considere la posibilidad de utilizar uno de los nuevos certificados de entidad de certificación rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 o rds-ca-ecc384-g1. Si actualmente no utiliza SSL/TLS con verificación de certificados, es posible que aún tenga un certificado de CA caducado y que deba actualizarlo con un certificado de CA nuevo si tiene previsto utilizar SSL/TLS con verificación de certificados para conectarse a sus bases de datos de RDS.

Amazon RDS proporciona nuevos certificados de entidad de certificación como una práctica recomendada de seguridad de AWS. Para obtener información sobre los nuevos certificados y las regiones de AWS compatibles, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

Para actualizar el certificado de CA de la base de datos, utilice los métodos siguientes:

- Actualización del certificado de entidad de certificación modificando la instancia de base de datos
- Actualización del certificado de entidad de certificación mediante la aplicación de mantenimiento

Antes de actualizar sus instancias de base de datos para usar el nuevo certificado de CA, asegúrese de actualizar sus clientes o aplicaciones que se conectan a sus bases de datos de RDS.

Consideraciones sobre la rotación de certificados

Tenga en cuenta las siguientes situaciones antes de rotar el certificado:

- Amazon RDS Proxy y Aurora Serverless v1 usen certificados de AWS Certificate Manager (ACM). Si utiliza RDS Proxy, al rotar el certificado SSL/TLS, no es necesario que actualice las aplicaciones que utilizan las conexiones de RDS Proxy. Para obtener más información, consulte Uso de TLS/SSL con RDS Proxy .

- Si usa Aurora Serverless v1, no es necesario descargar certificados de Amazon RDS. Para obtener más información, consulte Uso de TLS/SSL con Aurora Serverless v1 .
- Si utiliza la versión 1.15 de una aplicación Go con una instancia de base de datos que se haya creado o actualizado con el certificado rds-ca-2019 antes del 28 de julio de 2020, debe actualizar el certificado de nuevo. Actualice el certificado a rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 o rds-ca-ecc384-g1 en función de su motor .

  Utilice el comando `modify-db-instance` con el nuevo identificador de certificado de CA. Puede buscar las CA que están disponibles para un motor de base de datos y una versión de motor de base de datos específicos mediante el comando `describe-db-engine-versions`.

  Si creó su base de datos o actualizó su certificado después del 28 de julio de 2020, no se requiere ninguna acción. Para obtener más información, consulte Go GitHub issue #39568.

Actualización del certificado de entidad de certificación modificando la instancia de base de datos

En el siguiente ejemplo, se actualiza el certificado de la entidad de certificación de rds-ca-2019 a rds-ca-rsa2048-g1. Puede elegir un certificado diferente. Para obtener más información, consulte Entidades de certificación.

Actualice el almacén de confianza de aplicaciones para reducir el tiempo de inactividad asociado a la actualización del certificado de CA. Para obtener más información acerca de los reinicios asociados a la rotación de certificados de la entidad de certificación, consulte Rotación automática de certificados del servidor.

Actualización del certificado de entidad de certificación modificando la instancia de base de datos

1. Descargue el nuevo certificado SSL/TLS como se describe en Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.
2. Actualice las aplicaciones para que usen el nuevo certificado SSL/TLS.

   Los métodos para actualizar aplicaciones para nuevos certificados SSL/TLS dependen de sus aplicaciones específicas. Trabaje con sus desarrolladores de aplicaciones para actualizar los certificados SSL/TLS para sus aplicaciones.

   Para obtener información sobre la comprobación de conexiones SSL/TLS y la actualización de cada motor de base de datos, consulte los siguientes temas:

- Actualización de aplicaciones para la conexión a los clústeres de base de datos de MySQL de Aurora con los nuevos certificados TLS
- Actualización de aplicaciones para la conexión a los clústeres de base de datos de PostgreSQL de Aurora con los nuevos certificados SSL/TLS

Para obtener el mismo script que actualice un almacén de confianza para un sistema operativo Linux, consulte Script de muestra para la importación de certificados en su almacén de confianza.

> **Note**
>
> El paquete de certificados contiene certificados tanto para la CA antigua como para la nueva, por lo que puede actualizar su aplicación de forma segura y mantener la conectividad durante el período de transición. Si utiliza AWS Database Migration Service para migrar una base de datos a un clúster de base de datos, recomendamos utilizar el paquete de certificados para garantizar la conectividad durante la migración.

3. Modifique la instancia de base de datos para cambiar la CA de rds-ca-2019 a rds-ca-rsa2048-g1. Para comprobar si la base de datos requiere un reinicio para actualizar los certificados de CA, utilice el comando describe-db-engine-versions y compruebe el indicador `SupportsCertificateRotationWithoutRestart`.

> **Note**
>
> Reinicie su clúster de Babelfish después de modificarlo para actualizar el certificado de CA.

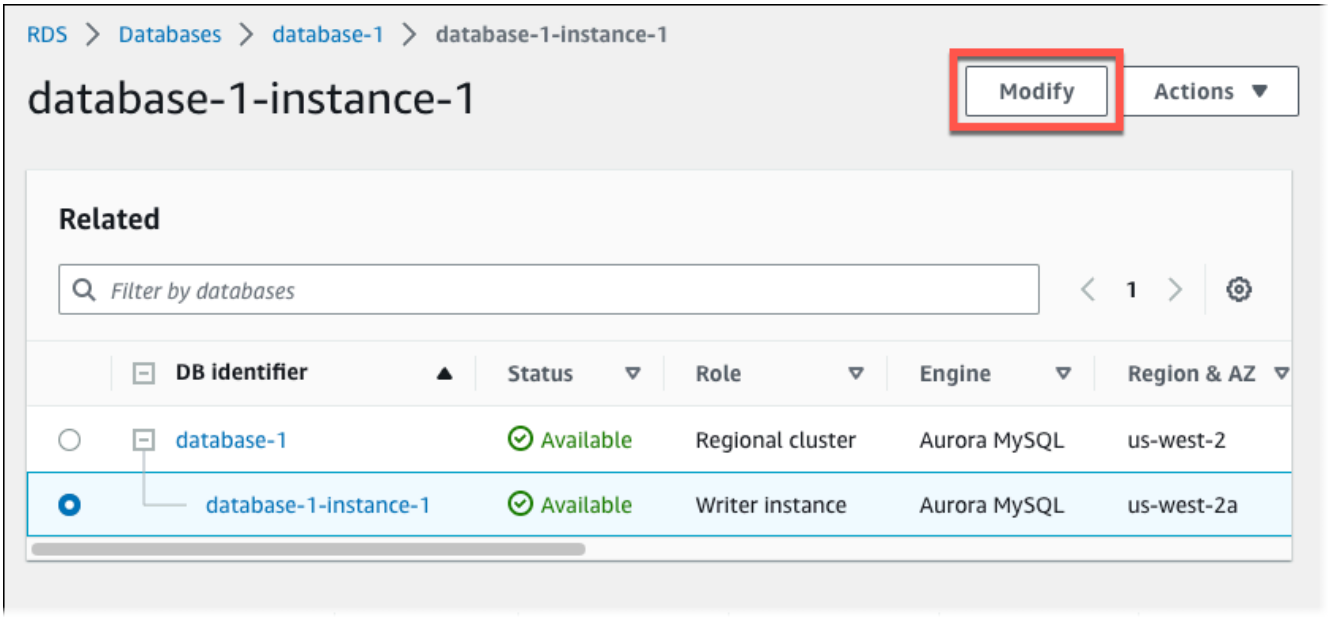
> **Important**
>
> Si tiene problemas de conectividad después de que el certificado caduque, utilice la opción de aplicación inmediata. Seleccione Apply immediately (Aplicar inmediatamente) en la consola o especifique la opción `--apply-immediately` con la AWS CLI. De manera predeterminada, esta operación está programada para ejecutarse durante su próximo periodo de mantenimiento.

Para establecer una invalidación para la entidad de certificación del clúster que sea diferente de la entidad de certificación de RDS predeterminada, utilice el comando de la CLI modify-certificates.

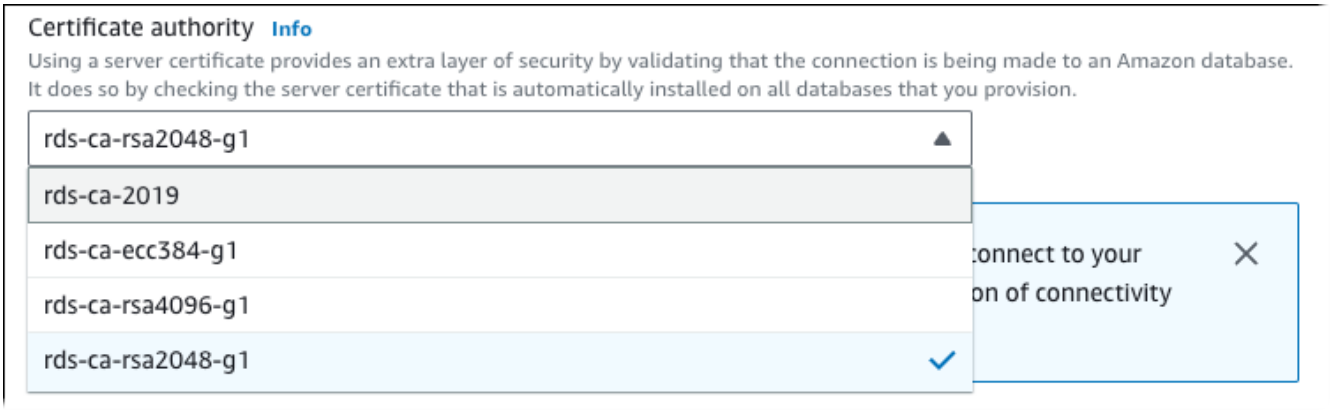
Puede utilizar la Consola de administración de AWS o la AWS CLI para cambiar el certificado de CA de rds-ca-2019 a rds-ca-rsa2048-g1 para una instancia de base de datos .

Console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Bases de datos y, a continuación, elija la instancia de base de datos que desea modificar.
3. Elija Modificar.



4. En la sección Conectividad, elija rds-ca-rsa2048-g1.



5. Elija Continue y consulte el resumen de las modificaciones.
6. Para aplicar los cambios inmediatamente, elija Apply immediately.
7. En la página de confirmación, revise los cambios. Si son correctos, elija Modificar la instancia de base de datos para guardar los cambios.

> **Important**
>
> Cuando programe esta operación, asegúrese de haber actualizado de antemano su tienda de confianza del lado del cliente.

O bien, elija Back (Atrás) para editar los cambios o Cancel (Cancelar) para cancelarlos.

AWS CLI

Para utilizar la AWS CLI para cambiar la CA de rds-ca-2019 a rds-ca-rsa2048-g1 para una instancia de base de datos , llame al comando modify-db-instance o modify-db-clúster. Especifique el identificador de instancia de base de datos y la opción `--ca-certificate-identifier`.

Utilice el parámetro `--apply-immediately` para aplicar la actualización inmediatamente. De manera predeterminada, esta operación está programada para ejecutarse durante su próximo periodo de mantenimiento.

> **Important**
>
> Cuando programe esta operación, asegúrese de haber actualizado de antemano su tienda de confianza del lado del cliente.

Example

En el siguiente ejemplo, se modifica `mydbinstance` al establecer el certificado de CA en `rds-ca-rsa2048-g1`.

Para Linux, macOS o:Unix

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Para Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> Si su instancia requiere reinicio, puede utilizar el comando de la CLI modify-db-instance y especificar la opción `--no-certificate-rotation-restart`.

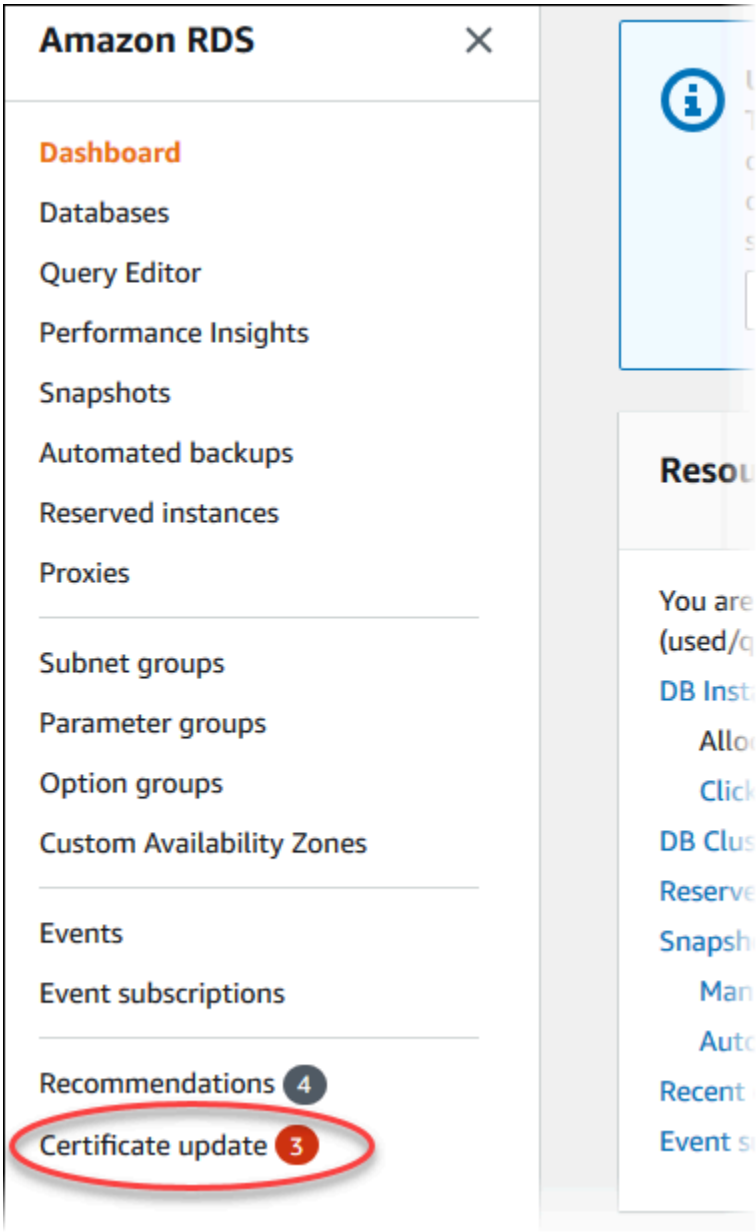
Actualización del certificado de entidad de certificación mediante la aplicación de mantenimiento

Realice los siguientes pasos para actualizar el certificado de CA aplicando el mantenimiento.

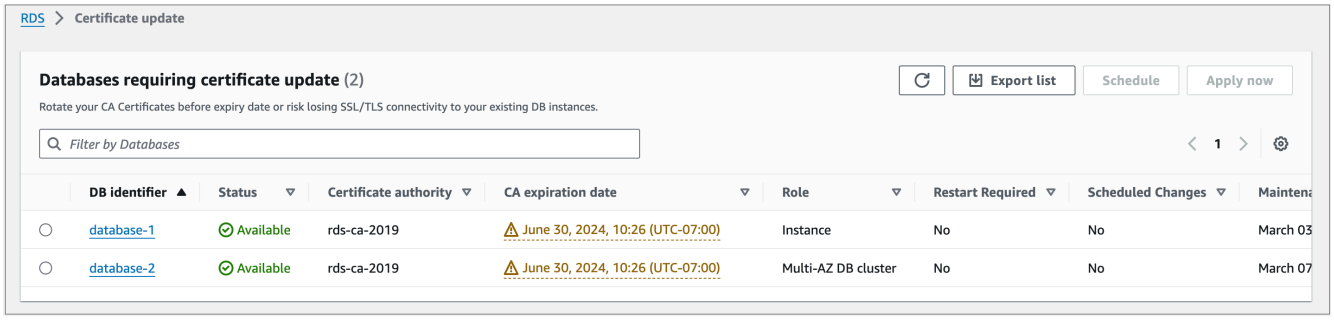
Console

Actualización del certificado de CA aplicando el mantenimiento

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Actualización del certificado.



Se muestra la página Bases de datos que requieren actualización de certificados.



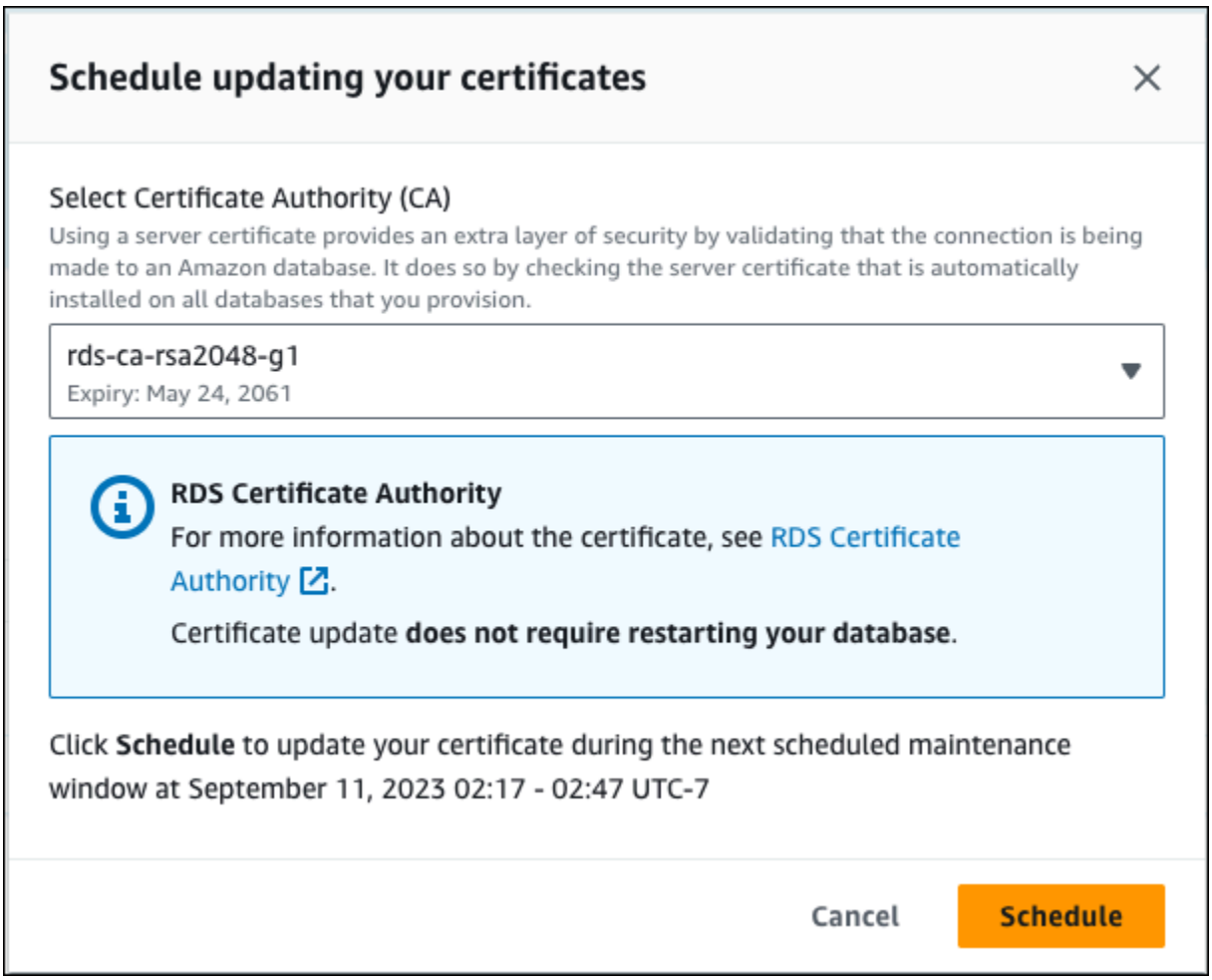
> **Note**
>
> Esta página solo muestra las instancias de base de datos de la Región de AWS actual. Si tiene instancias de base de datos en más de una Región de AWS, consulte esta página en cada Región de AWS para ver todas las instancias de base de datos con certificados SSL/TLS antiguos.

3. Elija la instancia de base de datos que desea actualizar.

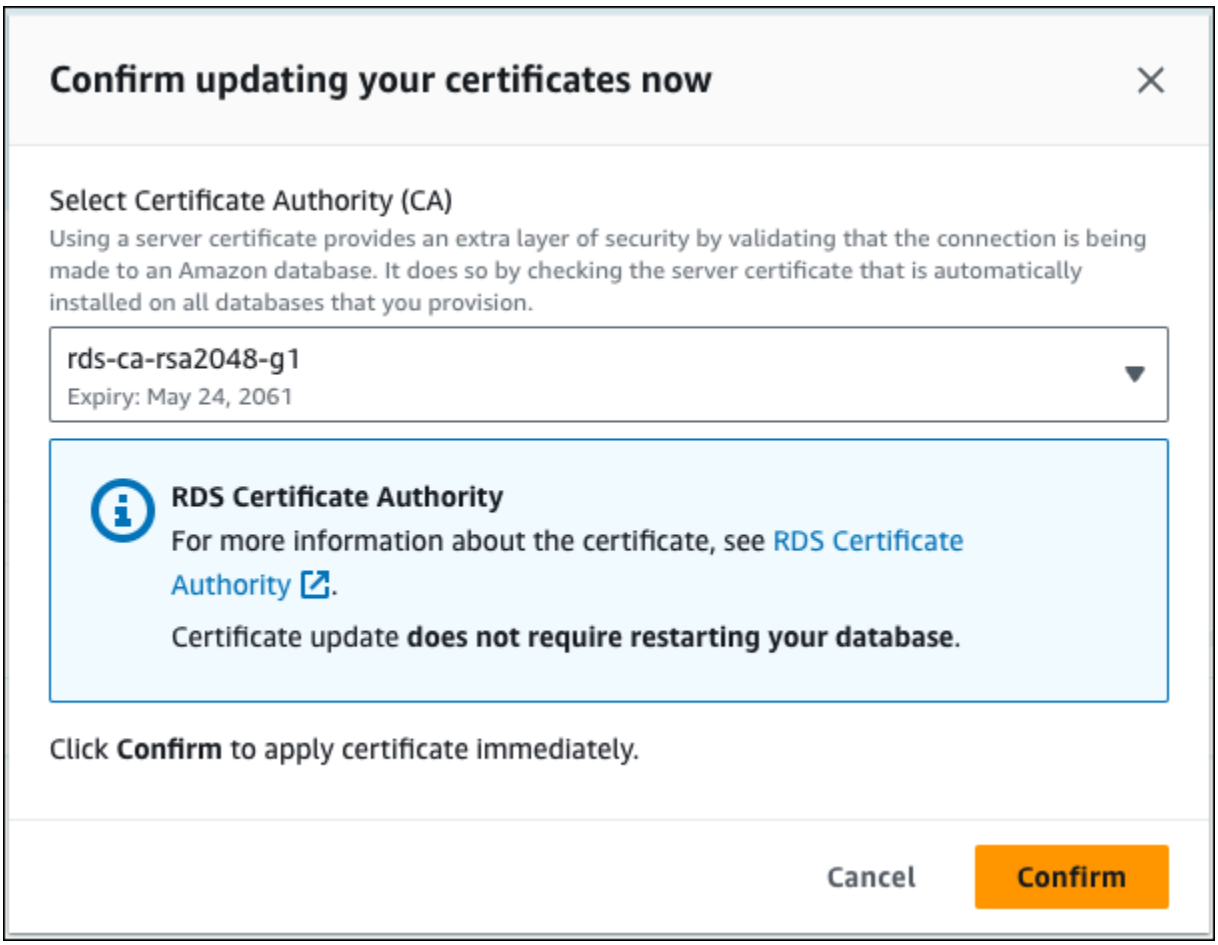
   Elija Programación para programar la rotación de certificados para la siguiente ventana de mantenimiento. Para aplicar la rotación inmediatamente, elija Aplicar ahora.

   > **Important**
   >
   > Si tiene problemas de conectividad después de que el certificado caduque, utilice la opción Aplicar ahora.

4. a. Si elige Programación, se le solicitará que confirme la rotación del certificado de CA. Este mensaje también indica el período programado para la actualización.



b. Si elige Aplicar ahora, se le solicita que confirme la rotación del certificado de CA.



> **Important**
>
> Antes de programar la rotación de certificados de CA en la base de datos, actualice las aplicaciones cliente que utilicen SSL/TLS y el certificado de servidor para conectarse. Estas actualizaciones son específicas de su motor de base de datos. Después de actualizar estas aplicaciones cliente, puede confirmar la rotación del certificado de CA.

Para continuar, seleccione la casilla de verificación y, a continuación, seleccione Confirm (Confirmar).

5. Repita los pasos 3 y 4 para cada instancia de base de datos que desee actualizar.

Rotación automática de certificados del servidor

Si su CA raíz admite la rotación automática de certificados del servidor, RDS administra automáticamente la rotación del certificado del servidor de base de datos. RDS utiliza la misma CA raíz para esta rotación automática, por lo que no es necesario descargar un nuevo paquete de CA. Consulte Entidades de certificación.

La rotación y la validez del certificado del servidor de base de datos dependen del motor de base de datos:

- Si su motor de base de datos admite la rotación sin reinicio, RDS rota automáticamente el certificado del servidor de base de datos sin que usted tenga que realizar ninguna acción. RDS intenta rotar el certificado del servidor de base de datos en el período de mantenimiento que prefiera a la mitad de la vida del certificado del servidor de base de datos. El nuevo certificado del servidor de base de datos es válido durante 12 meses.
- Si el motor de base de datos no admite la rotación sin reiniciar, Amazon RDS hace visible una acción de mantenimiento pendiente `server-certificate-rotation` a través de la API Describe-Pending-Maintenance-Actions, al final de la vida útil del certificado o al menos 3 meses antes de que caduque. Puede aplicar la rotación mediante la API apply-pending-maintenance-action. El nuevo certificado del servidor de base de datos es válido durante 36 meses.

Utilice el comando describe-db-engine-versions e inspeccione el indicador `SupportsCertificateRotationWithoutRestart` para identificar si la versión del motor de base de datos admite la rotación del certificado sin reinicio. Para obtener más información, consulte Configuración de la CA para su base de datos .

Script de muestra para la importación de certificados en su almacén de confianza

A continuación se muestran scripts de shell de ejemplo que importan el paquete de certificados a un almacén de confianza.

Cada script de shell de muestra utiliza keytool, que forma parte del kit de desarrollo de Java (JDK). Para obtener información sobre la instalación de JDK, consulte la Guía de instalación de JDK.

Linux

A continuación se muestra un ejemplo de script de intérprete de comandos que importa el paquete de certificados a un almacén de confianza en un sistema operativo Linux.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

A continuación se muestra un ejemplo de script de intérprete de comandos que importa el paquete de certificados a un almacén de confianza en macOS.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit
```

```
curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

Para obtener más información sobre los procedimientos recomendados al usar SSL con Amazon RDS, consulte Best practices for successful SSL connections to Amazon RDS para Oracle.

Privacidad del tráfico entre redes

Las conexiones están protegidas entre Amazon Aurora y las aplicaciones en las instalaciones y entre Amazon Aurora y otros recursos de AWS dentro de la misma región de AWS.

Tráfico entre el servicio y las aplicaciones y clientes locales

Tiene dos opciones de conectividad entre su red privada y AWS:

- Una conexión de Site-to-Site VPN de AWS. Para obtener más información, consulte ¿Qué es AWS Site-to-Site VPN?
- Una conexión de Direct Connect. Para obtener más información, consulte ¿Qué es Direct Connect?

Accederá a Amazon Aurora a través de la red mediante las operaciones de la API publicadas por AWS. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puedes utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Administración de la identidad y el acceso en Amazon Aurora

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda a los gestionadores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon RDS. IAM es un servicio de Servicio de AWS que se puedes utilizar sin cargo adicional.

Temas

- Público
- Autenticación con identidades
- Administración de acceso mediante políticas
- Cómo funciona Amazon Aurora con IAM
- Ejemplos de políticas basadas en identidad para Amazon Aurora
- Políticas administradas por AWS para Amazon RDS
- Actualizaciones de Amazon RDS a políticas administradas por AWS
- Prevención de los problemas del suplente confuso entre servicios
- Autenticación de bases de datos de IAM
- Solución de problemas de identidades y accesos en Amazon Aurora

Público

La forma en que utiliza AWS Identity and Access Management (IAM) difiere en función del trabajo que realiza en Amazon Aurora.

Usuario de servicio: si utiliza el servicio Aurora para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Aurora para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos a su administrador. Si no puede acceder a una característica en Aurora, consulte Solución de problemas de identidades y accesos en Amazon Aurora.

Administrador de servicio: si está a cargo de los recursos de –Aurora en su empresa, probablemente tenga acceso completo a Aurora. Su trabajo consiste en determinar qué a características y recursos de Aurora deben acceder sus empleados. A continuación, debe enviar solicitudes a su administrador

de para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con Aurora, consulte Cómo funciona Amazon Aurora con IAM.

Administrador: si es un administrador de IAM, es posible que quiera conocer información sobre cómo escribir políticas para administrar el acceso a Aurora. Para ver ejemplos de políticas basadas en la identidad de Aurora que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidad para Amazon Aurora.

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado como Usuario raíz de la cuenta de AWS, un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de un origen de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales de Google/Facebook. Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In.

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Usuario raíz de la cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión llamada Cuenta de AWS del usuario raíz que tiene acceso completo a todos los Servicios de AWS y recursos. Recomendamos encarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidades para acceder a Servicios de AWS con credenciales temporales.

Una identidad federada es un usuario del directorio empresarial, proveedor de identidades web o Directory Service que accede a Servicios de AWS mediante credenciales de un origen de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos AWS IAM Identity Center. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales a largo plazo. Para obtener más información, consulte Solicitar que los usuarios humanos utilicen la federación con un proveedor de identidades para acceder a AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Puede autenticar en si clúster de bases de datos utilizando autenticación de base de datos de IAM.

La autenticación de base de datos de IAM funciona con Aurora. Para obtener más información sobre la autenticación en su clúster de bases de datos con IAM, consulte Autenticación de bases de datos de IAM .

Roles de IAM

Un rol de IAM es una identidad de la Cuenta de AWS que dispone de permisos específicos. Es similar a un usuario, pero no está asociado a una persona específica. Puede asumir temporalmente un rol de IAM en la Consola de administración de AWS cambiando de roles. Puede asumir un rol llamando a una operación de la AWS CLI o de la API de AWS, o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Permisos de usuario temporales: un usuario puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles de federación, consulte Crear un rol para un proveedor de identidad de terceros (federación) en la

Guía de usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulta Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center.

- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. No obstante, con algunos Servicios de AWS se puede adjuntar una política directamente a un recurso (en lugar de utilizar un rol como representante). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan características de otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo: las sesiones de acceso directo (FAS) utilizan los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
  - Rol vinculado a los servicios: un rol vinculado a servicios es un tipo de rol de servicio que está vinculado a un Servicio de AWS. El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados a servicios aparecen en la Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.
- Aplicaciones que se ejecutan en Amazon EC2: puede utilizar un rol de IAM que le permita administrar credenciales temporales para las aplicaciones que se ejecutan en una instancia de EC2 y realizan solicitudes a la AWS CLI o a la API de AWS. Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar un rol de AWS a una instancia de EC2 y ponerla a disposición de todas las aplicaciones, cree un perfil de instancia adjuntado a

la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para obtener más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información acerca del uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en vez de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades de IAM o recursos de AWS. Una política es un objeto de AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando una entidad (usuario raíz, usuario o rol de IAM) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Un administrador puede utilizar las políticas para especificar quién tiene acceso a los recursos de AWS y qué acciones se pueden realizar en dichos recursos. Cada entidad de IAM (conjunto de permisos o rol) comienza sin permisos. En otras palabras, de forma predeterminada, los usuarios no pueden hacer nada, ni siquiera cambiar sus propias contraseñas. Para conceder permiso a un usuario para hacer algo, el administrador debe adjuntarle una política de permisos. O bien el administrador puede agregar al usuario a un grupo que tenga los permisos necesarios. Cuando el administrador concede permisos a un grupo, todos los usuarios de ese grupo obtienen los permisos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con dicha política puede obtener información del usuario de la Consola de administración de AWS, la AWS CLI o la API de AWS.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como por ejemplo, un conjunto de permisos o un rol. Estas políticas controlan qué acciones puede realizar dicha identidad, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidades, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único conjunto de permisos o un rol. Las políticas administradas son políticas independientes que puede asociar a varios conjuntos de permisos o roles de su cuenta de AWS. Las políticas administradas incluyen las políticas administradas por AWS y las políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Para obtener información sobre las políticas administradas de AWS que son específicas de Amazon Aurora, consulte Políticas administradas por AWS para Amazon RDS.

Otros tipos de políticas

AWS admite otros tipos de políticas adicionales menos frecuentes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad (conjunto de permisos o rol). Puede establecer un límite de permisos para una identidad. Los permisos resultantes son la intersección de las políticas basadas en identidad de la entidad y los límites de sus permisos. Las políticas basadas en recursos que especifiquen el conjunto de permisos o rol en el campo `Principal` no están restringidos por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicio (SCP): las SCP son políticas de JSON que especifican los permisos máximos de una organización o una unidad organizativa (OU) en AWS Organizations. AWS Organizations es un servicio que le permite agrupar y administrar de manera centralizada varias cuentas de AWS que posea su empresa. Si habilita todas las características en una empresa, entonces podrá aplicar políticas de control de servicio (SCP) a una o todas sus cuentas. Una SCP limita los permisos para las entidades de las cuentas de miembros, incluido cada Usuario raíz de la cuenta de AWS. Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidad del conjunto de permisos o rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas

anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información acerca de cómo AWS decide si permitir o no una solicitud cuando hay varios tipos de políticas implicados, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon Aurora con IAM

Antes de utilizar IAM para administrar el acceso a Amazon Aurora, debe saber qué características de IAM están disponibles para usar con Aurora.

En la siguiente tabla, encontrará las características de IAM que puede usar con Amazon Aurora:

Característica de IAM	Compatibilidad de Amazon Aurora
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACL	No
Control de acceso basado en atributos (ABAC) (etiquetas en políticas)	Sí
Credenciales temporales	Sí
Sesiones de acceso directo	Sí
Roles de servicio	Sí

Característica de IAM	Compatibilidad de Amazon Aurora
Roles vinculados al servicio	Sí

Para obtener una perspectiva general de cómo funciona Amazon Aurora y otros servicios de AWS con IAM, consulte los servicios de AWS que funcionan con IAM en la guía del usuario de IAM.

Temas

- Políticas de Aurora basadas en identidades
- Políticas basadas en recursos de Aurora
- Acciones de política de Aurora
- Recursos de políticas de Aurora
- Claves de condición de políticas para Aurora
- Listas de control de acceso (ACL) de Aurora
- Control de acceso basado en atributos (ABAC) en políticas con etiquetas de Aurora
- Uso de credenciales temporales con Aurora
- Sesiones de acceso directo para Aurora
- Roles de servicio para Aurora
- Roles vinculados a servicios para Aurora

Políticas de Aurora basadas en identidades

Admite las políticas basadas en identidad: sí.

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas de Aurora basadas en identidades

Para ver ejemplos de políticas basadas en identidad de Aurora, consulte Ejemplos de políticas basadas en identidad para Amazon Aurora.

Políticas basadas en recursos de Aurora

Admite políticas basadas en recursos: no.

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o Servicios de AWS.

Para habilitar el acceso entre cuentas, puedes especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Acciones de política de Aurora

Admite las acciones de política: sí.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones de políticas de Aurora utilizan el siguiente prefijo antes de la acción: `rds:`. Por ejemplo, para conceder a alguien permiso para eliminar un punto de enlace de Amazon RDS con la operación de la API `DescribeDBInstances` de `rds:DescribeDBInstances`, incluya la acción en su política. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. Aurora define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

Para especificar varias acciones de en una única instrucción, sepárelas con comas del siguiente modo.

```
"Action": [
      "rds:action1",
      "rds:action2"
```

Puede utilizar caracteres comodín (*) para especificar varias acciones . Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción.

```
"Action": "rds:Describe*"
```

Para ver una lista de las acciones de Aurora, consulte Acciones definidas por Amazon RDS en la referencia de autorizaciones de servicio.

Recursos de políticas de Aurora

Admite los recursos de políticas: sí.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Para las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

El recurso de instancia de base de datos tiene el siguiente nombre de recurso de Amazon (ARN).

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

Para obtener más información acerca del formato de los ARN, consulte Nombres de recursos de Amazon (ARN) y espacios de nombres de servicios de AWS.

Por ejemplo, para especificar la instancia de base de datos `dbtest` en su instrucción, utilice el siguiente ARN.

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

Para especificar todas las instancias de base de datos que pertenecen a una cuenta específica, utilice el carácter comodín (*).

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

Algunas operaciones de API de RDS, como las empleadas para la creación de recursos, no se pueden llevar a cabo en un recurso específico. En dichos casos, utilice el carácter comodín (*).

```
"Resource": "*"
```

En muchas operaciones de la API de Amazon RDS se utilizan varios recursos. Por ejemplo, `CreateDBInstance` crea una instancia de base de datos. Puede especificar que un usuario de debe usar un grupo de seguridad y un grupo de parámetros específicos al crear una instancia de base de datos. Para especificar varios recursos en una única instrucción, separe los ARN con comas.

```
"Resource": [
      "resource1",
      "resource2"
```

Para ver una lista de las acciones de Aurora, consulte Recursos definidos por Amazon RDS en la referencia de autorizaciones de servicio. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por Amazon RDS.

Claves de condición de políticas para Aurora

Admite claves de condición de políticas específicas del servicio: sí.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puedes crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Aurora define su propio conjunto de claves de condición y también admite el uso de algunas claves de condición globales. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Todas las operaciones de API de RDS admiten la clave de condición `aws:RequestedRegion`.

Para ver una lista de las claves de condición de Aurora, consulte Claves de condición de Amazon RDS en la referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por Amazon RDS.

Listas de control de acceso (ACL) de Aurora

Admite las listas de control de acceso (ACL): no

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Control de acceso basado en atributos (ABAC) en políticas con etiquetas de Aurora

Admite las etiquetas de control de acceso basado en atributos (ABAC) en las políticas: sí

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos denominados etiquetas. Puede asociar etiquetas a entidades de IAM y recursos de AWS y, a continuación, diseñar políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Para obtener más información acerca del etiquetado de recursos de Aurora, consulte Especificación de condiciones: uso de etiquetas personalizadas. Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Conceda permiso para acciones en un recurso con una etiqueta específica con dos valores diferentes..

Uso de credenciales temporales con Aurora

Admite credenciales temporales: sí.

Las credenciales temporales proporcionan acceso a corto plazo a los recursos de AWS y se crean automáticamente cuando se utiliza la federación o se cambia de rol. AWS recomienda que genere de forma dinámica credenciales temporales en lugar de usar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM y Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Sesiones de acceso directo para Aurora

Admite sesiones de acceso directo: sí.

Las sesiones de acceso directo (FAS) utilizan los permisos de la entidad principal para llamar a un Servicio de AWS, combinados con el Servicio de AWS solicitante para realizar solicitudes a servicios posteriores. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Roles de servicio para Aurora

Admite roles de servicio: sí.

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Aurora. Edite los roles de servicio solo cuando Aurora proporcione orientación para hacerlo.

Roles vinculados a servicios para Aurora

Admite roles vinculados al servicio: sí.

Un rol vinculado a servicios es un tipo de rol de servicio que está vinculado a un Servicio de AWS. El servicio puedes asumir el rol para realizar una acción en su nombre. Los roles vinculados a servicios aparecen en la Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para obtener más información acerca de cómo usar los roles vinculados a servicios de Aurora, consulte Uso de roles vinculados a servicios de Amazon Aurora.

Ejemplos de políticas basadas en identidad para Amazon Aurora

De forma predeterminada, los conjuntos de permisos y roles no tienen permiso para crear ni modificar recursos de Aurora. Tampoco pueden realizar tareas mediante la Consola de administración de AWS, la AWS CLI o la API de AWS. Un administrador debe crear políticas de IAM que concedan conjuntos de permisos y permisos de roles para realizar operaciones de API concretas en los recursos especificados necesarios. El administrador debe asociar esas políticas a los conjuntos de permisos o roles que necesiten esos permisos.

Para obtener más información acerca de cómo crear una política basada en identidad de IAM con estos documentos de políticas de JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Temas

- Prácticas recomendadas relativas a políticas
- Mediante la consola de Aurora
- Permisos necesarios para usar la consola
- Cómo permitir a los usuarios consultar sus propios permisos
- Políticas de permisos para crear, modificar y eliminar recursos en Aurora
- Políticas de ejemplo: uso de claves de condición
- Especificación de condiciones: uso de etiquetas personalizadas
- Concesión de permisos para etiquetar recursos de Aurora durante la creación

Prácticas recomendadas relativas a políticas

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon RDS de su cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comienza con las políticas administradas por AWSy continúa con los permisos de privilegio mínimo: a fin de comenzar a conceder permisos a los usuarios y las cargas de tarea, utiliza las políticas administradas por AWS, que conceden permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Se recomienda definir políticas gestionadas por el cliente de AWS específicas para sus casos de uso a fin de reducir aún más los permisos. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puedes escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado como, por ejemplo, CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesite usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para obtener una mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA

a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Mediante la consola de Aurora

Para acceder a la consola de Amazon Aurora, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y consultar los detalles sobre los recursos de Amazon Aurora en su cuenta de Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que conceda permisos mínimos para la consola a los usuarios que solo realizan llamadas a la AWS CLI o a la API de AWS. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intenta realizar.

Para asegurarse de que esas entidades puedan seguir usando la consola de Aurora, asocie también la siguiente política administrada por AWS a las entidades.

```
AmazonRDSReadOnlyAccess
```

Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Permisos necesarios para usar la consola

Para que un usuario pueda trabajar con la consola, debe tener un conjunto mínimo de permisos. Estos permisos permiten al usuario describir los recursos de Amazon Aurora de su cuenta de AWS y proporcionar otra información relacionada, incluida información de red y seguridad de Amazon EC2.

Si crea una política de IAM que sea más restrictiva que el mínimo de permisos necesarios, la consola no funciona del modo esperado para los usuarios con esa política de IAM. Para asegurarse de que esos usuarios puedan seguir usando la consola, asocie también la política administrada `AmazonRDSReadOnlyAccess` al usuario, según se explica en Administración de acceso mediante políticas.

No es necesario que conceda permisos mínimos para la consola a los usuarios que solo realizan llamadas a la AWS CLI o a la API de Amazon RDS.

La siguiente política concede acceso completo a todos los recursos de Amazon Aurora para la cuenta de AWS raíz:

```
AmazonRDSFullAccess
```

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Políticas de permisos para crear, modificar y eliminar recursos en Aurora

En las siguientes secciones, encontrará ejemplos de políticas de permisos para otorgar y limitar el acceso a los recursos:

Permitir a un usuario crear en instancias de base de datos en una cuenta de AWS

A continuación, se muestra el ejemplo de una política que permite que la cuenta con el ID `123456789012` pueda crear instancias de base de datos para su cuenta AWS. La política requiere que el nombre de la nueva instancia de base de datos comience por `test`. La nueva instancia de base de datos también debe utilizar el motor de base de datos MySQL y la clase de instancia de base de datos `db.t2.micro`. Además, la nueva instancia de base de datos debe usar un grupo de opciones y un grupo de parámetros de base de datos que comience por `default` y debe utilizar el grupo de subred `default`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
```

```
            }
         }
      }
   ]
}
```

En la política se incluye una sola instrucción que especifica los siguientes permisos para el usuario de:

- La política permite a la cuenta crear una instancia de base de datos utilizando la operación CreateDBInstance de la API (esto también se aplica al comando create-db-instance de la AWS CLI y a la Consola de administración de AWS).
- El elemento `Resource` especifica que el usuario puede realizar acciones en o con recursos. Puede especificar los recursos mediante un nombre de recurso de Amazon (ARN). Este ARN incluye el nombre del servicio al que pertenece el recurso (`rds`), la región AWS (`*` indica cualquier región de este ejemplo), el número de cuenta de AWS (`123456789012` es el número de cuenta en este ejemplo) y el tipo de recurso. Para obtener más información acerca de la creación de nombres ARN, consulte Nombres de recursos de Amazon (ARN) en Amazon RDS.

  El elemento `Resource` del ejemplo especifica las siguientes restricciones políticas en los recursos del usuario:

  - El identificador de instancias de bases de datos para la nueva instancia de base de datos debe comenzar por `test` (por ejemplo, `testCustomerData1`, `test-region2-data`).
  - El grupo de opciones de la nueva instancia de base de datos debe empezar por `default`.
  - El grupo de parámetros de base de datos de la nueva instancia de base de datos debe empezar por `default`.
  - El grupo de subred de la nueva instancia de base de datos debe ser el grupo de subred `default`.
- El elemento `Condition` especifica que el motor de base de datos debe ser MySQL, mientras que la clase de instancia de base de datos debe ser `db.t2.micro`. El elemento `Condition` especifica las condiciones en las que se debe aplicar una política. Puede añadir permisos o restricciones adicionales mediante el elemento `Condition`. Para obtener más información acerca de cómo especificar condiciones, consulte Claves de condición de políticas para Aurora. Este ejemplo especifica el estado del `rds:DatabaseEngine` y la `rds:DatabaseClass`. Para obtener más información acerca de los valores de estado válidos para `rds:DatabaseEngine`, consulte la lista bajo el parámetro `Engine` en CreateDBInstance. Para obtener información acerca de

los valores de estado válidos para `rds:DatabaseClass`, consulte Motores de base de datos compatibles para clases de instancia de base de datos.

La política no especifica el elemento `Principal`, ya que en una política basada en la identidad no se especifica el elemento principal que obtiene el permiso. Al asociar una política a un usuario, el usuario es la entidad principal implícita. Cuando se asocia una política de permisos a un rol de IAM, la entidad principal identificada en la política de confianza del rol obtiene los permisos.

Para ver una lista de las acciones de Aurora, consulte Acciones definidas por Amazon RDS en la referencia de autorizaciones de servicio.

Permitir que un usuario realice cualquier acción Describe con cualquier recurso de RDS

La siguiente política de permisos concede permisos a un usuario para ejecutar todas las acciones que empiezan por `Describe`. Estas acciones muestran información acerca de un recurso de RDS, como una instancia de base de datos. El carácter de comodín (*) en el elemento `Resource` indica que las acciones están permitidas para todos los recursos de Amazon Aurora que pertenecen a la cuenta.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

Permitirle al usuario crear una instancia de base de datos que use los grupos de parámetros de base de datos y de subredes especificados

La política de permisos siguiente otorga permisos para permitir que el usuario solo pueda crear una instancia de base de datos que use el grupo de parámetros de base de datos `mydbpg` y el grupo de subredes de base de datos `mydbsubnetgroup`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
         ]
      }
   ]
}
```

Conceda permiso para acciones en un recurso con una etiqueta específica con dos valores diferentes.

Puede utilizar las condiciones de su política basada en la identidad para controlar el acceso a los recursos de Aurora basados en etiquetas. La siguiente política da permiso para aplicar la operación de API `CreateDBSnapshot` y `stage` en instancias de base de datos con la etiqueta `stage` establecida en `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
```

```
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La siguiente política da permiso para aplicar la operación de API `ModifyDBInstance` en instancias de base de datos con la etiqueta `stage` establecida en `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
```

```
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

Evitar que un usuario elimine una instancia de base de datos

La siguiente política de permisos concede permisos para impedir que un usuario elimine una instancia de base de datos específica. Por ejemplo, puede servir para impedir la eliminación de instancias de base de datos de producción a cualquier usuario que no sea un administrador.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

Denegar todo el acceso a un recurso

Puede denegar explícitamente el acceso a un recurso. Las políticas de denegación tienen prioridad sobre las políticas de permiso. La política siguiente niega explícitamente a un usuario la capacidad de administrar un recurso:

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
   ]
}
```

Políticas de ejemplo: uso de claves de condición

Los siguientes ejemplos muestran cómo puede usar claves de condición en las políticas de permisos de IAM para Amazon Aurora.

Ejemplo 1: conceder permiso para crear una instancia de base de datos que utilice un motor de base de datos específico y no sea Multi-AZ.

La siguiente política utiliza una clave de condición de RDS y permite al usuario crear solamente instancias de base de datos que utilizan el motor de base de datos MySQL y no utilizan Multi-AZ. El elemento `Condition` indica el requisito de que el motor de base de datos sea MySQL.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
```

```
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
}
```

Ejemplo 2: denegar permiso explícitamente para crear instancias de base de datos para determinadas clases de instancia de base de datos y crear instancias de base de datos que utilizan IOPS provisionadas

La siguiente política deniega permiso explícitamente para crear instancias de base de datos que utilizan las clases de instancia de base de datos `r3.8xlarge` y `m4.10xlarge`, que son las clases de instancia de base de datos más costosas y de mayor tamaño. Esta política también evita que los usuarios creen instancias de base de datos que utilizan IOPS provisionadas, las cuales tienen un costo adicional.

Al denegarse permiso explícitamente se sustituye a cualquier otro permiso concedido. Esto garantiza que las identidades no obtengan accidentalmente permisos que el usuario no desee conceder nunca.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
```

```
                "db.m4.10xlarge"
             ]
          }
       }
    },
    {
       "Sid": "DenyPIOPSCreate",
       "Effect": "Deny",
       "Action": "rds:CreateDBInstance",
       "Resource": "*",
       "Condition": {
          "NumericNotEquals": {
             "rds:Piops": "0"
          }
       }
    }
  ]
}
```

Ejemplo 3: limitar el conjunto de claves y valores de etiquetas que se pueden usar para etiquetar un recurso

En la siguiente política se usa una clave condicional de RDS y permite añadir una etiqueta con la clave `stage` a un recurso con los valores `test`, `qa` y `production`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagEdits",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "arn:aws:rds:us-east-1:123456789012:db:db-123456",
      "Condition": {
        "StringEquals": {
```

```
          "rds:req-tag/stage": [
            "test",
            "qa",
            "production"
          ]
        }
      }
    }
  ]
}
```

Especificación de condiciones: uso de etiquetas personalizadas

Amazon Aurora admite la especificación de condiciones en una política de IAM que utiliza etiquetas personalizadas.

Por ejemplo, suponga que añade una etiqueta con el nombre `environment` a sus instancias de base de datos con valores como `beta`, `staging`, `production`, etc. Si lo hace, puede crear una política que restrinja a ciertos usuarios en instancias de bases de datos basándose en el valor de la etiqueta `environment`.

> **Note**
>
> Los identificadores de etiquetas personalizados distinguen entre mayúsculas y minúsculas.

En la tabla siguiente, se enumeran los identificadores de etiqueta de RDS que puede usar en un elemento `Condition`.

Identificador de etiqueta de RDS	Se aplica a
`db-tag`	Instancias de base de datos, incluidas las réplicas de lectura
`snapshot-tag`	Instantáneas de base de datos
`ri-tag`	Instancias de base de datos reservadas
`og-tag`	Grupos de opciones de base de datos

Identificador de etiqueta de RDS	Se aplica a
`pg-tag`	Grupos de parámetros de base de datos
`subgrp-tag`	Grupos de subred de base de datos
`es-tag`	Suscripciones de eventos
`cluster-tag`	Clústeres de base de datos
`cluster-pg-tag`	Grupos de parámetros de clúster de bases de datos
`cluster-snapshot-tag`	Instantáneas de clúster de bases de datos

La sintaxis de una condición de etiqueta personalizada es la siguiente:

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name":
["value"]} }
```

Por ejemplo, el elemento `Condition` siguiente se aplica a instancias de base de datos con una etiqueta llamada `environment` y un valor de etiqueta `production`.

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

Para obtener información acerca de la creación etiquetas, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS.

> **Important**
>
> Si administra el acceso a sus recursos de RDS mediante el etiquetado, recomendamos que proteja el acceso a las etiquetas. Puede administrar el acceso a etiquetas creando políticas para las acciones `AddTagsToResource` y `RemoveTagsFromResource`. Por ejemplo, la política siguiente deniega a los usuarios la posibilidad de agregar o quitar etiquetas para todos los recursos. A continuación, puede crear políticas para permitir que usuarios específicos agreguen o quiten etiquetas.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

Para ver una lista de las acciones de Aurora, consulte Acciones definidas por Amazon RDS en la referencia de autorizaciones de servicio.

Políticas de ejemplo: uso de etiquetas personalizadas

Los siguientes ejemplos muestran cómo puede usar etiquetas personalizadas en las políticas de permisos de IAM para Amazon Aurora. Para obtener más información sobre cómo agregar etiquetas a un recurso de Amazon Aurora, consulte Nombres de recursos de Amazon (ARN) en Amazon RDS.

> **Note**
>
> Todos los ejemplos utilizan la región us-west-2 y contienen identificadores de cuenta ficticios.

Ejemplo 1: conceda permiso para acciones en un recurso con una etiqueta específica con dos valores diferentes.

La siguiente política da permiso para aplicar la operación de API `CreateDBSnapshot` en instancias de base de datos con la etiqueta `stage` establecida en `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La siguiente política da permiso para aplicar la operación de API `ModifyDBInstance` en instancias de base de datos con la etiqueta `stage` establecida en `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
          "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
            ]
       },
       {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
               "rds:db-tag/stage":[
                  "development",
                  "test"
                  ]
               }
            }
       }
    ]
}
```

Ejemplo 2: deniegue explícitamente permiso para crear una instancia de base de datos que utilice grupos de parámetros de base de datos especificados.

La siguiente política deniega explícitamente permiso para crear una instancia de base de datos que utilice grupos de parámetros de base de datos con valores de etiqueta específicos. Podría aplicar esta política si necesita que se utilice siempre un grupo de parámetros de base de datos específico, creado por el cliente, al crear instancias de base de datos. Las políticas que utilizan `Deny` suelen aplicarse para restringir el acceso concedido por una política más amplia.

Al denegarse permiso explícitamente se sustituye a cualquier otro permiso concedido. Esto garantiza que las identidades no obtengan accidentalmente permisos que el usuario no desee conceder nunca.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

Ejemplo 3: conceda permiso para acciones en una instancia de base de datos con un nombre de instancia cuyo prefijo sea un nombre de usuario.

La siguiente política da permiso para llamar a cualquier API (salvo `AddTagsToResource` o `RemoveTagsFromResource`) en una instancia de base de datos cuyo prefijo sea un nombre de usuario y que tenga una etiqueta llamada `stage` igual a `devo` o que no tenga ninguna etiqueta llamada `stage`.

La línea `Resource` en la política identifica un recurso por su nombre de recurso de Amazon (ARN). Para obtener más información sobre el uso de ARN con recursos de Amazon Aurora, consulte Nombres de recursos de Amazon (ARN) en Amazon RDS.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

Concesión de permisos para etiquetar recursos de Aurora durante la creación

Algunas operaciones de la API de RDS le permiten especificar etiquetas durante la creación del recurso. Puede utilizar etiquetas de recursos para implementar el control basado en atributos (ABAC). Para obtener más información, consulte ¿Qué es ABAC para AWS? y Control del acceso a los recursos de AWS mediante etiquetas.

Para permitir que los usuarios etiqueten los recursos durante su creación, es preciso que tengan permisos para utilizar la acción que crea el recurso, por ejemplo `rds:CreateDBCluster`. Si se especifican etiquetas en la acción de creación, RDS realiza una autorización adicional en la acción `rds:AddTagsToResource` para verificar que los usuarios tengan permisos para crear etiquetas. Por lo tanto, los usuarios también deben tener permisos explícitos para usar la acción `rds:AddTagsToResource`.

En la definición de la política de IAM para la acción `rds:AddTagsToResource`, puede utilizar la clave de condición `aws:RequestTag` para requerir etiquetas en una solicitud de etiquetado de un recurso.

Por ejemplo, la siguiente política permite a los usuarios crear instancias de base de datos y aplicar etiquetas durante la creación de instancias de base de datos, pero solo con claves de etiqueta específicas (`environment` o `project`):

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBInstance"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "aws:RequestTag/environment": ["production", "development"],
                   "aws:RequestTag/project": ["dataanalytics", "webapp"]
               },
               "ForAllValues:StringEquals": {
                   "aws:TagKeys": ["environment", "project"]
               }
           }
       }
   ]
}
```

Esta política deniega cualquier solicitud de creación de instancias de base de datos que incluya etiquetas distintas de las etiquetas `environment` o `project`, o que no especifique ninguna de estas etiquetas. Además, los usuarios deben especificar valores para las etiquetas que coincidan con los valores permitidos en la política.

La siguiente política permite a los usuarios crear clústeres de base de datos y aplicar cualquier etiqueta durante la creación, excepto la etiqueta `environment=prod`:

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBCluster"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringNotEquals": {
                   "aws:RequestTag/environment": "prod"
               }
           }
       }
   ]
}
```

Acciones de la API de RDS compatibles para etiquetar durante la creación

Las siguientes acciones de la API de RDS admiten el etiquetado al crear un recurso. Para estas acciones, puede especificar etiquetas al crear el recurso:

- `CreateBlueGreenDeployment`
- `CreateCustomDBEngineVersion`
- `CreateDBCluster`
- `CreateDBClusterEndpoint`
- `CreateDBClusterParameterGroup`
- `CreateDBClusterSnapshot`
- `CreateDBInstance`

- CreateDBInstanceReadReplica
- CreateDBParameterGroup
- CreateDBProxy
- CreateDBProxyEndpoint
- CreateDBSecurityGroup
- CreateDBShardGroup
- CreateDBSnapshot
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- CreateIntegration
- CreateOptionGroup
- CreateTenantDatabase
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CopyDBParameterGroup
- CopyDBSnapshot
- CopyOptionGroup
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime
- PurchaseReservedDBInstancesOffering

Si utiliza la AWS CLI o la API para crear un recurso con etiquetas, el parámetro `Tags` se utiliza para aplicar etiquetas a los recursos durante la creación.

En el caso de estas acciones de la API, si se produce un error al etiquetar, el recurso no se crea y se produce un error en la solicitud. Esto garantiza que los recursos se creen con etiquetas o, de lo contrario, no se creen sin las etiquetas correspondientes.

Políticas administradas por AWS para Amazon RDS

Para añadir permisos a conjuntos de permisos y roles, es más fácil utilizar políticas administradas de AWS que escribirlas uno mismo. Se necesita tiempo y experiencia para crear políticas administradas por el cliente de IAM que le brinden a su equipo solo los permisos necesarios. Para comenzar rápidamente, puede utilizar nuestras políticas administradas de AWS. Estas políticas cubren casos de uso comunes y están disponibles en su Cuenta de AWS. Para obtener más información acerca de las políticas administradas de AWS, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Los Servicios de AWS mantienen y actualizan las políticas administradas por AWS. No puede cambiar los permisos en las políticas administradas de AWS. En ocasiones, los servicios agregan permisos adicionales a una política administrada por AWS para admitir características nuevas. Este tipo de actualización afecta a todas las identidades (conjuntos de permisos y roles) donde se asocia la política. Es más probable que los servicios actualicen una política administrada por AWScuando se lanza una nueva característica o cuando se ponen a disposición nuevas operaciones. Los servicios no quitan permisos de una política administrada por AWS, por lo que las actualizaciones de políticas no deterioran los permisos existentes.

Además, AWS admite políticas administradas para funciones de trabajo que abarcan varios servicios. Por ejemplo, la política administrada por `ReadOnlyAccess` AWS proporciona acceso de solo lectura a todos los recursos y a Servicios de AWS. Cuando un servicio lanza una nueva característica, AWS agrega permisos de solo lectura para las operaciones y los recursos nuevos. Para obtener una lista y descripciones de las políticas de funciones de trabajo, consulte Políticas administradas de AWS para funciones de trabajo en la Guía del usuario de IAM.

Temas

- Política administrada por:AWS AmazonRDSReadOnlyAccess
- Política administrada por:AWS AmazonRDSFullAccess
- Política administrada por:AWS AmazonRDSDataFullAccess
- Política administrada por:AWS AmazonRDSEnhancedMonitoringRole
- Política administrada por:AWS AmazonRDSPerformanceInsightsReadOnly
- Política administrada por AWS: AmazonRDSPerformanceInsightsFullAccess
- Política administrada por:AWS AmazonRDSDirectoryServiceAccess
- Política administrada por:AWS AmazonRDSServiceRolePolicy
- Política administrada de:AWS AmazonRDSPreviewServiceRolePolicy

- Política administrada de:AWS AmazonRDSBetaServiceRolePolicy

Política administrada por:AWS AmazonRDSReadOnlyAccess

Esta política permite acceso de solo lectura a Amazon RDS mediante la Consola de administración de AWS.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `rds`: permite a las entidades principales describir los recursos de Amazon RDS y enumerar las etiquetas de los recursos de Amazon RDS.
- `cloudwatch`: permite a las entidades principales obtener estadísticas de métricas de Amazon CloudWatch.
- `ec2`: permite a las entidades principales describir las zonas de disponibilidad y los recursos de red.
- `logs`: permite a las entidades principales describir los flujos de registro de CloudWatch Logs de los grupos de registros y obtener eventos de registro de CloudWatch Logs.
- `devops-guru`: permite a las entidades principales describir los recursos que incluyen la cobertura de Amazon DevOps Guru, que se especifica mediante nombres de pila o etiquetas de recursos de CloudFormation.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSReadOnlyAccess en la Guía de referencia de políticas administradas de AWS.

Política administrada por:AWS AmazonRDSFullAccess

Esta política proporciona acceso completo a Amazon RDS mediante la Consola de administración de AWS.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `rds`: permite a las entidades principales obtener acceso completo a Amazon RDS.
- `application-autoscaling`: permite a las entidades principales describir y administrar los objetivos y las políticas de escalado de Application Auto Scaling.

- `cloudwatch`: permite a las entidades principales obtener estadísticas métricas de CloudWatch y administrar alarmas de CloudWatch.
- `ec2`: permite a las entidades principales describir las zonas de disponibilidad y los recursos de red.
- `logs`: permite a las entidades principales describir los flujos de registro de CloudWatch Logs de los grupos de registros y obtener eventos de registro de CloudWatch Logs.
- `outposts`: permite a las entidades principales obtener tipos de instancias AWS Outposts.
- `pi`: permite a las entidades principales obtener métricas de Información sobre rendimiento.
- `sns`: permite a las entidades principales acceder a las suscripciones y temas de Amazon Simple Notification Service (Amazon SNS), y publicar mensajes de Amazon SNS.
- `devops-guru`: permite a las entidades principales describir los recursos que incluyen la cobertura de Amazon DevOps Guru, que se especifica mediante nombres de pila o etiquetas de recursos de CloudFormation.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSFullAccess en la Guía de referencia de políticas administradas de AWS.

Política administrada por:AWS AmazonRDSDataFullAccess

Esta política permite tener acceso completo para utilizar la API de datos y el editor de consultas en los clústeres de Aurora Serverless en una Cuenta de AWS determinada. Esta política permite a la Cuenta de AWS obtener el valor de un secreto de AWS Secrets Manager.

Puede adjuntar la política de `AmazonRDSDataFullAccess` a las identidades de IAM.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `dbqms`: permite a las entidades principales acceder, crear, eliminar, describir y actualizar consultas. El Database Query Metadata Service (`dbqms`) es un servicio únicamente interno. Proporciona sus consultas recientes y guardadas para el editor de consultas en la Consola de administración de AWS para varios servicios de Servicios de AWS, incluido Amazon RDS.
- `rds-data`: permite a las entidades principales ejecutar instrucciones SQL en bases de datos de Aurora Serverless.
- `secretsmanager`: permite a las entidades principales obtener el valor de un secreto de AWS Secrets Manager.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSDataFullAccess en la Guía de referencia de políticas administradas de AWS.

Política administrada por:AWS AmazonRDSEnhancedMonitoringRole

Esta política proporciona acceso a registros de Amazon Cloudwatch para Supervisión mejorada de Amazon RDS.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `logs`: permite a las entidades principales crear grupos de registros y políticas de retención de CloudWatch Logs, y crear y describir flujos de registro de CloudWatch Logs de los grupos de registro. También permite a las entidades principales poner y obtener eventos de registro de CloudWatch Logs.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSEnhancedMonitoringRole en la Guía de referencia de políticas administradas de AWS.

Política administrada por:AWS AmazonRDSPerformanceInsightsReadOnly

Esta política proporciona acceso de solo lectura a Información sobre rendimiento de Amazon RDS para instancias de base de datos de Amazon RDS y clústeres de base de datos de Amazon Aurora.

Ahora, esta política incluye `Sid` (ID de instrucción) como identificador en las instrucciones de la política.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `rds`: permite a las entidades principales describir instancias de base de datos de Amazon RDS y clústeres de base de datos de Amazon Aurora
- `pi`: permite a las entidades principales realizar llamadas a la API de Información sobre rendimiento de Amazon RDS y acceder a las métricas de Información sobre rendimiento.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSPerformanceInsightsReadOnly en la Guía de referencia de políticas administradas de AWS.

Política administrada por AWS: AmazonRDSPerformanceInsightsFullAccess

Esta política proporciona acceso completo a Información de rendimiento de Amazon RDS para instancias de base de datos de Amazon RDS y clústeres de base de datos de Amazon Aurora.

Ahora, esta política incluye `Sid` (ID de instrucción) como identificador en las instrucciones de la política.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `rds`: permite a las entidades principales describir instancias de base de datos de Amazon RDS y clústeres de base de datos de Amazon Aurora
- `pi`: permite a los entidades principales realizar llamadas a la API de Información de rendimiento de Amazon RDS y crear, ver y eliminar informes de análisis de rendimiento.
- `cloudwatch`: permite a las entidades principales enumerar todas las métricas de Amazon CloudWatch y obtener estadísticas y datos de las métricas.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSPerformanceInsightsFullAccess en la Guía de referencia de políticas administradas de AWS.

Política administrada por:AWS AmazonRDSDirectoryServiceAccess

Esta política permite a Amazon RDS realizar llamadas al Directory Service.

Detalles de los permisos

Esta política incluye el siguiente permiso:

- `ds`: permite a las entidades principales describir directorios y autorización de control de Directory Service a los directorios de Directory Service.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSDirectoryServiceAccess en la Guía de referencia de políticas administradas de AWS.

Política administrada por:AWS AmazonRDSServiceRolePolicy

No puede adjuntar la política `AmazonRDSServiceRolePolicy` a sus entidades de IAM. Esta política está adjunta a un rol vinculado a servicios que permite a Amazon RDS realizar acciones en su nombre. Para obtener más información, consulte Permisos de roles vinculados a servicios de Amazon Aurora.

Política administrada de:AWS AmazonRDSPreviewServiceRolePolicy

No debe adjuntar `AmazonRDSPreviewServiceRolePolicy` a sus entidades IAM. Esta política está asociada a un rol vinculado a servicios que permite que Amazon RDS llame a los servicios de AWS en nombre de sus recursos de base de datos de RDS. Para obtener más información, consulte Rol vinculado a servicios para Amazon RDS Preview.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `ec2`: permite a las entidades principales describir las zonas de disponibilidad y los recursos de red.
- `secretsmanager`: permite a las entidades principales obtener el valor de un secreto de AWS Secrets Manager.
- `cloudwatch`,:`logs` permite a Amazon RDS cargar registros y métricas de instancias de base de datos a CloudWatch a través del agente de CloudWatch.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSDataFullAccess en la Guía de referencia de políticas administradas de AWS.

Política administrada de:AWS AmazonRDSBetaServiceRolePolicy

No debe adjuntar `AmazonRDSBetaServiceRolePolicy` a sus entidades IAM. Esta política está asociada a un rol vinculado a servicios que permite que Amazon RDS llame a los servicios de AWS en nombre de sus recursos de base de datos de RDS. Para obtener más información, consulte Permisos de roles vinculados a servicios para Amazon RDS Beta.

Detalles de los permisos

Esta política incluye los permisos siguientes:

- `ec2`: permite a Amazon RDS realizar operaciones de copia de seguridad en la instancia de base de datos, lo que proporciona capacidades de restauración en un momento dado.
- `secretsmanager`: permite a Amazon RDS gestionar los secretos específicos de la instancia de base de datos creados por Amazon RDS.
- `cloudwatch`,:`logs` permite a Amazon RDS cargar registros y métricas de instancias de base de datos a CloudWatch a través del agente de CloudWatch.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSBetaServiceRolePolicy en la Guía de referencia de políticas administradas de AWS.

Actualizaciones de Amazon RDS a políticas administradas por AWS

Es posible consultar los detalles sobre las actualizaciones de las políticas administradas por AWS para Amazon RDS desde que este servicio comenzó a hacer un seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página de historial de documentos de la API de Amazon RDS.

Cambio	Descripción	Fecha
Política administrada de:AWS AmazonRDSPreviewServiceRolePolicy: actualización de una política actual	Amazon RDS ha eliminado el permiso `sns:Publish` de `AmazonRDSPreviewServiceRolePolicy` del rol vinculado a un servicio `AWSServiceRoleForRDSPreview`. Para obtener más información, consulte Política administrada de:AWS AmazonRDSPreviewServiceRolePolicy.	7 de agosto de 2024
Política administrada de:AWS AmazonRDSBetaServiceRolePolicy: actualización de una política actual	Amazon RDS ha eliminado el permiso `sns:Publish` de `AmazonRDSBetaServiceRolePolicy` del rol vinculado a un servicio `AWSServiceRoleForRDSBeta`. Para obtener más información, consulte Política administrada de:AWS AmazonRDSBetaServiceRolePolicy.	7 de agosto de 2024
Política administrada por:AWS AmazonRDSServiceRolePolicy: actualización de una política actual	Amazon RDS ha eliminado el permiso `sns:Publish` de `AmazonRDSServiceRolePolicy` del rol vinculado	2 de julio de 2024

Cambio	Descripción	Fecha
	a un servicio `AWSServiceRoleForRDS`. Para obtener más información, consulte Política administrada por:AWS AmazonRDSServiceRolePolicy.	
Políticas administradas por AWS para Amazon RDS: actualización de una política actual	Amazon RDS agregó un nuevo permiso a la `AmazonRDSCustomServiceRolePolicy` del rol vinculado al servicio `AWSServiceRoleForRDSCustom` para permitir que RDS Custom para SQL Server modifique el tipo de instancia de host de la base de datos subyacente. RDS también agregó el permiso `ec2:DescribeInstanceTypes` para obtener información sobre el tipo de instancia para el host de la base de datos. Para obtener más información, consulte Políticas administradas por AWS para Amazon RDS.	8 de abril de 2024

Cambio	Descripción	Fecha
Políticas administradas por AWS para Amazon RDS: política nueva	Amazon RDS agregó una nueva política administrada denominada `AmazonRDSCustomInstanceProfileRolePolicy` para permitir a RDS Custom realizar acciones de automatización y tareas de administración de bases de datos a través de un perfil de instancia de EC2. Para obtener más información, consulte Políticas administradas por AWS para Amazon RDS.	27 de febrero de 2024
Permisos de roles vinculados a servicios de Amazon Aurora: actualización de una política actual	Amazon RDS ha agregado nuevos ID de instrucciones a la `AmazonRDSServiceRolePolicy` del rol vinculado a un servicio `AWSServiceRoleForRDS` . Para obtener más información, consulte Permisos de roles vinculados a servicios de Amazon Aurora.	19 de enero de 2024

Cambio	Descripción	Fecha
Políticas administradas por AWS para Amazon RDS: actualización de políticas existentes	Las políticas administradas `AmazonRDSPerformanceInsightsReadOnly` y `AmazonRDSPerformanceInsightsFullAccess` incluyen ahora `Sid` (ID de instrucción) como identificador en las instrucciones de la política. Para obtener más información, consulte Política administrada por:AWS AmazonRDS PerformanceInsightsReadOnly y Política administrada por AWS: AmazonRDS PerformanceInsightsFullAccess.	23 de octubre de 2023
Políticas administradas por AWS para Amazon RDS: actualización de una política actual	Amazon RDS ha añadido nuevos permisos a la política administrada `AmazonRDSFullAccess`. Los permisos le permiten generar, ver y eliminar el informe de análisis de rendimiento durante un período de tiempo. Para obtener más información sobre la configuración de políticas de acceso para la Información de rendimiento, consulte Configuración de directivas de acceso para información sobre rendimiento	17 de agosto de 2023

Cambio	Descripción	Fecha
Políticas administradas por AWS para Amazon RDS: nueva política y actualización de la política existente	Amazon RDS ha añadido nuevos permisos a la política administrada `AmazonRDSPerformanceInsightsReadOnly` y una nueva política administrada denominada `AmazonRDSPerformanceInsightsFullAccess`. Estos permisos le permiten analizar la Información de rendimiento durante un período de tiempo, ver los resultados del análisis junto con las recomendaciones y eliminar los informes. Para obtener más información sobre la configuración de políticas de acceso para la Información de rendimiento, consulte Configuración de directivas de acceso para información sobre rendimiento	16 de agosto de 2023

Cambio	Descripción	Fecha
Políticas administradas por AWS para Amazon RDS: actualización de una política actual	Amazon RDS ha añadido el espacio de nombres de Amazon CloudWatch `ListMetrics` a `AmazonRDSFullAccess` y `AmazonRDSReadOnlyAccess`. Este espacio de nombres es necesario para que Amazon RDS publique métricas de uso de recursos específicas. Para obtener más información, consulte Overview of managing access permissions to your CloudWatch resources (Información general sobre la administración de los permisos de acceso a los recursos de CloudWatch) en la Guía del usuario de Amazon CloudWatch.	4 de abril de 2023

Cambio	Descripción	Fecha
Permisos de roles vinculados a servicios de Amazon Aurora: actualización de una política actual	Amazon RDS ha añadido nuevos permisos a la `AmazonRDSServiceRolePolicy` del rol vinculado a un servicio `AWSServiceRoleForRDS` para su integración con AWS Secrets Manager. RDS debe integrarse con Secrets Manager para administrar las contraseñas de los usuarios maestros en Secrets Manager. El secreto utiliza una convención de nomenclatura reservada y restringe las actualizaciones de los clientes. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.	22 de diciembre de 2022

Cambio	Descripción	Fecha
Políticas administradas por AWS para Amazon RDS: actualización de políticas existentes	Amazon RDS ha agregado un nuevo permiso a la política `AmazonRDSFullAccess` y las políticas administradas de `AmazonRDSReadOnlyAccess` para permitirle activar Amazon DevOps Guru en la consola de RDS. Este permiso es necesario para comprobar si DevOps Guru está activado. Para obtener más información, consulte Configuración de las políticas de acceso de IAM para DevOps Guru para RDS.	19 de diciembre de 2022
Permisos de roles vinculados a servicios de Amazon Aurora: actualización de una política actual	Amazon RDS ha añadido nuevos espacios de nombres de Amazon CloudWatch a `AmazonRDSPreviewServiceRolePolicy` para `PutMetricData` . Este espacio de nombres es necesario para que Amazon RDS publique métricas de uso de recursos. Para obtener más información, consulte Uso de claves de condición para limitar el acceso a los espacios de nombres de CloudWatch en la guía del usuario de Amazon CloudWatch.	7 de junio de 2022

Cambio	Descripción	Fecha
Permisos de roles vinculados a servicios de Amazon Aurora: actualización de una política actual	Amazon RDS ha añadido nuevos espacios de nombres de Amazon CloudWatch a `AmazonRDSBetaServiceRolePolicy` para `PutMetricData` . Este espacio de nombres es necesario para que Amazon RDS publique métricas de uso de recursos. Para obtener más información, consulte Uso de claves de condición para limitar el acceso a los espacios de nombres de CloudWatch en la guía del usuario de Amazon CloudWatch.	7 de junio de 2022

Cambio	Descripción	Fecha
Permisos de roles vinculados a servicios de Amazon Aurora: actualización de una política actual	Amazon RDS ha añadido nuevos espacios de nombres de Amazon CloudWatch a `AWSServiceRoleForRDS` para `PutMetricData` . Este espacio de nombres es necesario para que Amazon RDS publique métricas de uso de recursos. Para obtener más informaci ón, consulte Uso de claves de condición para limitar el acceso a los espacios de nombres de CloudWatch en la guía del usuario de Amazon CloudWatch.	22 de abril de 2022

Cambio	Descripción	Fecha
Políticas administradas por AWS para Amazon RDS: política nueva	Amazon RDS ha añadido una nueva política administrada llamada `AmazonRDSPerformanceInsightsReadOnly` para permitir que Amazon RDS llame a servicios de AWS en nombre de sus instancias de bases de datos. Para obtener más información sobre la configuración de políticas de acceso para la Información de rendimiento, consulte Configuración de directivas de acceso para información sobre rendimiento	10 de marzo de 2022

Cambio	Descripción	Fecha
Permisos de roles vinculados a servicios de Amazon Aurora: actualización de una política actual	Amazon RDS ha añadido nuevos espacios de nombres de Amazon CloudWatch a `AWSServiceRoleForRDS` para `PutMetricData` . Estos espacios de nombres son necesarios para Amazon DocumentDB (compatible con MongoDB) y Amazon Neptune para publicar métricas de CloudWatch. Para obtener más informaci ón, consulte Uso de claves de condición para limitar el acceso a los espacios de nombres de CloudWatch en la guía del usuario de Amazon CloudWatch.	4 de marzo de 2022
Amazon RDS ha comenzado a hacer un seguimiento de los cambios	Amazon RDS ha comenzado a realizar un seguimiento de los cambios en sus políticas administradas por AWS.	26 de octubre de 2021

Prevención de los problemas del suplente confuso entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación entre servicios puede dar lugar al problema de la sustitución confusa.

La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que pueden ayudarlo a proteger sus datos en todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta. Para obtener más información, consulte El problema del suplente confuso en la Guía del usuario de IAM.

A fin de limitar los permisos que Amazon RDS da a otro servicio para un recurso específico, le recomendamos utilizar las claves de contexto de condición global de `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos.

En algunos casos, el valor de `aws:SourceArn` no contiene el ID de la cuenta, por ejemplo, al utilizar el nombre de recurso de Amazon (ARN) para un bucket de Simple Storage Service (Amazon S3). En estos casos, asegúrese de utilizar ambas claves de contexto de condición global para limitar los permisos. En algunos casos, se utilizan las claves de contexto de condición global y el valor de `aws:SourceArn` contiene el ID de la cuenta. En estos casos, asegúrese de que el valor de `aws:SourceAccount` y la cuenta en `aws:SourceArn` utilicen el mismo ID de cuenta cuando se utilizan en la misma instrucción de política. Si quiere que solo se asocie un recurso al acceso entre servicios, utilice `aws:SourceArn`. Si quiere permitir que cualquier recurso de esa cuenta de AWS se asocie al uso entre servicios, utilice `aws:SourceAccount`.

Asegúrese de que el valor de `aws:SourceArn` sea un ARN para un tipo de recurso de Amazon RDS. Para obtener más información, consulte Nombres de recursos de Amazon (ARN) en Amazon RDS.

La forma más eficaz de protegerse contra el problema del suplente confuso es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. En algunos casos, es posible que no sepa el ARN completo del recurso o que esté especificando varios recursos. En estos casos, utilice la clave de condición de contexto global de `aws:SourceArn` con comodines (*) para las partes desconocidas del ARN. Un ejemplo es: . `arn:aws:rds:*:123456789012:*`.

En el ejemplo siguiente, se muestra cómo se pueden utilizar las claves de contexto de condición global de `aws:SourceArn` y `aws:SourceAccount` en Amazon RDS para evitar el problema del suplente confuso.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Para ver más ejemplos de las políticas que utilizan las claves de contexto de condición global de `aws:SourceArn` y `aws:SourceAccount`, consulte las siguientes secciones:

- Concesión de permisos para publicar notificaciones en un tema de Amazon SNS
- Configuración del acceso a un bucket de Amazon S3 (importación de PostgreSQL)
- Configuración del acceso a un bucket de Amazon S3 (exportación de PostgreSQL)

Autenticación de bases de datos de IAM

Puede autenticar en su clúster de bases de datos mediante la autenticación de base de datos de AWS Identity and Access Management (IAM). La autenticación de base de datos de IAM funciona con Aurora MySQL y Aurora PostgreSQL. Con este método de autenticación, no es necesario usar una contraseña al conectarse a un clúster de bases de datos. En su lugar, puede usar un token de autenticación.

Un token de autenticación es una cadena única de caracteres que genera Amazon Aurora bajo demanda. Los tokens de autenticación se generan mediante AWS Signature versión 4. Cada token tiene una vida útil de 15 minutos. No es necesario almacenar credenciales de usuario en la base de datos, ya que la autenticación se administra de forma externa mediante IAM. También puede seguir utilizando la autenticación de base de datos estándar. El token solo se utiliza para la autenticación y no afecta a la sesión después de establecerse.

La autenticación de bases de datos de IAM proporciona los siguientes beneficios:

- El tráfico de red hacia y desde la base de datos se cifra mediante Secure Socket Layer (SSL) o Transport Layer Security (TLS). Para obtener más información sobre el uso de SSL/TLS con Amazon Aurora, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.
- Puede usar IAM para administrar de forma centralizada el acceso a sus recursos de base de datos, en lugar de administrar el acceso individualmente en cada clúster de bases de datos.
- Para las aplicaciones que se ejecutan en Amazon EC2, puede usar las credenciales del perfil específicas de la instancia de EC2 para obtener acceso a su base de datos en lugar de una contraseña, para mayor seguridad.

En general, considere la posibilidad de utilizar la autenticación de base de datos de IAM cuando sus aplicaciones creen menos de 200 conexiones por segundo y no desee administrar los nombres de usuario y las contraseñas directamente en el código de la aplicación.

El controlador JDBC de Amazon Web Services (AWS) admite la autenticación de base de datos de IAM. Para obtener más información, consulte AWS IAM Authentication Plugin en el repositorio GitHub del controlador JDBC de Amazon Web Services (AWS).

El controlador de Python de Amazon Web Services (AWS) admite la autenticación de base de datos de IAM. Para obtener más información, consulte AWS IAM Authentication Plugin en el repositorio GitHub del controlador de Python de Amazon Web Services (AWS).

Consulte los siguientes temas para aprender a utilizar el proceso de configuración de IAM para la autenticación de bases de datos:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM
- Conexión a al clúster de bases de datos con la autenticación de IAM

Disponibilidad en regiones y versiones

La disponibilidad de las características varía según las versiones específicas de cada motor de base de datos de Aurora y entre Regiones de AWS. Para obtener más información sobre la disponibilidad en las versiones y las regiones de la autenticación de base de datos de IAM y Aurora, consulte Regiones y motores de base de datos Aurora admitidos para autenticación de bases de datos IAM.

Respecto de Aurora MySQL, todas las clases de instancia de base de datos admitidas son compatibles con la autenticación de bases de datos de IAM, excepto db.t2.small y db.t3.small. Para obtener más información sobre las clases de instancias de bases de datos admitidas, consulte Motores de base de datos compatibles para clases de instancia de base de datos.

Soporte de CLI y SDK

La autenticación de bases de datos de IAM está disponible para la AWS CLI y para los siguientes SDK de AWS específicos para cada lenguaje:

- AWS SDK para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWS SDK para Java
- AWS SDK para JavaScript
- AWS SDK para PHP
- AWS SDK para Python (Boto3)
- AWS SDK para Ruby

Restricciones a la autenticación de bases de datos de IAM

Si utiliza la autenticación de base de datos de IAM, se aplicarán las siguientes limitaciones:

- Actualmente, la autenticación de base de datos de IAM no admite todas las claves de contexto de condición global.

  Para obtener más información sobre las claves de condición globales, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.
- Para PostgreSQL, si se agrega el rol de IAM (`rds_iam`) a un usuario (incluido el usuario maestro de RDS), la autenticación de IAM tiene prioridad sobre la autenticación de la contraseña, por lo que el usuario debe iniciar sesión como un usuario de IAM.
- No puede utilizar un registro DNS personalizado de Route 53 en lugar del punto de conexión del clúster de base de datos para generar el token de autenticación.
- CloudWatch y CloudTrail no registran la autenticación de IAM. Estos servicios no rastrean las llamadas a la API `generate-db-auth-token` que autorizan a la función de IAM a habilitar la conexión a la base de datos.
- La autenticación de la base de datos de IAM requiere recursos informáticos en el clúster de la base de datos. Debe tener entre 300 MiB y 1000 MiB de memoria adicional en la base de datos para obtener una conectividad fiable. Para ver la memoria necesaria para la carga de trabajo, compare la columna RES de los procesos RDS en la lista de procesos de supervisión mejorada antes y después de habilitar la autenticación de bases de datos de IAM. Consulte Visualización de métricas OS en la consola de RDS.

  Si está utilizando una instancia de clase ampliable, evite quedarse sin memoria mediante la reducción de la memoria utilizada por otros parámetros como búferes y caché en la misma cantidad.
- Para Aurora MySQL, no puede utilizar la autenticación basada en contraseñas para un usuario de base de datos que configure con la autenticación de IAM.
- La autenticación de base de datos de IAM no es compatible con RDS en Outposts para ningún motor.

Recomendaciones para la autenticación de base de datos de IAM

Recomendamos lo siguiente cuando se utiliza la autenticación de base de datos de IAM:

- Utilice la autenticación de base de datos de IAM cuando la aplicación necesite menos de 200 conexiones nuevas por segundo para la autenticación de bases de datos de IAM.

  Los motores de base de datos que funcionan con Amazon Aurora no imponen ninguna restricción a los intentos de autenticación por segundo. Sin embargo, al usar la autenticación de bases de datos de IAM, su aplicación debe generar un token de autenticación. A continuación, su aplicación usa ese token para conectarse a el clúster de bases de datos. Si supera el límite máximo de nuevas conexiones por segundo, la sobrecarga adicional de la autenticación de bases de datos de IAM puede dar lugar a la limitación controlada de las conexiones.

  Considere la posibilidad de utilizar la agrupación de conexiones en sus aplicaciones para mitigar la creación constante de conexiones. Esto puede reducir la sobrecarga de la autenticación de bases de datos de IAM y permitir que las aplicaciones reutilicen las conexiones existentes. De forma alternativa, también puede utilizar RDS Proxy para estos casos de uso. RDS Proxy tiene costos adicionales. Consulte los precios de RDS Proxy.
- El tamaño de un token de autenticación de base de datos de IAM depende de muchos factores, como la cantidad de etiquetas de IAM, las políticas de servicio de IAM, las longitudes del ARN y otras propiedades de IAM y de la base de datos. El tamaño mínimo de este token suele ser de aproximadamente 1 KB, pero puede ser mayor. Dado que este token se utiliza como contraseña en la cadena de conexión a la base de datos mediante la autenticación de IAM, debe asegurarse de que ni el controlador de la base de datos (por ejemplo, ODBC) ni ninguna herramienta limiten ni trunquen de otro modo este token debido a su tamaño. Un token truncado provocará un error en la validación de autenticación que realiza la base de datos e IAM.
- Si utiliza credenciales temporales al crear un token de autenticación de base de datos de IAM, las credenciales temporales deben seguir siendo válidas cuando utilice el token de autenticación de base de datos de IAM para realizar una solicitud de conexión.

Claves de contexto de condición globales de AWS admitidas

La autenticación de base de datos de IAM no admite el siguiente subconjunto de claves de contexto de condición globales de AWS.

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`

- `aws:UserAgent`
- `aws:VpcSourceIp`

Para obtener más información, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

Activación y desactivación de la autenticación de bases de datos de IAM

De forma predeterminada, la autenticación de bases de datos de IAM está deshabilitada en los clústeres de base de datos. Puede activar o desactivar la autenticación de bases de datos de IAM mediante la Consola de administración de AWS, la AWS CLI o la API.

Puede habilitar la autenticación de base de datos de IAM cuando realice una de las siguientes acciones:

- Para crear un nuevo clúster de bases de datos con la autenticación de base de datos de IAM activada, consulte Creación de un clúster de base de datos de Amazon Aurora.
- Para modificar un clúster de bases de datos para activar la autenticación de bases de datos de IAM, consulte Modificación de un clúster de base de datos de Amazon Aurora.
- Para restaurar un clúster de bases de datos a partir de una instantánea con la autenticación de base de datos de IAM activada, consulte Restauración de una instantánea de clúster de base de datos.
- Para restaurar un clúster de bases de datos a un momento dado con la autenticación de base de datos de IAM habilitada, consulte Restauración de un clúster de base de dato a un momento indicado.

Consola

Cada flujo de trabajo de creación o modificación tiene una sección Database authentication (Autenticación de base de datos), donde puede activar o desactivar la autenticación de base de datos de IAM. En esa sección, elija Password and IAM database authentication (Autenticación de bases de datos con contraseña e IAM) para activar la autenticación de base de datos de IAM.

Para activar o desactivar la autenticación de IAM para un clúster de bases de datos existente

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, seleccione Databases (Bases de datos).

3. Elija el clúster de bases de datos que desea modificar.

> **Note**
>
> Solo puede habilitar la autenticación de IAM si todas las instancias de base de datos del clúster de bases de datos son compatibles con IAM. Compruebe los requisitos de compatibilidad en Disponibilidad en regiones y versiones.

4. Elija Modify.
5. En la sección Autenticación de base de datos, elija Autenticación de bases de datos de IAM para habilitar la autenticación de bases de datos de IAM. Elija Autenticación con contraseña o Contraseña y autenticación Kerberos para deshabilitar la autenticación de IAM.
6. También puede elegir habilitar la publicación de registros de autenticación de bases de datos de IAM en Registros de CloudWatch. En Exportaciones de registros, elija la opción registro iam-db-auth-error. La publicación de los registros en Registros de CloudWatch consume almacenamiento y se generan cargos por dicho almacenamiento. Asegúrese de eliminar los Registros de CloudWatch que ya no necesite.
7. Elija Continue.
8. Para aplicar los cambios inmediatamente, elija Immediately (Inmediatamente) en la sección Scheduling of modifications (Programación de modificaciones).
9. Elija Modify cluster (Modificar clúster).

AWS CLI

Para crear un clúster de bases de datos nuevo con la autenticación de IAM mediante la AWS CLI, use el comando `create-db-cluster`. Especifique la opción `--enable-iam-database-authentication`.

Para actualizar un clúster de bases de datos existente para que tenga o no tenga autenticación de IAM, utilice el comando AWS CLI de la `modify-db-cluster`. Especifique la opción `--enable-iam-database-authentication` o `--no-enable-iam-database-authentication`, como proceda.

> **Note**
>
> Solo puede habilitar la autenticación de IAM si todas las instancias de base de datos del clúster de bases de datos son compatibles con IAM. Compruebe los requisitos de compatibilidad en Disponibilidad en regiones y versiones.

De forma predeterminada, Aurora realiza la modificación durante el siguiente periodo de mantenimiento. Si desea invalidar esto y habilitar la autenticación de bases de datos de IAM lo antes posible, use el parámetro `--apply-immediately`.

Si restaura un clúster de bases de datos, use uno de los siguientes comandos de AWS CLI:

- `restore-db-cluster-to-point-in-time`
- `restore-db-cluster-from-db-snapshot`

De forma predeterminada, la configuración de la autenticación de bases de datos de IAM será la de la instantánea de origen. Para cambiar esta configuración, establezca la opción `--enable-iam-database-authentication` o `--no-enable-iam-database-authentication`, como proceda.

API de RDS

Para crear una instancia de base de datos nueva con la autenticación de IAM mediante la API, use la operación de la API `CreateDBCluster`. Defina el parámetro `EnableIAMDatabaseAuthentication` como `true`.

Para actualizar un clúster de bases de datos existente para que tenga o no tenga autenticación de IAM, utilice la operación de la API `ModifyDBCluster`. Establezca el parámetro `EnableIAMDatabaseAuthentication` en `true` para habilitar la autenticación de IAM o en `false` para deshabilitarla.

> **Note**
>
> Solo puede habilitar la autenticación de IAM si todas las instancias de base de datos del clúster de bases de datos son compatibles con IAM. Compruebe los requisitos de compatibilidad en Disponibilidad en regiones y versiones.

Si restaura un clúster de bases de datos, use una de las siguientes operaciones de la API:

- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`

De forma predeterminada, la configuración de la autenticación de bases de datos de IAM será la de la instantánea de origen. Para cambiar esta configuración, establezca el parámetro `EnableIAMDatabaseAuthentication` en `true` para habilitar la autenticación de IAM o `false` para deshabilitarla.

Creación y uso de una política de IAM para el acceso a bases de datos de IAM

Para permitir a un usuario o rol conectarse a su clúster de base de datos, debe crear una política de IAM. Después de eso, puede asociar la política a un conjunto de permisos o un rol.

> **Note**
>
> Para obtener más información acerca de las políticas de IAM, consulte Administración de la identidad y el acceso en Amazon Aurora.

La siguiente política de ejemplo permite a un usuario conectarse a un clúster de base de datos mediante la autenticación de bases de datos de IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:db-
ABCDEFGHIJKL01234/db_user"
            ]
        }
```

```
    ]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:cluster-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

Important

Un usuario con permisos de administrador puede acceder a los clústeres de base de datos sin permiso explícito en una política de IAM. Si desea restringir el acceso del administrador a los clústeres de de base de datos, puede crear un rol de IAM con los permisos privilegiados menores y asignarlo al administrador.

Note

No confunda el prefijo `rds-db:` con otros prefijos de operación de la API de RDS que empiezan por `rds:`. Puede usar el prefijo `rds-db:` y la acción `rds-db:connect` solo para la autenticación de bases de datos de IAM. No son válidos en ningún otro contexto.

La política de ejemplo incluye una sola instrucción con los siguientes elementos:

- `Effect`: especifique `Allow` para conceder acceso al clúster de bases de datos. Si no permite el acceso de forma explícita, el acceso se deniega de forma predeterminada.
- `Action`: especifique `rds-db:connect` para permitir las conexiones al clúster de bases de datos.
- `Resource`: especifique un nombre de recurso de Amazon (ARN) que describa una cuenta de base de datos en un clúster de bases de datos. El formato del ARN es el siguiente.

```
arn:aws:rds-db:region:account-id:dbuser:DbClusterResourceId/db-user-name
```

En este formato, reemplace lo siguiente:

- *`region`* es la región de AWS para la y clúster de bases de datos. En la política de ejemplo, la región de AWS es `us-east-2`.
- *`account-id`* es el número de cuenta de AWS para la y clúster de bases de datos. En la política de ejemplo, el número de cuenta es `1234567890`. El usuario debe estar en la misma cuenta que la cuenta de el clúster de base de datos.

  Para realizar el acceso entre cuentas, cree un rol de IAM con la política que se muestra arriba en la cuenta para el clúster de base de datos y permita que su otra cuenta asuma el rol.
- *`DbClusterResourceId`* es el identificador del clúster de bases de datos. Este identificador es único para una región de AWS y nunca cambia. En la política de ejemplo, el identificador es `cluster-ABCDEFGHIJKL01234`.

  Para buscar un ID de recurso de clúster de bases de datos en la Consola de administración de AWS de Amazon Aurora, elija el clúster de bases de datos para ver los detalles. A continuación, elija la pestaña Configuration (Configuración). El Resource ID (ID de recurso) se muestra en la sección Configuration (Configuración).

  También puede usar el comando de la AWS CLI para enumerar los identificadores e ID de recurso para todas sus y clústeres de base de datos en la región de AWS actual, como se muestra a continuación.

```
aws rds describe-db-clusters --query "DBClusters[*].
[DBClusterIdentifier,DbClusterResourceId]"
```

> **Note**
>
> Si se está conectando a una base de datos a través del proxy de RDS, especifique el ID del recurso proxy; por ejemplo, `prx-ABCDEFGHIJKL01234`. Para obtener información sobre el uso de la autenticación de bases de datos de IAM con el proxy de RDS, consulte Conexión a una base de datos mediante autenticación de IAM.

- *`db-user-name`* es el nombre de la cuenta de base de datos que se asociará a la autenticación de IAM. En la política de ejemplo, la cuenta de base de datos es `db_user`.

Puede crear otros ARN que admitan diversos patrones de acceso. La siguiente política permite el acceso a dos cuentas de base de datos diferentes en un cluster de base de datos.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
mary_roe"
         ]
      }
   ]
}
```

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/mary_roe"
         ]
      }
   ]
}
```

La siguiente política usa el carácter "*" a fin de buscar coincidencias con todas las y clústeres de base de datos y cuentas de base de datos para una cuenta de AWS y una región de AWS determinadas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:*/*"
            ]
        }
    ]
}
```

La siguiente política busca coincidencias con todas las y clústeres de base de datos para una cuenta de AWS y una región de AWS determinadas. Sin embargo, la política solo concede acceso a clústeres de base de datos que tienen una cuenta de base de datos `jane_doe`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

El usuario o el rol solo tiene acceso a las mismas bases de datos que el usuario de la base de datos. Por ejemplo, suponga que su clúster de bases de datos tiene una base de datos denominada dev y otra llamada test. Si el usuario de base de datos `jane_doe` solo tiene acceso a dev, cualquier usuario o rol que obtenga acceso a ese clúster de base de datos con el usuario `jane_doe` también tendrá acceso únicamente a dev. Esta restricción del acceso también se aplica a otros objetos de la base de datos tales como tablas, vistas, etc.

Un administrador debe crear políticas de IAM que concedan permisos a las entidades para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe asociar esas políticas a los conjuntos de permisos o roles que necesiten esos permisos. Para ver algunos ejemplos de políticas, consulte Ejemplos de políticas basadas en identidad para Amazon Aurora.

Asociación de una política de IAM a un conjunto de permisos o un rol

Tras crear una política de IAM que permita la autenticación de bases de datos, es necesario asociar la política a un conjunto de permisos o un rol. Para ver un tutorial acerca de este tema, consulte Crear y asociar su primera política administrada por el cliente en la Guía del usuario de IAM.

Mientras realiza el tutorial, puede usar uno de los ejemplos de política mostrados en esta sección como punto de partida y adaptarlo a sus necesidades. Al final del tutorial, tiene un conjunto de permisos con una política asociada que puede utilizar la acción `rds-db:connect`.

> **Note**
>
> Puede asignar varios conjuntos de permisos o roles a la misma cuenta de usuario de base de datos. Por ejemplo, suponga que su política de IAM ha especificado el siguiente ARN del recurso.
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> Si adjunta la política a los usuarios Jane, Bob y Diego, cada uno de esos usuarios puede conectarse al clúster especificado de base de datos por medio de la cuenta de la base de datos de `jane_doe`.

Creación de cuentas de base de datos utilizando autenticación de IAM

Con la autenticación de bases de datos de IAM, no es necesario asignar contraseñas de la base de datos a las cuentas de usuario creadas. Si quita un usuario asignado a una cuenta de base de datos, también debe quitar la cuenta de base de datos con la instrucción `DROP USER`.

> **Note**
>
> El nombre de usuario utilizado para la autenticación de IAM debe coincidir con el nombre de usuario en la base de datos.

Temas

- Uso de la autenticación de IAM con Aurora MySQL
- Uso de la autenticación de IAM con Aurora PostgreSQL

Uso de la autenticación de IAM con Aurora MySQL

Con Aurora MySQL, `AWSAuthenticationPlugin` gestiona la autenticación. Se trata de un complemento proporcionado por AWS que funciona perfectamente con IAM para autenticar a sus usuarios. Conecte al clúster de de base de datos como usuario maestro o como usuario diferente que pueda crear usuarios y conceder privilegios. Tras la conexión, lance la instrucción `CREATE USER`, tal como se muestra en el siguiente ejemplo.

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

La cláusula `IDENTIFIED WITH` permite que Aurora MySQL usen `AWSAuthenticationPlugin` para autenticar la cuenta de base de datos (`jane_doe`). La cláusula de `AS 'RDS'` hace referencia al método de autenticación. Asegúrese de que el nombre de usuario de la base de datos especificado sea igual a un recurso de la política de IAM para el acceso a la base de datos de IAM. Para obtener más información, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM.

> **Note**
>
> Si ve el siguiente mensaje, significa que el complemento proporcionado por AWS no está disponible para la y clúster de bases de datos actual.
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
> Para identificar este error, verifique que usa una configuración admitida y que ha habilitado la autenticación de bases de datos de IAM en su clúster de bases de datos. Para obtener más información, consulte Disponibilidad en regiones y versiones y Activación y desactivación de la autenticación de bases de datos de IAM.

Después de crear una cuenta mediante `AWSAuthenticationPlugin`, puede administrarla de la misma forma que otras cuentas de base de datos. Por ejemplo, puede modificar los privilegios de cuenta con las instrucciones `GRANT` y `REVOKE`, o bien modificar diversos atributos de cuenta con la instrucción `ALTER USER`.

El tráfico de la red de la base de datos se cifra mediante SSL/TLS cuando se utiliza IAM. Para permitir las conexiones SSL, modifique la cuenta de usuario con el siguiente comando.

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

Uso de la autenticación de IAM con Aurora PostgreSQL

Para usar la autenticación de IAM con Aurora PostgreSQL, conéctese al clúster de base de datos como usuario maestro o como usuario diferente que pueda crear usuarios y conceder privilegios. Tras la conexión, cree usuarios de base de datos y, a continuación, concédales el rol `rds_iam` tal como se muestra en el siguiente ejemplo.

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

Asegúrese de que el nombre de usuario de la base de datos especificado sea igual a un recurso de la política de IAM para el acceso a la base de datos de IAM. Para obtener más información, consulte Creación y uso de una política de IAM para el acceso a bases de datos de IAM. Debe conceder el rol `rds_iam` para utilizar la autenticación de IAM. También puede utilizar suscripciones anidadas o concesiones indirectas del rol.

Tenga en cuenta que un usuario de base de datos de PostgreSQL puede usar una autenticación de IAM o Kerberos, pero no ambas, por lo que este usuario tampoco puede tener el rol `rds_ad`. Esto se aplica también a las membresías anidadas. Para obtener más información, consulte Paso 7: crear usuarios de PostgreSQL para las entidades principales de Kerberos .

Conexión a al clúster de bases de datos con la autenticación de IAM

Con la autenticación de bases de datos de IAM, puede usar un token de autenticación al conectarse a su clúster de bases de datos. Un token de autenticación es una cadena de caracteres que usa en lugar de una contraseña. Después de generar un token de autenticación, será válido durante 15 minutos antes de caducar. Si intenta conectarse mediante un token caducado, la solicitud de conexión se deniega.

Todos los tokens de autenticación deben ir acompañados de una firma válida, mediante AWS Signature versión 4. (Para obtener más información, consulte Proceso de firma Signature Version 4 en la Referencia general de AWS.). AWS CLI y un SDK de AWS, como AWS SDK para Java o AWS SDK para Python (Boto3), pueden firmar automáticamente cada token que cree.

Puede utilizar un token de autenticación cuando se conecte a Amazon Aurora desde otro servicio de AWS, como AWS Lambda. Al utilizar un token, puede evitar introducir una contraseña en el código. De forma opcional, puede usar un SDK de AWS para crear y firmar mediante programación un token de autenticación.

Una vez que tenga un token de autenticación de IAM firmado, podrá conectarse a un clúster de bases de datos de Aurora. A continuación, puede aprender cómo hacer esto mediante una herramienta de línea de comandos o un SDK de AWS, como AWS SDK para Java o AWS SDK para Python (Boto3).

Para obtener más información, consulte las siguientes entradas del blog:

- Use IAM authentication to connect with SQL Workbench/J to Aurora MySQL or Amazon RDS for MySQL
- Using IAM authentication to connect with pgAdmin Amazon Aurora PostgreSQL or Amazon RDS para PostgreSQL (Usar la autenticación de IAM para conectar pgAdmin con Amazon Aurora PostgreSQL o Amazon RDS para PostgreSQL)

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM

Temas

- Conexión a su clúster de de base de datos mediante la autenticación IAM con los controladores de AWS
- Conexión a su clúster de bases de datos con autenticación de IAM desde la línea de comandos: AWS CLI y cliente de MySQL
- Conexión a su clúster de bases de datos desde la línea de comandos: AWS CLI y psql Client
- Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para .NET
- Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para Go
- Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para Java

- Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para Python (Boto3)

Conexión a su clúster de de base de datos mediante la autenticación IAM con los controladores de AWS

El conjunto de controladores de AWS se ha diseñado para permitir tiempos de transición y conmutación por error más rápidos y autenticarse con AWS Secrets Manager, AWS Identity and Access Management (IAM) e identidad federada. Los controladores de AWS se basan en la supervisión del estado del clúster de base de datos y en el conocimiento de la topología del clúster para determinar quién es el nuevo escritor. Este enfoque reduce los tiempos de transición y conmutación por error a segundos de un solo dígito, en comparación con las decenas de segundos de los controladores de código abierto.

Para obtener más información sobre los controladores de AWS, consulte el controlador de idioma correspondiente a su clúster de base de datos de Aurora MySQL o Aurora PostgreSQL.

Conexión a su clúster de bases de datos con autenticación de IAM desde la línea de comandos: AWS CLI y cliente de MySQL

Puede conectarse desde la línea de comando a un clúster de bases de datos de Auroracon AWS CLI y la herramienta de línea de comandos de `mysql` como se describe a continuación.

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM

> **Note**
>
> Para obtener información sobre cómo conectarse a la base de datos mediante SQL Workbench/J con la autenticación IAM, consulte la publicación de blog Use IAM authentication to connect with SQL Workbench/J to Aurora MySQL or Amazon RDS for MySQL.

Temas

- Generación de un token de autenticación de IAM
- Conexión a su clúster de bases de datos

Generación de un token de autenticación de IAM

En el siguiente ejemplo se muestra cómo obtener un token de autenticación firmado mediante la AWS CLI.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
   --port 3306 \
   --region us-west-2 \
   --username jane_doe
```

En el ejemplo, los parámetros son los siguientes:

- `--hostname`: el nombre de host del clúster de bases de datos a los que desea obtener acceso.
- `--port`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `--region`: la región de AWS en la que se ejecuta el clúster de bases de datos
- `--username`: la cuenta de base de datos a la que desea acceder.

Los primeros caracteres del token tienen un aspecto similar al siguiente.

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión de el clúster de base de datos para generar el token de autenticación.

Conexión a su clúster de bases de datos

El formato general para conectarse se muestra a continuación.

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

Los parámetros son los siguientes:

- `--host`: el nombre de host del clúster de bases de datos a los que desea obtener acceso.
- `--port`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `--ssl-ca`: la ruta completa al archivo de certificado SSL que contiene la clave pública

  Para obtener más información, consulte Conexiones TLS a clústeres de base de datos de Aurora MySQL.

  Para descargar un certificado SSL, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.
- `--enable-cleartext-plugin`: un valor que especifica que `AWSAuthenticationPlugin` debe usarse para esta conexión.

  Si está utilizando un cliente MariaDB, la opción `--enable-cleartext-plugin` no es necesaria.
- `--user`: la cuenta de base de datos a la que desea acceder.
- `--password`: un token de autenticación de IAM firmado.

El token de autenticación consta de varios cientos de caracteres. Puede ser difícil de tratar en la línea de comando. Una forma de solucionar esto es guardar el token en una variable de entorno y, a continuación, usar esa variable al conectarse. En el siguiente ejemplo se muestra una forma de realizar esta alternativa. En el ejemplo, */sample_dir/* es la ruta completa al archivo de certificado SSL que contiene la clave pública.

```
RDSHOST="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

Al conectarse mediante `AWSAuthenticationPlugin`, la conexión está protegida mediante SSL. Para verificar esto, escriba lo siguiente en el símbolo del sistema `mysql>`.

```
show status like 'Ssl%';
```

En las siguientes líneas de la salida aparecen más detalles.

```
+---------------+-------------+
| Variable_name | Value

                                                                    |

+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                    |

| ...           | ...
| Ssl_version   | TLSv1.1

                                                                    |

| ...           | ...
+-----------------------------+
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Conexión a su clúster de bases de datos desde la línea de comandos: AWS CLI y psql Client

Puede conectarse desde la línea de comando a un clúster de bases de datos de Aurora PostgreSQL con AWS CLI y la herramienta de línea de comandos psql como se describe a continuación.

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM

> **Note**
>
> Para obtener información acerca de cómo conectarse a la base de datos mediante pgAdmin con la autenticación IAM, consulte la entrada de blog Use IAM authentication to connect with pgAdmin to Amazon Aurora PostgreSQL or Amazon RDS for PostgreSQL (Usar la autenticación de IAM para conectarse con pgAdmin a Amazon Aurora PostgreSQL o Amazon RDS for PostgreSQL).

Temas

- Generación de un token de autenticación de IAM
- Conexión a un clúster de Aurora PostgreSQL

Generación de un token de autenticación de IAM

El token de autenticación consta de varios cientos de caracteres por que puede ser difícil de tratar en la línea de comando. Una forma de solucionar esto es guardar el token en una variable de entorno y, a continuación, usar esa variable al conectarse. En el siguiente ejemplo se muestra cómo usar la AWS CLI para obtener un token de autenticación firmado mediante el comando `generate-db-auth-token` y almacenarlo en una variable de entorno `PGPASSWORD`.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

En el ejemplo, los parámetros para el comando `generate-db-auth-token` son los siguientes:

- `--hostname`: el nombre de host del clúster (punto de enlace del clúster) de base de datos a los que desea obtener acceso.
- `--port`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `--region`: la región de AWS en la que se ejecuta el clúster de bases de datos
- `--username`: la cuenta de base de datos a la que desea acceder.

Los primeros caracteres del token generado tienen un aspecto similar al siguiente.

```
mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión de el clúster de base de datos para generar el token de autenticación.

Conexión a un clúster de Aurora PostgreSQL

El formato general para usar psql para conectarse se muestra a continuación.

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

Los parámetros son los siguientes:

- `host`: el nombre de host del clúster (punto de enlace del clúster) de base de datos a los que desea obtener acceso.
- `port`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `sslmode`: el modo de SSL que se debe utilizar.

  Cuando se utiliza `sslmode=verify-full`, la conexión SSL verifica el punto de conexión del clúster de bases de datos con respecto al punto de enlace del certificado SSL.
- `sslrootcert`: la ruta completa al archivo de certificado SSL que contiene la clave pública

  Para obtener más información, consulte Protección de los datos de Aurora PostgreSQL con SSL/TLS.

  Para descargar un certificado SSL, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.
- `dbname`: la base de datos a la que desea obtener acceso.
- `user`: la cuenta de base de datos a la que desea acceder.
- `password`: un token de autenticación de IAM firmado.

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión de el clúster de base de datos para generar el token de autenticación.

El siguiente ejemplo muestra el uso de psql para conectarse. En el ejemplo, psql utiliza la variable de entorno `RDSHOST` para el host y la variable de entorno `PGPASSWORD` para el token generado. Además, *`/sample_dir/`* es la ruta completa al archivo de certificado SSL que contiene la clave pública.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"

psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para .NET

Puede conectarse a un clúster de bases de datos de Aurora MySQL o Aurora PostgreSQL con el AWS SDK para .NET como se describe a continuación.

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM

Ejemplos

En los siguientes ejemplos de código, se muestra cómo se genera un token de autenticación y cómo se utiliza para conectarse a un clúster de bases de datos.

Para ejecutar este ejemplo de código, necesita AWS SDK para .NET, que se encuentra en el sitio de AWS. Los paquetes `AWSSDK.CORE` y `AWSSDK.RDS` son necesarios. Para conectarse a un clúster de base de datos, use el conector de base de datos .NET para el motor de base de datos, como MySqlConnector para MariaDB o MySQL, o Npgsql para PostgreSQL.

Este código se conecta a un clúster de bases de datos de Aurora MySQL. Modifique los valores de las siguientes variables según sea necesario:

- `server`: el punto de enlace del clúster de bases de datos que desea acceder
- `user`: la cuenta de base de datos a la que desea acceder.
- `database`: la base de datos a la que desea obtener acceso.
- `port`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `SslMode`: el modo de SSL que se debe utilizar.

  Cuando se utiliza `SslMode=Required`, la conexión SSL verifica el punto de conexión del clúster de bases de datos con respecto al punto de enlace del certificado SSL.
- `SslCa`: la ruta completa al certificado SSL de Amazon Aurora

  Para descargar un certificado, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión del clúster de base de datos para generar el token de autenticación.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
```

```
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new
 MySqlConnection($"server=mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);

      // Execute a query
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
      conn.Close();
    }
  }
}
```

Este código se conecta a un clúster de bases de datos de Aurora PostgreSQL.

Modifique los valores de las siguientes variables según sea necesario:

- `Server`: el punto de enlace del clúster de bases de datos que desea acceder
- `User ID`: la cuenta de base de datos a la que desea acceder.
- `Database`: la base de datos a la que desea obtener acceso.
- `Port`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `SSL Mode`: el modo de SSL que se debe utilizar.

  Cuando se utiliza `SSL Mode=Required`, la conexión SSL verifica el punto de conexión del clúster de bases de datos con respecto al punto de enlace del certificado SSL.

- `Root Certificate`: la ruta completa al certificado SSL de Amazon Aurora

  Para descargar un certificado, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión del clúster de base de datos para generar el token de autenticación.

```
using System;
using Npgsql;
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;User Id=jane_doe;Password={pwd};Database=mydb;SSL
 Mode=Require;Root Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
```

```
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para Go

Puede conectarse a un clúster de bases de datos de Aurora MySQL o Aurora PostgreSQL con el AWS SDK para Go como se describe a continuación.

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM

Ejemplos

Para ejecutar estos ejemplos de código, necesita AWS SDK para Go, que se encuentra en el sitio de AWS.

Modifique los valores de las siguientes variables según sea necesario:

- `dbName`: la base de datos a la que desea obtener acceso.
- `dbUser`: la cuenta de base de datos a la que desea acceder.
- `dbHost`: el punto de enlace del clúster de bases de datos que desea acceder

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión de el clúster de base de datos para generar el token de autenticación.

- `dbPort`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `region`: la región de AWS en la que se ejecuta el clúster de bases de datos

Además, debe asegurarse de que las bibliotecas importadas en el código de muestra existen en el sistema.

> **Important**
>
> En los ejemplos de esta sección se utiliza el código siguiente para proporcionar credenciales que tienen acceso a una base de datos desde un entorno local:
> `creds := credentials.NewEnvCredentials()`
> Si accede a una base de datos desde un servicio de AWS, como Amazon EC2 o Amazon ECS, puede reemplazar el código por el siguiente código:
> `sess := session.Must(session.NewSession())`
> `creds := sess.Config.Credentials`
> Si realiza este cambio, asegúrese de agregar la siguiente importación:
> `"github.com/aws/aws-sdk-go/aws/session"`

Temas

- Conexión mediante la autenticación de IAM y el V2 AWS SDK para Go
- Conexión mediante la autenticación de IAM y el V1 AWS SDK para Go

Conexión mediante la autenticación de IAM y el V2 AWS SDK para Go

Se puede conectar a un de instancia de base de datos mediante la autenticación de IAM y el V2AWS SDK para Go.

En los siguientes ejemplos de código, se muestra cómo se genera un token de autenticación y cómo se utiliza para conectarse a un clúster de bases de datos.

Este código se conecta a un clúster de bases de datos de Aurora MySQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }
```

```
    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Este código se conecta a un clúster de bases de datos de Aurora PostgreSQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
```

```
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Conexión mediante la autenticación de IAM y el V1 AWS SDK para Go

Conexión a un clúster de bases de datos mediante la autenticación de IAM y el V1 AWS SDK para Go

En los siguientes ejemplos de código, se muestra cómo se genera un token de autenticación y cómo se utiliza para conectarse a un clúster de bases de datos.

Este código se conecta a un clúster de bases de datos de Aurora MySQL.

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
```

```
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Este código se conecta a un clúster de bases de datos de Aurora PostgreSQL.

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
```

```
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para Java

Puede conectarse a un clúster de bases de datos de Aurora MySQL o Aurora PostgreSQL con el AWS SDK para Java como se describe a continuación.

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM
- Configurar el SDK de AWS para Java

Para ver ejemplos de cómo usar el SDK para Java 2.x, consulte Ejemplos de Amazon RDS que utilizan SDK para Java 2.x. También puede utilizar el contenedor JDBC avanzado de AWS; consulte la documentación del contenedor JDBC avanzado de AWS.

Temas

- Generación de un token de autenticación de IAM
- Creación manual de un token de autenticación de IAM
- Conexión a su clúster de bases de datos

Generación de un token de autenticación de IAM

Si escribe programas mediante AWS SDK para Java, puede obtener un token de autenticación firmado mediante la clase `RdsIamAuthTokenGenerator`. El uso de esta clase requiere que proporcione las credenciales de AWS. Para hacer esto, puede crear una instancia de la clase `DefaultAWSCredentialsProviderChain`. `DefaultAWSCredentialsProviderChain` usa la primera clave de acceso y clave secreta de AWS que encuentra en la cadena predeterminada de proveedores de credenciales. A fin de obtener más información acerca de las claves de acceso de AWS, consulte Administración de claves de acceso para usuarios.

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión de el clúster de base de datos para generar el token de autenticación.

Tras crear una instancia de `RdsIamAuthTokenGenerator`, puede llamar al método `getAuthToken` para obtener un token firmado. Proporcione la región de AWS el nombre de host, el número de puerto y el nombre de usuario. En el siguiente ejemplo de código se ilustra cómo hacerlo.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {
```

```
    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());

     return authToken;
    }

}
```

Creación manual de un token de autenticación de IAM

En Java, la forma más sencilla de generar un token de autenticación es usar `RdsIamAuthTokenGenerator`. Esta clase crea automáticamente un token de autenticación y, a continuación, lo firma mediante AWS Signature versión 4. Para obtener más información, consulte Proceso de firma Signature Version 4 en la Referencia general de AWS.

Sin embargo, también puede crear y firmar un token de autenticación manualmente, como se muestra en el siguiente ejemplo de código.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
```

```
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
 SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
        DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
     String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
     String awsSecretKey = creds.getCredentials().getAWSSecretKey();
        String expiryMinutes = "900";
```

```
       System.out.println("Step 1:  Create a canonical request:");
       String canonicalString = createCanonicalString(username, awsAccessKey, date,
dateTimeStamp, region, expiryMinutes, instanceName, port);
       System.out.println(canonicalString);
       System.out.println();

       System.out.println("Step 2:  Create a string to sign:");
       String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
       System.out.println(stringToSign);
       System.out.println();

       System.out.println("Step 3:  Calculate the signature:");
       String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
       System.out.println(signature);
       System.out.println();

       System.out.println("Step 4:  Add the signing info to the request");

       System.out.println(appendSignature(signature));
       System.out.println();

   }

   //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
should be in format YYYYMMDDTHHMMSSZ
   public static String createCanonicalString(String user, String accessKey, String
date, String dateTime, String region, String expiryPeriod, String hostName, String
port) throws Exception {
       canonicalQueryParameters.put("Action", action);
       canonicalQueryParameters.put("DBUser", user);
       canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
       canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
       canonicalQueryParameters.put("X-Amz-Date", dateTime);
       canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
       canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
       String canonicalQueryString = "";
       while(!canonicalQueryParameters.isEmpty()) {
           String currentQueryParameter = canonicalQueryParameters.firstKey();
           String currentQueryParameterValue =
canonicalQueryParameters.remove(currentQueryParameter);
```

```
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
 currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));

    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                              SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
```

```
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                   String dateStamp, String regionName, String
serviceName) {
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                      SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
```

```
        }
    }
}
```

Conexión a su clúster de bases de datos

El siguiente ejemplo de código muestra cómo generar un token de autenticación y, a continuación, usarlo para conectarse a un clúster que ejecuta Aurora MySQL.

Para ejecutar este ejemplo de código, necesita AWS SDK para Java, que se encuentra en el sitio de AWS. Además, necesitará lo siguiente:

- MySQL Connector/J. Este ejemplo de código se ha probado con `mysql-connector-java-5.1.33-bin.jar`.
- Un certificado intermedio para Amazon Aurora que es específico de una región de AWS. (Para obtener más información, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.) En tiempo de ejecución, el cargador de clases busca el certificado en el mismo directorio que este ejemplo de código Java, de modo que el cargador de clases pueda encontrarlo.
- Modifique los valores de las siguientes variables según sea necesario:
  - `RDS_INSTANCE_HOSTNAME`: el nombre de anfitrión del clúster de bases de datos que desea acceder.
  - `RDS_INSTANCE_PORT`: el número de puerto usado para conectarse al clúster de bases de datos de PostgreSQL.
  - `REGION_NAME`: la región de AWS en la que se ejecuta el clúster de bases de datos.
  - `DB_USER`: la cuenta de base de datos a la que desea acceder.
  - `SSL_CERTIFICATE`: un certificado de SSL para Amazon Aurora que es específico de una región de AWS.

    Para descargar un certificado para su región de AWS, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos. Coloque el certificado SSL en el mismo directorio que este archivo de programa Java, de modo que el cargador de clases pueda encontrar el certificado en tiempo de ejecución.

En este ejemplo de código, se obtienen las credenciales de AWS de la cadena predeterminada de proveedores de credenciales.

> **Note**
>
> Especifique una contraseña para `DEFAULT_KEY_STORE_PASSWORD` que no sea la que se muestra aquí como práctica recomendada de seguridad.

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
```

```
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
 Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
```

```
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
     * as the password. It also specifies that SSL verification is required.
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
     * @return
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
 AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
 AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

 .hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
 and password:
```

```
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
DEFAULT_KEY_STORE_PASSWORD);
    }

    /**
     * This method returns the path of the Key Store File needed for the SSL
verification during the IAM Database Authentication to
     * the db instance.
     * @return
     * @throws Exception
     */
    private static String createKeyStoreFile() throws Exception {
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
Exception {
```

```
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
 KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }

    /**
     * This method clears the SSL properties.
     * @throws Exception
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Conexión al clúster de base de datos mediante la autenticación de IAM y el AWS SDK para Python (Boto3)

Puede conectarse a un clúster de bases de datos de Aurora MySQL o Aurora PostgreSQL con el AWS SDK para Python (Boto3) como se describe a continuación.

Requisitos previos

A continuación, se muestran requisitos previos para conectarse al clúster de de base de datos mediante la autenticación de IAM:

- Activación y desactivación de la autenticación de bases de datos de IAM
- Creación y uso de una política de IAM para el acceso a bases de datos de IAM
- Creación de cuentas de base de datos utilizando autenticación de IAM

Además, debe asegurarse de que las bibliotecas importadas en el código de muestra existen en el sistema.

Ejemplos

Los ejemplos de código utilizan perfiles para credenciales compartidas. Para obtener información acerca de la especificación de credenciales, consulte Credenciales en la documentación de AWS SDK para Python (Boto3).

En los siguientes ejemplos de código, se muestra cómo se genera un token de autenticación y cómo se utiliza para conectarse a un clúster de bases de datos.

Para ejecutar este ejemplo de código, necesita AWS SDK para Python (Boto3), que se encuentra en el sitio de AWS.

Modifique los valores de las siguientes variables según sea necesario:

- `ENDPOINT`: el punto de enlace del clúster de bases de datos que desea acceder
- `PORT`: el número de puerto que se utiliza para conectarse al clúster de bases de datos.
- `USER`: la cuenta de base de datos a la que desea acceder.
- `REGION`: la región de AWS en la que se ejecuta el clúster de bases de datos
- `DBNAME`: la base de datos a la que desea obtener acceso.
- `SSLCERTIFICATE`: la ruta completa al certificado SSL de Amazon Aurora

  Para `ssl_ca`, especifique un certificado SSL. Para descargar un certificado SSL, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.

> **Note**
>
> No puede utilizar un registro DNS personalizado de Route 53 ni un punto de conexión personalizado de Aurora en lugar del punto de conexión del clúster de base de datos para generar el token de autenticación.

Este código se conecta a un clúster de bases de datos de Aurora MySQL.

Antes de ejecutar este código, siga las instrucciones del índice del paquete de Python para instalar el controlador PyMySQL.

```
import pymysql
import sys
import boto3
import os

ENDPOINT="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True, ssl_verify_cert=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Este código se conecta a un clúster de bases de datos de Aurora PostgreSQL.

Antes de ejecutar este código, instale `psycopg2` y siga las instrucciones en Psycopg documentation (Documentación de Psycopg).

```
import psycopg2
import sys
import boto3
import os
```

```
ENDPOINT="postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Si desea conectarse a un clúster de base de datos a través de un proxy, consulte Conexión a una base de datos mediante autenticación de IAM.

Solución de problemas de autenticación de bases de datos de IAM

A continuación, encontrará ideas para la solución de problemas comunes de autenticación de bases de datos de IAM e información sobre los registros y métricas de CloudWatch para la autenticación de base de datos de IAM.

Exportación de registros de errores de autenticación de bases de datos de IAM a Registros de CloudWatch

Los registros de errores de autenticación de bases de datos de IAM se almacenan en el host de base de datos y puede exportar estos registros a la cuenta de Registros de CloudWatch. Utilice los registros y métodos de solución de esta página para solucionar los problemas de autenticación de bases de datos de IAM.

Puede habilitar la exportación de registros a Registros de CloudWatch desde la consola, la AWS CLI y la API de RDS. Para obtener instrucciones sobre la consola, consulte Publicación de registros de base de datos en registros de Amazon Cloudwatch.

Para exportar los registros de errores de autenticación de bases de datos de IAM a Registros de CloudWatch al crear clúster de bases de datos desde la AWS CLI, utilice el siguiente comando:

```
aws rds create-db-cluster --db-cluster-identifier mydbinstance \
--region us-east-1 \
--engine postgres \
--engine-version 16 \
--master-username master \
--master-user-password password \
--publicly-accessible \
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

Para exportar los registros de errores de autenticación de bases de datos de IAM a Registros de CloudWatch al modificar clúster de bases de datos desde la AWS CLI, utilice el siguiente comando:

```
aws rds modify-db-cluster --db-instance-identifier mydbcluster \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

Para verificar si el clúster de bases de datos está exportando los registros de autenticación de base de datos de IAM a Registros de CloudWatch, compruebe si el parámetro `EnabledCloudwatchLogsExports` está establecido en `iam-db-auth-error` en la salida del comando `describe-db-instances`.

```
aws rds describe-db-cluster --region us-east-1 --db-cluster-identifier mydbcluster
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

Métricas de CloudWatch de autenticación de bases de datos de IAM

Amazon Aurora proporciona métricas casi en tiempo real sobre la autenticación de bases de datos de IAM a la cuenta de Amazon CloudWatch. En la siguiente tabla se enumeran las métricas de autenticación de bases de datos de IAM disponibles mediante CloudWatch:

Métrica	Descripción
`IamDbAuthConnectionRequests`	Número total de solicitudes de conexión realizadas con autenticación de bases de datos de IAM.
`IamDbAuthConnectionSuccess`	Número total de solicitudes de autenticación de bases de datos de IAM correctamente realizadas.
`IamDbAuthConnectionFailure`	Número total de solicitudes de autenticación de bases de datos de IAM con error.
`IamDbAuthConnectionFailureInvalidToken`	Número total de solicitudes de autenticación de bases de datos de IAM con error debido a un token no válido.
`IamDbAuthConnectionFailureInsufficientPermissions`	Número total de solicitudes de autenticación de bases de datos de IAM con error debido a políticas o permisos incorrectos.
`IamDbAuthConnectionFailureThrottling`	Número total de solicitudes de autenticación de bases de datos de IAM con error debido a la limitación de la autenticación de bases de datos de IAM.
`IamDbAuthConnectionFailureServerError`	Número total de solicitudes de autenticación de bases de datos de IAM con error debido a un error interno del servidor en la característica de autenticación de bases de datos de IAM.

Problemas y soluciones comunes de

Es posible que se encuentre con los siguientes problemas al utilizar la autenticación de bases de datos de IAM. Utilice los pasos de corrección de la tabla para resolver los problemas:

Error	Métricas	Causa	Solución

`[ERROR] Failed to authenticate the connection request for user` *`db_user`* `because the provided token is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidToken)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInvalidToken`
		El token de autenticación de bases de datos de IAM en la solicitud de conexión no es un token SigV4a válido o no tiene el formato correcto.	Compruebe la estrategia de generación de tokens en la aplicación. En algunos casos, asegúrese de que pasa el token con un formato válido. Si se trunca el token (o se aplica un formato de cadena incorrecto), el token no será válido.

`[ERROR] Failed to authenticate the connection request for user` *`db_user`* `because the token age is longer than 15 minutes. (Status Code: 400, Error Code:ExpiredToken)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInvalidToken`
		El token de autenticación de bases de datos de IAM ha caducado. Los tokens solo son válidos durante 15 minutos.	Compruebe la lógica de almacenamiento en caché de tokens o de reutilización de tokens en la aplicación. No debe reutilizar tokens que tengan más de 15 minutos.

`[ERROR] Failed to authorize the connection request for user`	`IamDbAuthConnectionFailure`	Este error es posible que se deba a las siguientes razones:	Verifique el rol o la política de IAM que está asumiendo en la aplicación. Asegúrese de que asume la

Error	Métricas	Causa	Solución
db_user because the IAM policy assumed by the caller 'arn:aws:sts::123456789012:assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform `rds-db:connect` on the DB instance. (Status Code: 403, Error Code:NotAuthorized)	IamDbAuthConnectionFailureInsufficientPermissions	• La política de IAM que asume la aplicación no autoriza la acción rds-db:connect . • Está asumiendo el rol o la política incorrectos para que *db_user* se conecte a la base de datos. • Está asumiendo la política correcta para *db_user*, pero no se está conectando a la base de datos correcta.	misma política para generar el token que para conectarse a la base de datos.
[ERROR] Failed to authorize the connection request for user *db_user* due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)	IamDbAuthConnectionFailure IamDbAuthConnectionFailureThrottling	Está realizando demasiadas solicitudes de conexión a la base de datos en un corto período de tiempo. El límite de limitación de autenticación de bases de datos de IAM es de 200 conexiones por segundo.	Reduzca la velocidad de establecimiento de nuevas conexiones con la autenticación de IAM. Considere implementar la agrupación de conexiones mediante Proxy de RDS para reutilizar las conexiones establecidas en la aplicación.

Error	Métricas	Causa	Solución

`[ERROR] Failed to authorize the connection request for user db_user due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Se ha producido un error interno al autorizar la conexión de la base de datos con la autenticación de bases de datos de IAM.	Póngase en contacto con https://aws.amazon.com/premiumsupport/ para investigar el problema.

Solución de problemas de identidades y accesos en Amazon Aurora

Utilice la información siguiente para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con Aurora e IAM.

Temas

- No tengo autorización para realizar una acción en Aurora
- No estoy autorizado a realizar la operación iam:PassRole
- Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de Aurora.

No tengo autorización para realizar una acción en Aurora

Si la Consola de administración de AWS le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

En el siguiente ejemplo, el error se produce cuando el usuario `mateojackson`, intenta utilizar la consola para ver detalles sobre un *`widget`*, pero no tiene permisos `rds:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *my-example-widget* mediante la acción `rds:GetWidget`.

No estoy autorizado a realizar la operación iam:PassRole

Si recibe un error que indica que no está autorizado para llevar a cabo la acción `iam:PassRole`, debe ponerse en contacto con su administrador para recibir ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión. Pida a la persona que actualice sus políticas de forma que pueda transferir un rol a Aurora.

Algunos servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario denominado `marymajor` intenta utilizar la consola para realizar una acción en Aurora. Sin embargo, la acción requiere que el servicio cuente con permisos otorgados por un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, Mary pide a su administrador que actualice sus políticas para que pueda realizar la acción `iam:PassRole`.

Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de Aurora.

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para obtener información acerca de si Aurora admite estas características, consulte Cómo funciona Amazon Aurora con IAM.

- Para obtener información acerca de cómo proporcionar acceso a los recursos de las cuentas de AWS de su propiedad, consulte Proporcionar acceso a un usuario de IAM a otra cuenta de AWS de la que es propietario en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos a cuentas de AWS de terceros, consulte Proporcionar acceso a cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobe cómo proporcionar acceso mediante una identidad federada, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Registro y monitoreo en Amazon Aurora

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon Aurora y las soluciones de AWS. Debe recopilar datos de supervisión de todas las partes de la solución de AWS para que pueda depurar un error multipunto de una forma más fácil si se produce. AWS proporciona varias herramientas para supervisar los recursos de Amazon Aurora y responder a posibles incidentes:

Alarmas de Amazon CloudWatch

Con las alarmas de Amazon CloudWatch, puede ver una métrica determinada durante el periodo especificado. Si la métrica supera un límite determinado, se envía una notificación a un tema de Amazon SNS o a una política de AWS Auto Scaling. Las alarmas de CloudWatch no invocan acciones simplemente porque se encuentren en determinado estado. En su lugar, el estado debe haber cambiado y debe mantenerse durante el número de periodos especificado.

AWS CloudTrailRegistros de

CloudTrail proporciona un registro de las acciones que realiza un usuario, un rol o un servicio de AWS en Amazon Aurora. CloudTrail captura todas las llamadas a la API para Amazon Aurora como eventos, incluidas las llamadas procedentes de la consola y de las llamadas de código a operaciones de la API de Amazon RDS. Mediante la información que recopila CloudTrail, puede determinar la solicitud que se envió a Amazon Aurora, la dirección IP desde la que se realizó la solicitud, quién realizó la solicitud, cuándo se realizó y detalles adicionales. Para obtener más información, consulte Supervisión de llamadas a la API de Amazon Aurora en AWS CloudTrail.

Enhanced Monitoring (Monitorización mejorada)

Amazon Aurora proporciona métricas en tiempo real para el sistema operativo (SO) en el que se ejecuta el clúster. Puede ver las métricas de el clúster de bases de datos con la consola o usar la salida JSON de la supervisión mejorada de Registros de Amazon CloudWatch en un sistema de supervisión de su elección. Para obtener más información, consulte Supervisión de las métricas del sistema operativo con Supervisión mejorada.

Amazon RDS Performance Insights

La información de rendimiento amplía las características de supervisión existentes de Amazon Aurora para ilustrar el rendimiento de la base de datos y le ayuda a analizar cualquier problema que le afecte. Con el panel de Performance Insights, puede visualizar la carga de la base de datos y filtrarla por esperas, instrucciones SQL, hosts o usuarios. Para obtener más información, consulte Monitoreo de la carga de base de datos con Performance Insights en Amazon Aurora.

Registros de la base de datos

Puede ver, descargar y monitorizar los registros de base de datos usando la Consola de administración de AWS, la AWS CLI o la API de RDS. Para obtener más información, consulte Supervisión de archivos de registro de Amazon Aurora.

Recomendaciones de Amazon Aurora

Amazon Aurora proporciona recomendaciones automatizadas de recursos de la base de datos. Estas recomendaciones proporcionan instrucciones de las prácticas recomendadas mediante el análisis de los datos de rendimiento, uso y configuración de el clúster de bases de datos. Para obtener más información, consulte Recomendaciones para Amazon Aurora.

Notificación de eventos de Amazon Aurora

Amazon Aurora usa Amazon Simple Notification Service (Amazon SNS) para proporcionar una notificación cuando se produce un evento de Amazon Aurora. Estas notificaciones pueden realizarse con cualquier método que admita Amazon SNS para una región de AWS, como un email, un mensaje de texto o una llamada a un punto de enlace HTTP. Para obtener más información, consulte Uso de notificaciones de eventos de Amazon RDS.

AWS Trusted Advisor

Trusted Advisor aprovecha las prácticas recomendadas aprendidas al atender a cientos de miles de clientes de AWS. Trusted Advisor inspecciona su entorno de AWS y realiza recomendaciones cuando surge la oportunidad de ahorrar dinero, mejorar el rendimiento y la disponibilidad del sistema o ayudar a cerrar deficiencias de seguridad. Todos los clientes de AWS tienen acceso

a cinco comprobaciones de Trusted Advisor. Los clientes con un plan de soporte Business o Enterprise pueden ver todas las comprobaciones de Trusted Advisor.

Trusted Advisor cuenta con las siguientes comprobaciones relacionadas con Amazon Aurora:

- Instancias de bases de datos inactivas de Amazon Aurora
- Riesgo de acceso de grupos de seguridad de Amazon Aurora
- Copias de seguridad de Amazon Aurora
- Multi-AZ de Amazon Aurora
- Accesibilidad de instancias de base de datos de Aurora

Para obtener más información acerca de estas comprobaciones, consulte Prácticas recomendadas de Trusted Advisor (verificaciones).

Secuencias de actividades de la base de datos

Las secuencias de actividades de la base de datos pueden proteger su bases de datos contra las amenazas internas, mediante el control del acceso de los DBA a las secuencias de actividades de la base de datos. Así, los DBA que administran la base de datos no tienen acceso a la recopilación, la transmisión, el almacenamiento y el procesamiento posterior de la secuencia de actividades de la base de datos. Las secuencias de actividad de la base de datos puede proporcionar medidas de seguridad que protejan su base de datos y cumplir los requisitos normativos y de cumplimiento. Para obtener más información, consulte Supervisión de Amazon Aurora con flujos de actividad de la base de datos.

Para obtener más información acerca de la supervisión de Aurora, consulte Supervisión de métricas en un clúster de Amazon Aurora.

Validación de la conformidad en Amazon Aurora

Auditores externos evalúan la seguridad y la conformidad de Amazon Aurora como parte de varios programas de conformidad de AWS. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros.

Para obtener una lista de AWS servicios en el ámbito de programas de cumplimiento específicos, consulte los AWS servicios en ámbito por programa de cumplimiento. Para obtener información general, consulte Programas de conformidad de AWS.

Puede descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de cumplimiento al utilizar Amazon Aurora está determinada por la confidencialidad de los datos, los objetivos de cumplimiento de la organización, y las leyes y regulaciones aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento de normas:

- Guías de inicio rápido de seguridad y conformidad: estas guías de implementación tratan consideraciones sobre arquitectura y ofrecen pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Architecting for HIPAA Security and Compliance on Amazon Web Services (Arquitectura para la seguridad y el cumplimiento de la HIPAA en Amazon Web Services): en este documento técnico, se describe cómo las empresas pueden utilizar AWS para crear aplicaciones que cumplan los requisitos de HIPAA.
- Recursos de conformidad de AWS: este conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- AWS Config: este servicio de AWS evalúa en qué medida las configuraciones de los recursos cumplen las prácticas internas, las directrices del sector y la normativa.
- AWS Security Hub CSPM: este Servicio de AWS proporciona una visión completa de su estado de seguridad en AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.

Resiliencia en Amazon Aurora

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las zonas de disponibilidad y las regiones de AWS, consulte Infraestructura global de AWS.

Además de la infraestructura global de AWS, Aurora ofrece características que lo ayudan con sus necesidades de resiliencia y copia de seguridad de los datos.

Copia de seguridad y restauración

Aurora crea copias de seguridad del volumen de clúster automáticamente y retiene los datos de restauración durante la duración del periodo de retención de copia de seguridad. Las copias de seguridad de Aurora son continuas y progresivas para que se pueda restaurar con rapidez a cualquier punto durante el periodo de retención de copia de seguridad. No se produce ningún impacto en el desempeño ni ninguna interrupción del servicio de base de datos durante la escritura de los datos de copia de seguridad. Puede especificar un periodo de retención de copia de seguridad de 1 a 35 días cuando cree o modifique un clúster de bases de datos.

Si desea conservar una copia de seguridad más allá del periodo de retención de copia de seguridad, también puede crear un snapshot de los datos del volumen de clúster. Aurora retiene los datos de restauración progresiva durante todo el periodo de retención de copia de seguridad. Por tanto, solo tiene que crear una instantánea de los datos que desee conservar después del periodo de retención de copia de seguridad. Puede crear un nuevo clúster de bases de datos a partir de la instantánea.

Puede recuperar sus datos creando un nuevo clúster de bases de datos Aurora a partir de los datos de copias de seguridad conservados por Aurora; o de una instantánea de clúster de bases de datos que haya guardado. Puede crear rápidamente una nueva copia del clúster de bases de datos a partir de los datos de la copia de seguridad de cualquier momento dado durante el periodo de retención de copia de seguridad. La naturaleza continua e incremental de las copias de seguridad de Aurora durante el periodo de retención de copia de seguridad implica que no es necesario realizar instantáneas de los datos con demasiada frecuencia para mejorar los tiempos de restauración.

Para obtener más información, consulte Copias de seguridad y restauración de un clúster de base de datos de Amazon Aurora.

Replicación

Las réplicas de Aurora son puntos de enlace independientes de un clúster de bases de datos Aurora que se utilizan preferentemente para ajustar la escala de las operaciones de lectura e incrementar la disponibilidad. Se puede distribuir un máximo de 15 réplicas de Aurora entre las distintas zonas de disponibilidad que abarca un clúster de bases de datos dentro de una región de AWS. El volumen del clúster de bases de datos consta de varias copias de los datos del clúster de bases de datos. No obstante, los datos del volumen del clúster se representan como un único volumen lógico para la instancia de base de datos principal y para las réplicas de Aurora del clúster de bases de datos. Si la instancia principal da error, una réplica de Aurora se convierte en la instancia de base de datos principal.

Aurora también admite opciones de replicación que son específicas de Aurora MySQL y Aurora PostgreSQL.

Para obtener más información, consulte Replicación con Amazon Aurora.

Conmutación por error

Aurora almacena copias de los datos en un clúster de bases de datos en varias zonas de disponibilidad en una única región de AWS. Este almacenamiento se produce independientemente de si las instancias de base de datos en el clúster de bases de datos abarca varias zonas de disponibilidad. Al crear réplicas de Aurora entre zonas de disponibilidad, Aurora las aprovisiona automáticamente y las mantiene de forma síncrona. La instancia de base de datos principal se replica de forma síncrona en distintas zonas de disponibilidad en réplicas de Aurora para proporcionar redundancia de datos, eliminar las congelaciones de E/S y minimizar los picos de latencia durante las copias de seguridad del sistema. Ejecutar un clúster de bases de datos con alta disponibilidad puede mejorar la disponibilidad durante el mantenimiento de sistema planificado y ayuda a proteger las bases de datos contra los errores y las interrupciones de las zonas de disponibilidad.

Para obtener más información, consulte Alta disponibilidad para Amazon Aurora.

Seguridad de la infraestructura en Amazon Aurora

Como se trata de un servicio administrado, Amazon Relational Database Service está protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de AWS y sobre cómo AWS protege la infraestructura, consulte Seguridad en la nube de AWS. Para diseñar su entorno de AWS siguiendo las prácticas recomendadas de seguridad de infraestructura, consulte Protección de la infraestructura en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a la API publicadas en AWS para acceder a Amazon RDS a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, Aurora ofrece características para ayudar a admitir la seguridad de la infraestructura.

Grupos de seguridad

Los grupos de seguridad controlan el acceso del tráfico entrante y saliente a un clúster de base de datos. De forma predeterminada, el acceso de red está deshabilitado para un clúster de base de datos. Puede especificar reglas en un grupo de seguridad que permitan el acceso desde un rango de direcciones IP, un puerto o un grupo de seguridad. Una vez configuradas las reglas de entrada, se aplican las mismas reglas a todos los clústeres de base de datos que estén asociadas a ese grupo de seguridad.

Para obtener más información, consulte Control de acceso con grupos de seguridad.

Public accessibility (Accesibilidad pública)

Cuando lance una instancia de base de datos dentro de una nube privada virtual (VPC, por sus siglas en inglés) basada en el servicio de Amazon VPC, podrá activar o desactivar la accesibilidad pública para esa instancia de base de datos. Para designar si la instancia de base de datos que se crea tiene un nombre de DNS que se resuelve en una dirección IP pública, utilice el parámetro Public accessibility (Accesibilidad pública). Con este parámetro podrá especificar si hay acceso público a la instancia de base de datos. Es posible modificar una instancia de base de datos para activar

o desactivar la accesibilidad pública modificando el parámetro Public accessibility (Accesibilidad pública).

Para obtener más información, consulte Cómo ocultar un clúster de base de datos en una VPC desde Internet..

> **Note**
>
> Si la instancia de base de datos se encuentra en una VPC pero no es accesible públicamente, también puede usar una conexión AWS Site-to-Site VPN o una conexión de Direct Connect para acceder a ella desde una red privada. Para obtener más información, consulte Privacidad del tráfico entre redes.

La API de Amazon RDS y los puntos de enlace de la VPC de tipo interfaz (AWS PrivateLink)

Puede establecer una conexión privada entre los puntos de enlace de la VPC y la API de Amazon RDS mediante la creación de un punto de enlace de la VPC de tipo interfaz. Puntos de enlace de tipo interfaz con tecnología de AWS PrivateLink.

AWS PrivateLink permite acceder de forma privada a las operaciones de la API de Amazon RDS sin una puerta de enlace de Internet, un dispositivo NAT, una conexión de VPN o una conexión de Direct Connect. Las instancias de bases de datos de su VPC no necesitan direcciones IP públicas para comunicarse con los puntos de conexión de la API de Amazon RDS para lanzar, modificar o terminar instancias de base de datos y clústeres de base de datos. Las instancias de bases de datos tampoco necesitan direcciones IP públicas para utilizar ninguna de las operaciones de la API de RDS disponibles. El tráfico entre la VPC y Amazon RDS no sale de la red de Amazon.

Cada punto de enlace de la interfaz está representado por una o más interfaces de red elásticas en las subredes. Para obtener más información sobre las interfaces de red elásticas, consulte Interfaces de red elásticas en la Guía del usuario de Amazon EC2.

Para obtener más información sobre puntos de enlace de la VPC, consulte Puntos de enlace de la VPC de tipo interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC. Para obtener información sobre las operaciones de la API de RDS, consulte la referencia de la API de Amazon RDS.

No necesita un punto de conexión de VPC de la interfaz para conectarse con un clúster de base de datos. Para obtener más información, consulte Escenarios de acceso a un clúster de base de datos en una VPC.

Consideraciones para los puntos de enlace de VPC

Antes de configurar un punto de enlace de la VPC de tipo interfaz para los puntos de enlace de la API de Amazon RDS, asegúrese de revisar Propiedades y limitaciones de puntos de enlace de interfaz en la Guía del usuario de Amazon VPC.

Todas las operaciones de la API de RDS relevantes para la administración de los recursos de Amazon Aurora están disponibles desde la VPC mediante el uso de AWS PrivateLink.

Las políticas de puntos de enlace de VPC son compatibles con los puntos de enlace de API de RDS. De forma predeterminada, se permite el acceso completo a las operaciones de API de RDS a través

del punto de enlace. Para obtener más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la guía del usuario de Amazon VPC.

Disponibilidad

La API de Amazon RDS actualmente admite puntos de enlace de la VPC en las siguientes regiones de AWS:

- US East (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- África (Ciudad del Cabo)
- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Mumbai)
- Asia-Pacífico (Nueva Zelanda)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Taipéi)
- Asia-Pacífico (Tailandia)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Oeste de Canadá (Calgary)
- China (Pekín)
- China (Ningxia)
- Europa (Fráncfort)
- Europa (Zúrich)
- Europa (Irlanda)
- Europa (Londres)
- Europa (París)
- Europa (Estocolmo)

- Europa (Milán)
- Israel (Tel Aviv)
- México (central)
- Medio Oriente (Baréin)
- América del Sur (São Paulo)
- AWS GovCloud (Este de EE. UU.)
- AWS GovCloud (EE. UU. Oeste)

Creación de un punto de enlace de la VPC de interfaz para Amazon RDS API

Puede crear un punto de enlace de la VPC para la API de Amazon RDS mediante la consola de Amazon VPC o AWS Command Line Interface (AWS CLI). Para obtener más información, consulte Creación de un punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto de enlace de la VPC para Amazon RDS API mediante el uso del nombre del servicio `com.amazonaws.region.rds`.

Salvo en las regiones de AWS en China, si habilita un DNS privado para el punto de enlace, podrá realizar solicitudes de la API a Amazon RDS con el punto de enlace de la VPC mediante el nombre de DNS predeterminado de la región de AWS, por ejemplo `rds.us-east-1.amazonaws.com`. En las regiones de China (Pekín) y China (Ningxia) de AWS, puede realizar solicitudes de la API con el punto de enlace de la VPC mediante `rds-api.cn-north-1.amazonaws.com.cn` y `rds-api.cn-northwest-1.amazonaws.com.cn`, respectivamente.

Para obtener más información, consulte Acceso a un servicio a través de un punto de enlace de interfaz en la Guía del usuario de Amazon VPC.

Creación de una política de punto de enlace de la VPC para la Amazon RDS API

Puede asociar una política de punto de enlace con el punto de enlace de la VPC que controla el acceso a la Amazon RDS API. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.

- Los recursos en los que se pueden llevar a cabo las acciones.

Para obtener más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la guía del usuario de Amazon VPC.

Ejemplo: política de punto de enlace de la VPC para acciones de la Amazon RDS API

A continuación, se muestra un ejemplo de una política de punto de enlace para la Amazon RDS API. Cuando se asocia a un punto de enlace, esta política concede acceso a las acciones de la Amazon RDS API enumeradas para todos las entidades principales de todos los recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

Ejemplo: política de punto de enlace de la VPC que deniega todo el acceso desde una cuenta de AWS especificada

La siguiente política de punto de conexión de VPC deniega a la cuenta de AWS `123456789012` todo el acceso a los recursos mediante el punto de conexión. La política permite todas las acciones de otras cuentas.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
```

```
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Prácticas recomendadas de seguridad para Amazon Aurora

Utilice cuentas de AWS Identity and Access Management (IAM) para controlar el acceso a las operaciones de la API de Amazon RDS, especialmente operaciones que crean, modifican o eliminan recursos de Amazon Aurora. Dichos recursos incluyen clústeres de base de datos, grupos de seguridad y grupos de parámetros. Utilice también IAM para controlar acciones que realizan acciones administrativas comunes como copias de seguridad y restauración de clústeres de base de datos.

- Cree un usuario individual para cada persona que administre recursos de Amazon Aurora, incluido usted mismo. No utilice las credenciales raíz de AWS para administrar los recursos de Amazon Aurora.
- Asigne a cada usuario el conjunto mínimo de permisos requerido para realizar sus tareas.
- Use los grupos de IAM para administrar con eficacia los permisos para varios usuarios.
- Rote con regularidad sus credenciales de IAM.
- Configure AWS Secrets Manager para rotar automáticamente los secretos para Amazon Aurora. Para obtener más información, consulte Rotación de sus secretos de AWS Secrets Manager en la guía del usuario de AWS Secrets Manager. También puede recuperar mediante programación las credenciales desde AWS Secrets Manager. Para obtener más información, consulte Recuperar el valor secreto en la guía del usuario de AWS Secrets Manager.

Para obtener más información acerca de la seguridad de Amazon Aurora, consulte Seguridad en Amazon Aurora. Para obtener más información acerca de IAM, consulte AWS Identity and Access Management. Para obtener información acerca de las prácticas recomendadas de IAM, consulte Prácticas recomendadas de IAM.

AWS Security Hub CSPM utiliza controles de seguridad para evaluar las configuraciones de los recursos y los estándares de seguridad para ayudarlo a cumplir varios marcos de conformidad. Para obtener más información sobre el uso de Security Hub para evaluar los recursos de Lambda,

consulte Amazon Relational Database Service controls (Controles de Amazon Relational Database Service) en la Guía del usuario de AWS Security Hub CSPM.

Puede supervisar el uso de RDS en relación con las prácticas recomendadas de seguridad con Security Hub. Para obtener más información, consulte ¿Qué es AWS Security Hub CSPM?.

Utilice la Consola de administración de AWS, la AWS CLI o la API de RDS para cambiar la contraseña para el usuario maestro. Si usa otra herramienta, como un cliente de SQL, para cambiar la contraseña del usuario maestro, podría provocar involuntariamente la revocación de privilegios para el usuario.

Amazon GuardDuty es un servicio de monitorización de la seguridad continuo que analiza y procesa una variedad de orígenes de datos, incluida la actividad de inicio de sesión de Amazon RDS. Utiliza fuentes de información de amenazas y machine learning para identificar la actividad inesperada y potencialmente no autorizada, comportamientos sospechosos de inicio de sesión, así como la actividad malintencionada en su entorno de AWS.

Cuando la protección de RDS de Amazon GuardDuty detecta un intento de inicio de sesión sospechoso o anómalo que implica una amenaza para su base de datos, GuardDuty genera un nuevo resultado con detalles de la base de datos que está potencialmente afectada. Para obtener más información, consulte Supervisión de amenazas con Amazon GuardDuty para protección de RDS para Amazon Aurora.

Control de acceso con grupos de seguridad

Los grupos de seguridad de VPC controlan el acceso del tráfico entrante y saliente de un clúster de base de datos. De forma predeterminada, el acceso de red está desactivado para un clúster de base de datos. Puede especificar reglas en un grupo de seguridad que permitan el acceso desde un rango de direcciones IP, un puerto o un grupo de seguridad. Una vez configuradas las reglas de entrada, se aplican las mismas reglas a todos los clústeres de base de datos que estén asociadas a ese grupo de seguridad. Puede especificar hasta 20 reglas en un grupo de seguridad.

Información general de los grupos de seguridad de VPC

Cada regla de grupo de seguridad de VPC permite a un origen específico acceder a un clúster de base de datos de una VPC asociada a ese grupo de seguridad de VPC. El origen puede ser un rango de direcciones (por ejemplo, 203.0.113.0/24), u otro grupo de seguridad de VPC. Cuando se especifica un grupo de seguridad de VPC como origen, se permite el tráfico entrante procedente de todas las instancias, normalmente servidores de aplicaciones, que utilizan el grupo de seguridad de

VPC. Los grupos de seguridad de VPC pueden tener reglas que rijan el tráfico entrante y saliente. Sin embargo, las reglas de tráfico saliente normalmente no se aplican a clústeres de bases de datos. Las reglas de tráfico saliente solo se aplican si el clúster de base de datos actúa como cliente. Debe utilizar la API de Amazon EC2 o la opción Security Group (Grupo de seguridad) de la consola de VPC para crear grupos de seguridad de VPC.

Cuando cree reglas para el grupo de seguridad de la VPC que permitan el acceso a los clústeres de la VPC, debe especificar un puerto para cada rango de direcciones a las que la regla permite el acceso. Por ejemplo, si desea habilitar el acceso Secure Shell (SSH) a las instancias de la VPC, puede crear una regla que permita el acceso al puerto TCP 22 para el rango de direcciones especificado.

Puede configurar varios grupos de seguridad de VPC que permitan el acceso a puertos distintos para las distintas instancias de la VPC. Por ejemplo, puede crear un grupo de seguridad de VPC que permita el acceso al puerto TCP 80 para los servidores web de la VPC. A continuación, puede crear otro grupo de seguridad de la VPC que permita el acceso al puerto TCP 3306 para las instancias de base de datos de Aurora MySQL en la VPC.

> **Note**
>
> En un clúster de bases de datos de Aurora, el grupo de seguridad de VPC asociado al clúster de bases de datos también se asocia a todas las instancias de base de datos en el clúster de bases de datos. Si cambia el grupo de seguridad de VPC para el clúster de bases de datos o para la instancia de base de datos, el cambio se aplica automáticamente a todas las instancias de base de datos en el clúster de bases de datos.

Para obtener más información sobre los grupos de seguridad de VPC, consulte Grupos de seguridad en la Guía del usuario de Amazon Virtual Private Cloud.

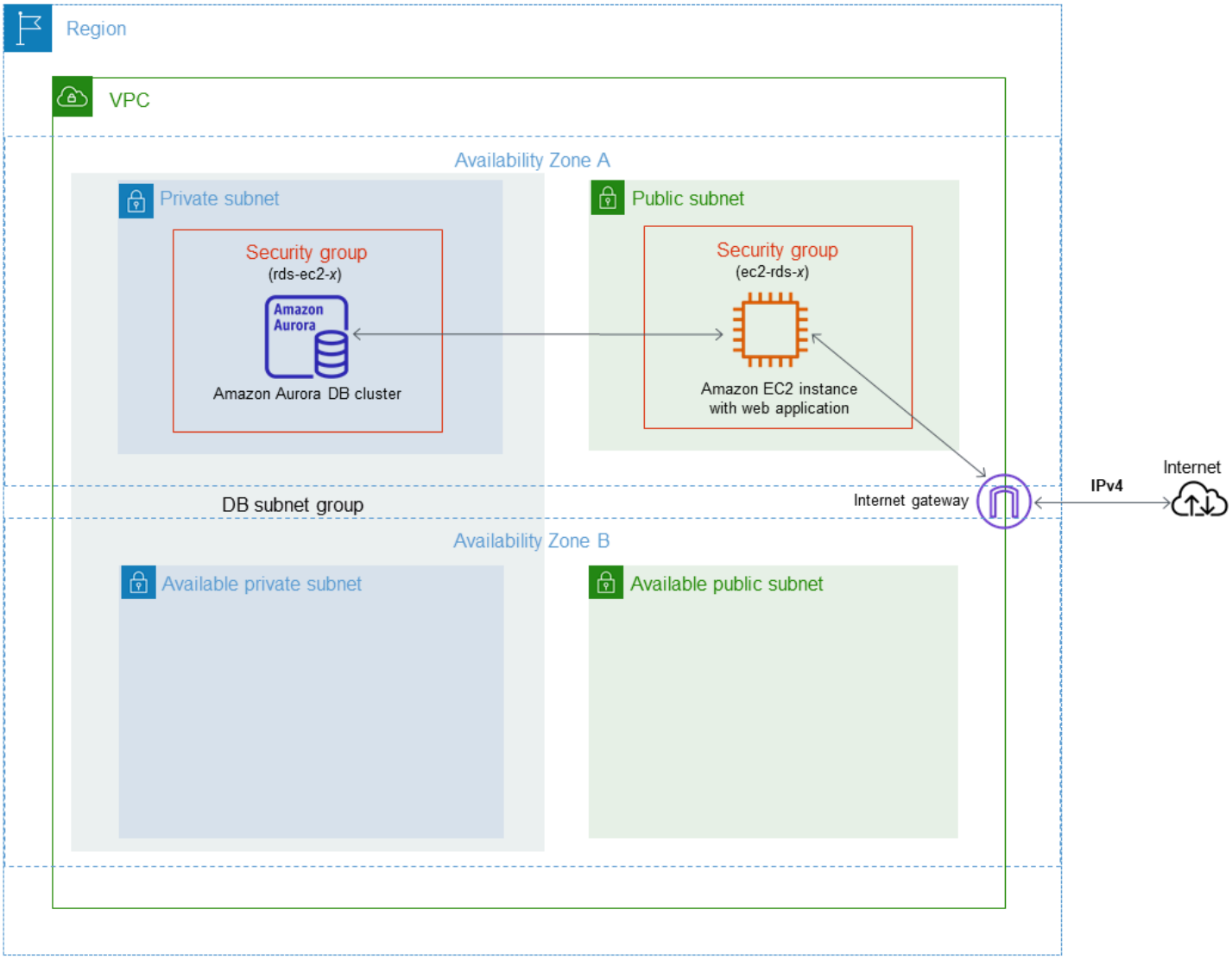
> **Note**
>
> Si el clúster de la base de datos se encuentra en una VPC pero no es accesible públicamente, también puede usar una conexión AWS Site-to-Site VPN o una conexión de Direct Connect para acceder desde una red privada. Para obtener más información, consulte Privacidad del tráfico entre redes.

Escenario de grupos de seguridad

Un uso común de clúster de base de datos en una VPC es compartir datos con un servidor de aplicaciones que se ejecuta en una instancia de Amazon EC2 de la misma VPC, al que obtiene acceso desde una aplicación cliente situada fuera de la VPC. Para este escenario, se utilizan las páginas de RDS y VPC de la Consola de administración de AWS o las operaciones de la API de RDS y EC2 para crear las instancias y los grupos de seguridad necesarios:

1. Cree un grupo de seguridad de VPC (por ejemplo, `sg-0123ec2example`) y defina reglas de entrada que utilicen las direcciones IP de la aplicación cliente como origen. Este grupo de seguridad permite que la aplicación cliente se conecte a las instancias EC2 de una VPC que utilice este grupo de seguridad.
2. Cree una instancia EC2 para la aplicación y añada la instancia EC2 al grupo de seguridad de VPC (`sg-0123ec2example`) que creó en el paso anterior.
3. Cree un segundo grupo de seguridad de VPC (por ejemplo, `sg-6789rdsexample`) y cree una regla nueva especificando el grupo de seguridad de VPC que creó en el paso 1 (`sg-0123ec2example`) como origen.
4. Cree un clúster nuevo de base de datos y agregue el clúster de base de datos al grupo de seguridad de la VPC (`sg-6789rdsexample`) que creó en el paso anterior. Cuando cree el clúster de base de datos, utilice el mismo número de puerto que especificó para la regla de grupo de seguridad de VPC (`sg-6789rdsexample`) que creó en el paso 3.

En el siguiente diagrama se muestra este escenario.



Para obtener instrucciones detalladas acerca de la configuración de una VPC para este escenario, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4). Para obtener más información acerca del uso de una VPC, consulte VPC de Amazon y Amazon Aurora.

Creación de un grupo de seguridad de VPC

Puede crear un grupo de seguridad de VPC para una instancia de base de datos mediante la consola de VPC. Para obtener información sobre la creación de un grupo de seguridad, consulte Proporcionar acceso al clúster de base de datos en la VPC mediante la creación de un grupo de seguridad y Grupos de seguridad en la Guía del usuario de Amazon Virtual Private Cloud.

Asociación de un grupo de seguridad con un clúster de bases de datos

Puede asociar un grupo de seguridad con un clúster de bases de datos mediante la opción Modificar el clúster de la consola de RDS, la API de Amazon RDS `ModifyDBCluster` o el comando `modify-db-cluster` de AWS CLI.

El siguiente ejemplo de la CLI asocia un grupo de VPC específico y elimina los grupos de seguridad de base de datos del clúster de base de datos.

```
aws rds modify-db-cluster --db-cluster-identifier dbName --vpc-security-group-ids sg-ID
```

Para obtener información acerca de la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Privilegios de la cuenta de usuario maestro

Cuando se crea un clúster nuevo de base de datos, el usuario maestro predeterminado que se utiliza obtiene determinados privilegios para ese clúster de bases de datos. No se puede cambiar el nombre de usuario maestro después de crear el clúster de base de datos.

> **Important**
>
> Le recomendamos encarecidamente que no utilice el usuario maestro directamente en sus aplicaciones. En lugar de ello, es mejor ceñirse a la práctica recomendada de utilizar un usuario de base de datos creado con los privilegios mínimos necesarios para su aplicación.

> **Note**
>
> Si elimina los permisos para el usuario maestro de forma accidental, puede restaurarlos modificando el clúster de base de datos y estableciendo una nueva contraseña para el usuario maestro. Para obtener más información acerca de la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora .

En la siguiente tabla se muestran los privilegios y los roles de base de datos que obtiene el usuario maestro para cada uno de los motores de bases de datos.

Motor de base de datos	Privilegio del sistema	Rol de base de datos
Aurora MySQL	Versión 2: `ALTER,ALTER ROUTINE, CREATE,CREATE ROUTINE,CREATE TEMPORARY TABLES,CREATE USER,CREATE VIEW,DELETE,DROP, EVENT,EXECUTE,GRANT OPTION,INDEX,INSERT, LOAD FROM S3,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATI ON CLIENT , REPLICATION SLAVE ,SELECT, SELECT INTO S3,SHOW DATABASES , SHOW VIEW,TRIGGER, UPDATE`	—
	Versión 3: `ALTER,APPLICATION_PASSWORD_ADMIN , ALTER ROUTINE,CONNECTION_ADMIN , CREATE,CREATE ROLE,CREATE ROUTINE,CREATE TEMPORARY TABLES, CREATE USER,CREATE VIEW, DELETE,DROP,DROP ROLE, EVENT,EXECUTE,INDEX, INSERT,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATION CLIENT , REPLICATI ON SLAVE ,ROLE_ADMIN , SET_USER_ ID ,SELECT,SHOW DATABASES ,SHOW VIEW,TRIGGER, UPDATE,XA_RECOVER_ADMIN`  A partir de la versión 3.04.0 de Aurora MySQL, el usuario maestro también obtiene el privilegio `SHOW_ROUTINE` . A partir de la versión 3.09.0 de Aurora MySQL, el usuario maestro también obtiene los privilegios	`rds_superuser_role` Para obtener más información acerca de rds_superuser_role, consulte Modelo de privilegios basado en roles.

Motor de base de datos	Privilegio del sistema	Rol de base de datos
	`FLUSH_OPTIMIZER_COSTS` , `FLUSH_STATUS` , `FLUSH_TABLES` y `FLUSH_USER_RESOURCES` .	
Aurora PostgreSQL	`LOGIN`,`NOSUPERUSER` , `INHERIT`,`CREATEDB`, `CREATEROLE` ,`NOREPLICATION` ,`VALID UNTIL 'infinity'`	`RDS_SUPERUSER` Para obtener más información acerca de RDS_SUPERUSER, consulte Descripción de los roles y permisos de PostgreSQL.

Uso de roles vinculados a servicios de Amazon Aurora

Amazon Aurora utiliza roles vinculados a servicios de AWS Identity and Access Management (IAM) Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a Amazon Aurora. Los roles vinculados a servicios están predefinidos por Amazon Aurora e incluyen todos los permisos que el servicio requiere para llamar a otros servicios de AWS en su nombre.

Un rol vinculado a un servicio simplifica el uso de Amazon Aurora porque ya no tendrá que agregar manualmente los permisos necesarios. Amazon Aurora define los permisos de sus roles vinculados a servicios y, a menos que esté definido de otra manera, solo Amazon Aurora puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se pueda asociar a ninguna otra entidad de IAM.

Las funciones se pueden eliminar únicamente después de eliminar primero sus recursos relacionados. De esta forma, se protegen los recursos de Amazon Aurora, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información sobre otros servicios que admiten roles vinculados al servicio, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Sí en la columna Rol vinculado al servicio. Elija una opción Sí con un enlace para ver la documentación acerca del rol vinculado a servicios en cuestión.

Permisos de roles vinculados a servicios de Amazon Aurora

Amazon Aurora utiliza el rol vinculado al servicio denominado AWSServiceRoleForRDS para permitir que Amazon RDS llame a servicios de AWS en nombre de sus clústeres de base de datos.

El rol vinculado al servicio AWSServiceRoleForRDS confía en que los siguientes servicios asuman el rol:

- `rds.amazonaws.com`

Este rol vinculado al servicio tiene una política de permisos adjunta llamada `AmazonRDSServiceRolePolicy`, que le otorga permisos para operar en su cuenta.

Para obtener más información sobre esta política, incluido el documento de política de JSON, consulte AmazonRDSServiceRolePolicy en la Guía de referencia de políticas administradas de AWS.

Note

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o función) crear, editar o eliminar la descripción de una función vinculada a un servicio. Si aparece el siguiente mensaje de error:
Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.
Asegúrese de que tiene habilitados los permisos siguientes:

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"rds.amazonaws.com"
        }
    }
}
```

Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Creación de un rol vinculado a un servicio de Amazon Aurora

No necesita crear manualmente un rol vinculado a un servicio. Cuando crea un clúster de base de datos, Amazon Aurora vuelve a crear por usted el rol vinculado al servicio.

Important

Si utilizaba el servicio Amazon Aurora antes del 1 de diciembre de 2017, cuando comenzó a admitir roles vinculados a servicios, entonces Amazon Aurora creó el rol AWSServiceRoleForRDS en su cuenta. Para obtener más información, consulte Un nuevo rol ha aparecido en mi cuenta de AWS.

Si elimina este rol vinculado a servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Cuando crea un clúster de base de datos, Amazon Aurora vuelve a crear por usted el rol vinculado al servicio.

Modificación de un rol vinculado a un servicio de Amazon Aurora

Amazon Aurora no permite editar el rol vinculado al servicio AWSServiceRoleForRDS. Después de crear un rol vinculado a un servicio, no puede cambiarle el nombre, ya que varias entidades pueden hacer referencia al mismo. Sin embargo, sí puede editar la descripción del rol con IAM. Para obtener más información, consulte Editar un rol vinculado al servicio en la Guía del usuario de IAM.

Eliminación de un rol vinculado a un servicio de Amazon Aurora

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a un servicio, le recomendamos que elimine dicho rol. De esta forma no tiene una entidad no utilizada que no se monitoree ni mantenga de forma activa. Sin embargo, debe eliminar todos los clústeres para poder eliminar el rol vinculado al servicio.

Limpiar un rol vinculado a un servicio

Antes de poder utilizar IAM para eliminar un rol vinculado a un servicio, primero debe confirmar que dicho rol no tiene sesiones activas y eliminar los recursos que utiliza.

Para comprobar si el rol vinculado a un servicio tiene una sesión activa en la consola de IAM

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola de IAM, elija Roles. Luego, elija el nombre (no la casilla de verificación) del rol AWSServiceRoleForRDS.
3. En la página Resumen del rol seleccionado, elija la pestaña Acceso reciente.
4. En la pestaña Acceso reciente, revise la actividad reciente del rol vinculado a servicios.

> **Note**
>
> Si no está seguro de si Amazon Aurora utiliza el rol AWSServiceRoleForRDS, puede intentar eliminar el rol para comprobarlo. Si el servicio está utilizando el rol, este no podrá eliminarse y podrá ver las regiones de AWS en las que se está utilizando. Si el rol se está utilizando, debe esperar que la sesión finalice para poder eliminarlo. No se puede revocar la sesión de un rol vinculado a un servicio.

Si desea eliminar el rol AWSServiceRoleForRDS, primero debe eliminar sus clústeres de base de datos totales.

Eliminación de todos los clústeres

Use alguno de estos procedimientos para eliminar un clúster. Repita el procedimiento para cada uno de los clústeres.

Para eliminar un clúster (consola)

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la lista Databases (Bases de datos), elija el clúster que desea eliminar.
3. En clúster Actions (Acciones de clúster), seleccione Delete (Eliminar).
4. Elija Eliminar.

Para eliminar un clúster (CLI)

Consulte `delete-db-cluster` en la referencia de comandos de AWS CLI.

Para eliminar un clúster (API)

Consulte `DeleteDBCluster` en la Amazon RDS API Reference.

Utilice la consola de IAM, la CLI de IAM o la API de IAM para eliminar el rol vinculado al servicio AWSServiceRoleForRDS. Para obtener más información, consulte Eliminar un rol vinculado a un servicio en la Guía del usuario de IAM.

Permisos de roles vinculados a servicios para Amazon RDS Beta

Amazon Aurora utiliza el rol vinculado al servicio llamado `AWSServiceRoleForRDSBeta` para que Amazon Aurora pueda llamar a los servicios AWS en nombre de sus recursos de base de datos de RDS.

El rol vinculado al servicio AWSServiceRoleForRDSBeta depende de los siguientes servicios para asumir el rol:

- `rds.amazonaws.com`

Este rol vinculado al servicio tiene una política de permisos adjunta llamada `AmazonRDSBetaServiceRolePolicy` que le otorga permisos para operar en

su cuenta. Para obtener más información, consulte Política administrada de:AWS AmazonRDSBetaServiceRolePolicy.

> **Note**
>
> Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o función) crear, editar o eliminar la descripción de una función vinculada a un servicio. Si aparece el siguiente mensaje de error:
> Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.
> Asegúrese de que tiene habilitados los permisos siguientes:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSBetaServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

Rol vinculado a servicios para Amazon RDS Preview

Amazon Aurora utiliza el rol vinculado al servicio llamado `AWSServiceRoleForRDSPreview` para que Amazon Aurora pueda llamar a los servicios AWS en nombre de sus recursos de base de datos de RDS.

El rol vinculado al servicio AWSServiceRoleForRDSPreview depende de los siguientes servicios para asumir el rol:

- `rds.amazonaws.com`

Este rol vinculado al servicio tiene una política de permisos adjunta llamada `AmazonRDSPreviewServiceRolePolicy` que le otorga permisos para operar en su cuenta. Para obtener más información, consulte Política administrada de:AWS AmazonRDSPreviewServiceRolePolicy.

> **Note**
>
> Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o función) crear, editar o eliminar la descripción de una función vinculada a un servicio. Si aparece el siguiente mensaje de error:
> Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.
> Asegúrese de que tiene habilitados los permisos siguientes:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSPreviewServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM.

VPC de Amazon y Amazon Aurora

Amazon Virtual Private Cloud (Amazon VPC) hace posible lanzar recursos de AWS, como clústeres de base de datos de Aurora, en una nube privada virtual (VPC).

Cuando utilice una VPC, puede controlar todos los aspectos del entorno de red virtual. Puede elegir su propio rango de direcciones IP, crear subredes y configurar listas de enrutamiento y control de acceso. Es posible ejecutar el clúster de base de datos en una VPC sin ningún coste adicional.

Las cuentas tienen una VPC predeterminada. Todos los nuevos clústeres de bases de datos se crean en la VPC predeterminada, a menos que se especifique lo contrario.

Temas

- Uso de una clúster de base de datos en una VPC
- Escenarios de acceso a un clúster de base de datos en una VPC
- Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4)
- Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble)

A continuación, encontrará una discusión acerca de la funcionalidad de la VPC relevante a clústeres de base de datos de Amazon Aurora. Para obtener más información sobre Amazon VPC, consulte la guía de introducción de Amazon VPC y la guía del usuario de Amazon VPC.

Uso de una clúster de base de datos en una VPC

Su clúster de base de datos debe estar dentro de la nube privada virtual (VPC). Una VPC es una red virtual aislada lógicamente de otras redes virtuales en la nube de AWS. Amazon VPC le permite lanzar recursos de AWS, como o una instancia de Amazon EC2, en una VPC. La VPC puede ser una VPC predeterminada que viene con la cuenta o una que se haya creado en ella. Todas las VPC están asociadas a la cuenta de AWS.

La VPC predeterminada tiene tres subredes que se pueden utilizar para aislar recursos dentro de la VPC. La VPC predeterminada también tiene una gateway de Internet que se puede utilizar para proporcionar acceso a los recursos situados dentro de la VPC desde fuera de la VPC.

Para obtener una lista de los escenarios relacionados con los clústeres de bases de datos de Amazon Aurora fuera de una VPC, consulte Escenarios de acceso a un clúster de base de datos en una VPC.

Temas

- Uso de una clúster de base de datos en una VPC
- Control de cifrado de VPC
- Uso de los grupos de subredes de base de datos
- Subredes compartidas
- Direccionamiento IP de Amazon Aurora
- Cómo ocultar un clúster de base de datos en una VPC desde Internet.
- Creación de un clúster de base de datos en una VPC

En los siguientes tutoriales se explica cómo crear una VPC que se puede utilizar en un escenario de Amazon Aurora habitual:

- Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4)
- Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble)

Uso de una clúster de base de datos en una VPC

A continuación se ofrecen algunos consejos para utilizar un clúster de base de datos en una VPC:

- La VPC debe tener dos subredes como mínimo. Estas subredes deben estar en dos zonas de disponibilidad distintas de la Región de AWS en la que desea implementar el clúster de base de datos. Una subred es un segmento del rango de direcciones IP de una VPC que puede especificar y que le permite agrupar clústeres de base de datos según sus necesidades operativas y de seguridad.
- Si desea que un clúster de base de datos de la VPC sea accesible públicamente, debe activar los atributos DNS hostnames y DNS resolution.
- La VPC debe tener un grupo de subredes de base de datos que haya creado. Para crear un grupo de subredes de base de datos, especifique las subredes que ha creado. Amazon Aurora elige una subred y una dirección IP dentro de esa subred para asociarla con la instancia de base de datos principal de su clúster de base de datos. La instancia de base de datos principal utiliza la zona de disponibilidad que contiene la subred.
- La VPC debe tener un grupo de seguridad de VPC que permita el acceso a el clúster de base de datos.

Para obtener más información, consulte Escenarios de acceso a un clúster de base de datos en una VPC.

- Los bloques de CIDR de cada una de las subredes deben ser lo suficientemente grandes como para acomodar direcciones IP de repuesto para que Amazon Aurora las use durante las actividades de mantenimiento, incluyendo la conmutación por error y el escalado de recursos de computación. Por ejemplo, un rango como 10.0.0/24 y 10.0.2.0/24 suele ser lo suficientemente grande.
- El atributo instance tenancy de una VPC puede definirse como default o dedicated. Todas las VPC predeterminadas tienen el atributo de tenencia de instancia definido como default, y una VPC predeterminada puede admitir cualquier clase de instancia de base de datos.

  Si opta por tener el clúster de base de datos en una VPC dedicada cuyo atributo de tenencia de instancia está establecido en dedicado, la clase de instancia de base de datos de el clúster debe ser uno de los tipos aprobados de instancia dedicada de Amazon EC2. Por ejemplo, la instancia dedicada r5.large de EC2 corresponde a la clase de instancia de base de datos r5.large. Para obtener información acerca de la tenencia de instancias en una VPC, consulte Instancias dedicadas en la Guía del usuario de Amazon Elastic Compute Cloud.

  Para obtener más información acerca de los tipos de instancias que puede haber en una instancia dedicada, consulte Instancias dedicadas de Amazon EC2 en la página de precios de Amazon EC2.

> **Note**
>
> Cuando establece el atributo de tenencia de instancia en dedicado para un clúster de base de datos, no garantiza que el clúster de base de datos se ejecute en un host dedicado.

Control de cifrado de VPC

Los controles de cifrado de VPC le permiten aplicar el cifrado en tránsito para todo el tráfico de red dentro de las VPC. Utilice el control de cifrado para cumplir con los requisitos de conformidad normativa, asegurándose de que solo se pueda aprovisionar hardware basado en Nitro con capacidad de cifrado en las VPC designadas. El control de cifrado también detecta los problemas de compatibilidad cuando se solicita la API y no durante el aprovisionamiento. Las cargas de trabajo actuales siguen funcionando y solo se bloquean las nuevas solicitudes incompatibles.

Configure los controles de cifrado de la VPC configurando el modo de control de la VPC en:

- desactivado (predeterminado)
- supervisar
- forzado

Para comprobar el modo de control actual de la VPC, utilice la Consola de administración de AWS o el comando de la API o la CLI DescribeVpcs.

Si la VPC aplica el cifrado, solo puede aprovisionar clústeres de base de datos basados en Nitro que admitan el cifrado en tránsito en esa VPC. Para obtener más información, consulte Tipos de clase de instancia de base de datos. Para obtener información sobre las instancias de Nitro, consulte Instancias integradas en AWS Nitro System en la Guía del usuario de Amazon EC2.

> **Note**
>
> Si intenta aprovisionar clústeres de base de datos incompatibles en una VPC con cifrado, Aurora devuelve una excepción `VpcEncryptionControlViolationException`.

Para Aurora Serverless para MySQL y PostreSQL, el control de cifrado requiere la versión de la plataforma 3 o superior.

Uso de los grupos de subredes de base de datos

Las subredes son segmentos del rango de direcciones IP de una VPC que se definen para agrupar recursos de acuerdo con las necesidades operativas y de seguridad. Un grupo de subredes de base de datos es una colección de subredes (normalmente privadas) que se crean en una VPC y que después se asignan a los clústeres de bases de datos. Un grupo de subredes de base de datos le permite especificar una VPC específica al crear clústeres de bases de datos utilizando la AWS CLI o la API de RDS. Si utiliza la consola, solo puede elegir la VPC y los grupos de subredes que desea utilizar.

Cada grupo de subredes de base de datos debe tener subredes como mínimo en dos zonas de disponibilidad de una determinada Región de AWS. Cuando crea un clúster de base de datos en una VPC, debe elegir un grupo de subredes de base de datos. Desde el grupo de subred de base de datos, Amazon Aurora elige una subred y una dirección IP dentro de esa subred para asociarla con la instancia de base de datos principal de su clúster de base de datos. La base de datos utiliza la zona de disponibilidad que contiene la subred. Aurora siempre asigna una dirección IP desde una subred que tiene espacio de dirección IP libre.

Las subredes de un grupo de subredes de base de datos son públicas o privadas. Las subredes son públicas o privadas, en función de la configuración que establezca para sus listas de control de acceso a la red (ACL de red) y tablas de enrutamiento. Para que un clúster de base de datos sea accesible públicamente, todas las subredes del grupo de subredes de base de datos deben ser públicas. Si una subred asociada a un clúster de base de datos de acceso público cambia de pública a privada, eso puede afectar a la disponibilidad de el clúster de base de datos.

Para crear un grupo de subredes de base de datos que admita el modo de pila doble, asegúrese de que cada subred que agregue al grupo de subredes de base de datos tenga un bloque de CIDR de protocolo de Internet versión 6 (IPv6) asociado. Para obtener más información, consulte Direccionamiento IP de Amazon Aurora y el tema sobre cómo migrar a IPv6 en la Guía del usuario de Amazon VPC.

Cuando Amazon Aurora crea un clúster de base de datos en una VPC, asigna una interfaz de red a el clúster de base de datos utilizando una dirección IP del grupo de subredes de base de datos. Sin embargo, le recomendamos que utilice el nombre del sistema de nombres de dominio (DNS) para conectarse a el clúster de base de datos. Se recomienda hacerlo porque la dirección IP subyacente cambia durante la conmutación por error.

> **Note**
>
> Para cada clúster de base de datos que ejecute en una VPC, asegúrese de reservar al menos una dirección en cada subred del grupo de subredes de base de datos para que la utilice Amazon Aurora para las acciones de recuperación.

Subredes compartidas

Puede crear un clúster de base de datos en una VPC compartida.

Algunas consideraciones a tener en cuenta al utilizar las VPC compartidas:

- Puede mover un clúster de base de datos de una subred de VPC compartida a una subred de VPC no compartida y viceversa.
- Los participantes de una VPC compartida deben crear un grupo de seguridad en la VPC que les permita crear un clúster de base de datos.
- Los propietarios y los participantes de una VPC compartida pueden acceder a la base de datos mediante consultas SQL. Sin embargo, solo el creador de un recurso puede realizar llamadas a la API en el recurso.

Direccionamiento IP de Amazon Aurora

Las direcciones IP permiten que los recursos de la VPC se comuniquen entre sí y con otros recursos a través de Internet. Amazon Aurora admite los protocolos de direcciones IPv4 e IPv6. De forma predeterminada, Amazon Aurora y Amazon VPC utilizan el protocolo de direccionamiento IPv4. No puedes desactivar este comportamiento. Al crear una VPC, debe especificar un bloque de CIDR IPv4 (un intervalo de direcciones IPv4 privadas). De manera opcional, puede asignar un bloque de CIDR IPv6 a su VPC y sus subredes y asignar direcciones IPv6 de dicho bloque a clústers de base de datos de su subred

La compatibilidad con el protocolo IPv6 amplía el número de direcciones IP admitidas. Al utilizar el protocolo IPv6, se asegura de tener suficientes direcciones disponibles para el futuro crecimiento de Internet. Los recursos de RDS nuevos y existentes pueden utilizar direcciones IPv4 e IPv6 dentro de su VPC. Configurar, proteger y traducir el tráfico de red entre los dos protocolos utilizados en diferentes partes de una aplicación puede provocar sobrecarga operativa. Puede estandarizar el protocolo IPv6 para los recursos de Amazon RDS para simplificar la configuración de la red.

Temas

- Direcciones IPv4
- Direcciones IPv6
- Modo de pila doble

Direcciones IPv4

Al crear una VPC, debe especificar un rango de direcciones IPv4 para la VPC como bloque de CIDR como `10.0.0.0/16`. Un grupo de subredes de base de datos define el rango de direcciones IP de este bloque de CIDR que puede utilizar un clúster de base de datos. Esta dirección IP puede ser privada o pública.

Una dirección IPv4 privada es una dirección IP a la que no se puede obtener acceso desde Internet. Se pueden usar direcciones IPv4 privadas para la comunicación entre el clúster de la base de datos y otros recursos, como instancias de Amazon EC2, en la misma VPC. Cada clúster de base de datos tiene una dirección IP privada para la comunicación en la VPC.

Una dirección IP pública es una dirección IPv4 a la que se puede acceder desde Internet. Se pueden usar direcciones públicas para la comunicación entre el clúster de la base de datos y los recursos en

Internet, como un cliente SQL. Debe controlar si un clúster de base de datos recibe una dirección IP pública.

Para ver un tutorial que muestra cómo crear una VPC con solo direcciones IPv4 privadas que puede utilizar con un escenario habitual de Amazon Aurora, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

Direcciones IPv6

De manera opcional, puede asociar un bloque de CIDR IPv6 a su VPC y sus subredes y asignar direcciones IPv6 desde dicho bloque a los recursos de su VPC. Cada dirección IPv6 es única a nivel mundial.

El bloque de CIDR IPv6 de su VPC se asigna automáticamente de entre el grupo de direcciones IPv6 de Amazon. Usted no puede elegir el rango.

Al conectarse a una dirección IPv6, asegúrese de que se cumplan las siguientes condiciones:

- El cliente se ha configurado de manera que se permita el tráfico de la base de datos a través de IPv6.
- Los grupos de seguridad de RDS utilizados por la instancia de base de datos están configurados correctamente para permitir el tráfico de cliente a la base de datos a través de IPv6.
- La pila del sistema operativo de cliente permite el tráfico en la dirección IPv6. Además, los controladores y bibliotecas del sistema operativo están configurados para elegir el punto de conexión de la instancia de base de datos predeterminado correcto (IPv4 o IPv6).

Para obtener más información sobre IPv6, consulte el tema sobre direccionamiento IP en la Guía del usuario de Amazon VPC.

Modo de pila doble

Cuando un clúster de base de datos puede comunicarse a través de los protocolos de direcciones tanto IPv4 como IPv6, se ejecuta en modo de pila doble. Por lo tanto, los recursos pueden comunicarse con el clúster de base de datos a través de IPv4, IPv6 o ambos. RDS deshabilita el acceso a la puerta de enlace de Internet para los puntos de conexión IPv6 de instancias de base de datos en modo de pila doble. RDS hace esto para garantizar que los puntos de conexión IPv6 sean privados y solo se pueda acceder a ellos desde la VPC.

Temas

- Modo de pila doble y grupos de subredes de base de datos
- Utilización de instancias de base de datos en modo de pila doble
- Modificación de clústeres de base de datos de solo IPv4 para utilizar el modo de pila doble
- Disponibilidad de clústeres de base de datos de red de pila doble
- Limitaciones de clústeres de base de datos de red de pila doble

Para ver un tutorial donde se muestra cómo crear una VPC con las direcciones IPv4 y IPv6 que puede utilizar en un escenario habitual de Amazon Aurora, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble).

Modo de pila doble y grupos de subredes de base de datos

Para utilizar el modo de pila doble, asegúrese de que cada subred del grupo de subredes de base de datos que asocie a el clúster de base de datos tenga un bloque de CIDR de IPv6 asociado. Puede crear un nuevo grupo de subredes de base de datos o modificar un grupo existente de subredes de base de datos para cumplir este requisito. Cuando un clúster de base de datos esté en modo de pila doble, los clientes podrán conectarse como siempre. Asegúrese de que los firewalls de seguridad del cliente y los grupos de seguridad de instancias de base de datos RDS estén configurados correctamente para permitir el tráfico a través de IPv6. Para conectarse, los clientes utilizan el punto de conexión de la instancia principal del clúster de base de datos. Las aplicaciones de cliente pueden especificar qué protocolo prefieren al conectarse a una base de datos. En modo de pila doble, el clúster de base de datos detecta el protocolo de red preferido del cliente (IPv4 o IPv6) y utiliza ese protocolo para la conexión.

Si un grupo de subredes de base de datos deja de admitir el modo de pila doble debido a la eliminación de subredes o a la disociación de CIDR, existe el riesgo de que se produzca un estado de red incompatible para las instancias de base de datos asociadas al grupo de subredes de base de datos. Además, no puede utilizar el grupo de subredes de base de datos al crear un clúster nuevo de base de datos en modo de pila doble.

Para determinar si un grupo de subredes de base de datos admite el modo de pila doble mediante la Consola de administración de AWS, consulte Network type (Tipo de red) en la página de detalles del grupo de subredes de base de datos. Para determinar si un grupo de subredes de base de datos admite el modo de pila doble mediante la AWS CLI, ejecute el comando describe-db-subnet-groups y la vista `SupportedNetworkTypes` en la salida.

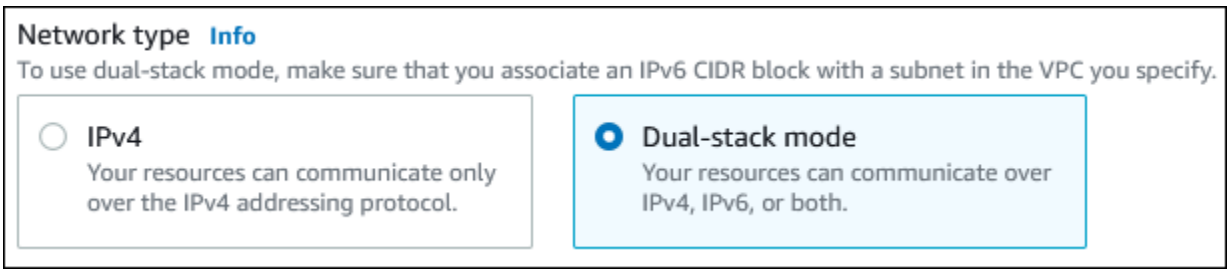
Las réplicas de lectura se tratan como instancias de base de datos independientes y pueden tener un tipo de red diferente al de la instancia de base de datos principal. Si cambia el tipo de red de

la instancia de base de datos principal de una réplica de lectura, la réplica de lectura no se verá afectada. Al restaurar una instancia de base de datos, puede restaurarla a cualquier tipo de red compatible.

Utilización de instancias de base de datos en modo de pila doble

Al crear o modificar un clúster de base de datos, puede especificar que el modo de pila doble permita que los recursos se comuniquen con su clúster de base de datos a través de IPv4, IPv6 o ambos.

Al utilizar la Consola de administración de AWS para crear o modificar una instancia de base de datos, puede especificar el modo de pila doble en la sección Network type (Tipo de red). En la imagen siguiente se muestra la sección Network type (Tipo de red) en la consola.



Si utiliza la AWS CLI para crear o modificar un clúster de base de datos, establezca la opción `--network-type` en `DUAL` para utilizar el modo de pila doble. Si utiliza la API de RDS para crear o modificar un clúster de base de datos, establezca el parámetro `NetworkType` en `DUAL` para utilizar el modo de pila doble. Al modificar el tipo de red de una instancia de base de datos, puede haber un tiempo de inactividad. Si el modo de pila doble no es compatible con la versión del motor de base de datos o el grupo de subredes de base de datos especificados, se devuelve el error `NetworkTypeNotSupported`.

Para obtener más información acerca de la creación de un clúster de base de datos, consulte Creación de un clúster de base de datos de Amazon Aurora. Para obtener más información sobre la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

Para determinar si un clúster de base de datos está en modo de pila doble mediante la consola, consulte Network type (Tipo de red) en la pestaña Connectivity & security (Conectividad y seguridad) de el clúster de base de datos.

Modificación de clústeres de base de datos de solo IPv4 para utilizar el modo de pila doble

Puede modificar el clúster de base de datos solo IPv4 para utilizar el modo de pila doble. Para ello, cambie el tipo de red de el clúster de base de datos. La modificación podría dar lugar a un tiempo de inactividad.

Se recomienda cambiar el tipo de red de los clústeres de base de datos de Amazon Aurora durante un período de mantenimiento. Actualmente, no se admite la configuración del tipo de red de las nuevas instancias en el modo de doble pila. Puede configurar el tipo de red manualmente mediante el comando `modify-db-cluster`.

Antes de modificar un clúster de base de datos para utilizar el modo de pila doble, asegúrese de que su grupo de subredes de base de datos admite el modo de pila doble. Si el grupo de subredes de base de datos asociado a el clúster de base de datos no admite el modo de pila doble, especifique otro grupo de subredes de base de datos que lo admita cuando modifique el clúster de base de datos. La modificación del grupo de subredes de base de datos de un clúster de base de datos puede provocar un tiempo de inactividad.

Si modifica el grupo de subredes de base de datos de un clúster de base de datos antes de cambiar el clúster de base de datos para utilizar el modo de doble pila, asegúrese de que el grupo de subredes de base de datos sea válido para el clúster de base de datos antes y después del cambio.

Le recomendamos que ejecute la API modify-db-clúster únicamente con el parámetro `--network-type` con el valor `DUAL` para cambiar la red de un clúster de Amazon Aurora al modo de doble pila. Si se añaden otros parámetros junto con el parámetro `--network-type` en la misma llamada a la API, se podría producir un tiempo de inactividad.

Si no puede conectarse a el clúster de base de datos después del cambio, asegúrese de que los firewalls de seguridad de la base de datos y del cliente y las tablas de enrutamiento se hayan configurado correctamente para permitir el tráfico a la base de datos de la red seleccionada (IPv4 o IPv6). Es posible que también tenga que modificar los parámetros, las bibliotecas o los controladores del sistema operativo para conectarse mediante una dirección IPv6.

Para modificar un clúster de base de datos solo IPv4 para utilizar el modo de pila doble

1. Modifique un grupo de subredes de base de datos para admitir el modo de pila doble o cree un grupo de subredes de base de datos que admita el modo de pila doble:

    a. Asocie un bloque de CIDR IPv6 a su VPC.

Para obtener más información, consulte el tema Agregue un bloque CIDR de IPv6 a su VPC en la Guía del usuario de Amazon VPC.

b. Adjunte el bloque de CIDR IPv6 a todas las subredes de su grupo de subredes de base de datos.

Para obtener más información, consulte el tema Agregue un bloque CIDR de IPv6 a su subred en la Guía del usuario de Amazon VPC.

c. Confirme que el grupo de subredes de base de datos admita el modo de pila doble.

Si utiliza la Consola de administración de AWS, seleccione el grupo de subredes de base de datos y asegúrese de que el valor Supported network types (Tipos de redes compatibles) sea Dual, IPv4 (Doble, IPv4).

Si utiliza la AWS CLI, ejecute el comando describe-db-subnet-groups y asegúrese de que el valor `SupportedNetworkType` de la instancia de base de datos sea `Dual, IPv4`.

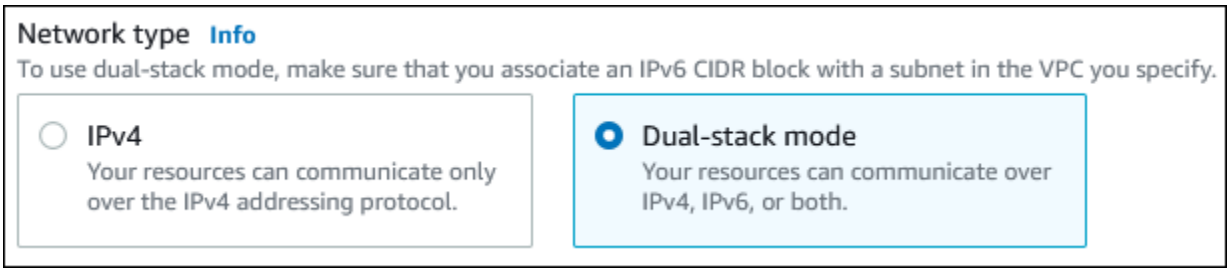
2. Modifique el grupo de seguridad asociado a el clúster de base de datos para permitir conexiones IPv6 a la base de datos o cree un nuevo grupo de seguridad que permita conexiones IPv6.

Para obtener instrucciones, consulte el tema sobre cómo crear un grupo de seguridad en la Guía del usuario de Amazon VPC.

3. Modifique la instancia base de datos para admitir el modo de pila doble. Para ello, defina Network type (Tipo de red) en Dual-stack mode (Modo de pila doble).

Si utiliza la consola, asegúrese de que la siguiente configuración sea correcta:

- Network type (Tipo de red): Dual-stack mode (Modo de pila doble)



- DB subnet group (Grupo de subredes de base de datos): el grupo de subredes de base de datos que configuró en un paso anterior
- Security group (Grupo de seguridad): la seguridad que configuró en el paso anterior

Si utiliza la AWS CLI, asegúrese de que la siguiente configuración sea correcta:

- `--network-type` – `dual`
- `--db-subnet-group-name`: el grupo de subredes de base de datos que configuró en un paso anterior
- `--vpc-security-group-ids`: el grupo de seguridad de la VPC que configuró en un paso anterior

Por ejemplo:

```
aws rds modify-db-cluster --db-cluster-identifier my-cluster --network-type "DUAL"
```

4. Confirme que el clúster de base de datos admite el modo de pila doble.

   Si utiliza la consola, elija la pestaña Configuration (Configuración) para la instancia de base de datos. En esa pestaña, asegúrese de que el valor de Network type (Tipo de red) es Dual-stack mode (Modo de pila doble).

   Si utiliza la AWS CLI, ejecute al comando describe-db-clústers y asegúrese de que el valor `NetworkType` del clúster de base de datos sea `dual`.

   Ejecute el comando `dig` en el punto de conexión de la instancia de base de datos del escritor para identificar la dirección IPv6 que tiene asociada.

   ```
   dig db-instance-endpoint AAAA
   ```

   Utilice el punto de conexión de la instancia de base de datos del escritor (no la dirección IPv6) para conectarse a el clúster de base de datos.

Disponibilidad de clústeres de base de datos de red de pila doble

Los clústeres de base de datos de red de doble pila están disponibles en todas las Regiones de AWS, a excepción de las siguientes:

- Asia-Pacífico (Hyderabad)
- Asia-Pacífico (Malasia)

- Asia-Pacífico (Melbourne)
- Asia-Pacífico (Tailandia)
- Oeste de Canadá (Calgary)
- Europa (España)
- Europa (Zúrich)
- Israel (Tel Aviv)
- México (central)
- Medio Oriente (EAU)

Las siguientes versiones del motor de base de datos admiten clústeres de base de datos de red de doble pila:

- Versiones de Aurora MySQL:
  - Versiones 3.02 y posteriores a la 3
  - Versión 2.09.1 y posteriores a la 2

  Para obtener más información sobre Aurora MySQL, consulte las notas de la versión de Aurora MySQL.
- Versiones de Aurora PostgreSQL:
  - Versiones 15.2 y posteriores
  - Versión 14.3 y versiones posteriores a la 14
  - Versión 13.7 y versiones posteriores a la 13

  Para obtener más información sobre Aurora PostgreSQL, consulte las notas de la versión de Aurora PostgreSQL.

Limitaciones de clústeres de base de datos de red de pila doble

Las siguientes limitaciones se aplican a los clústeres de base de datos de red de pila doble:

- Los clústeres de base de datos no pueden utilizar el protocolo IPv6 exclusivamente. Pueden utilizar IPv4 exclusivamente o utilizar el protocolo IPv4 y IPv6 (modo de pila doble).
- Amazon RDS no admite subredes IPv6 nativas.
- Los clústeres de base de datos que utilizan el modo de pila doble deben ser de tipo privado. No pueden ser accesibles públicamente.

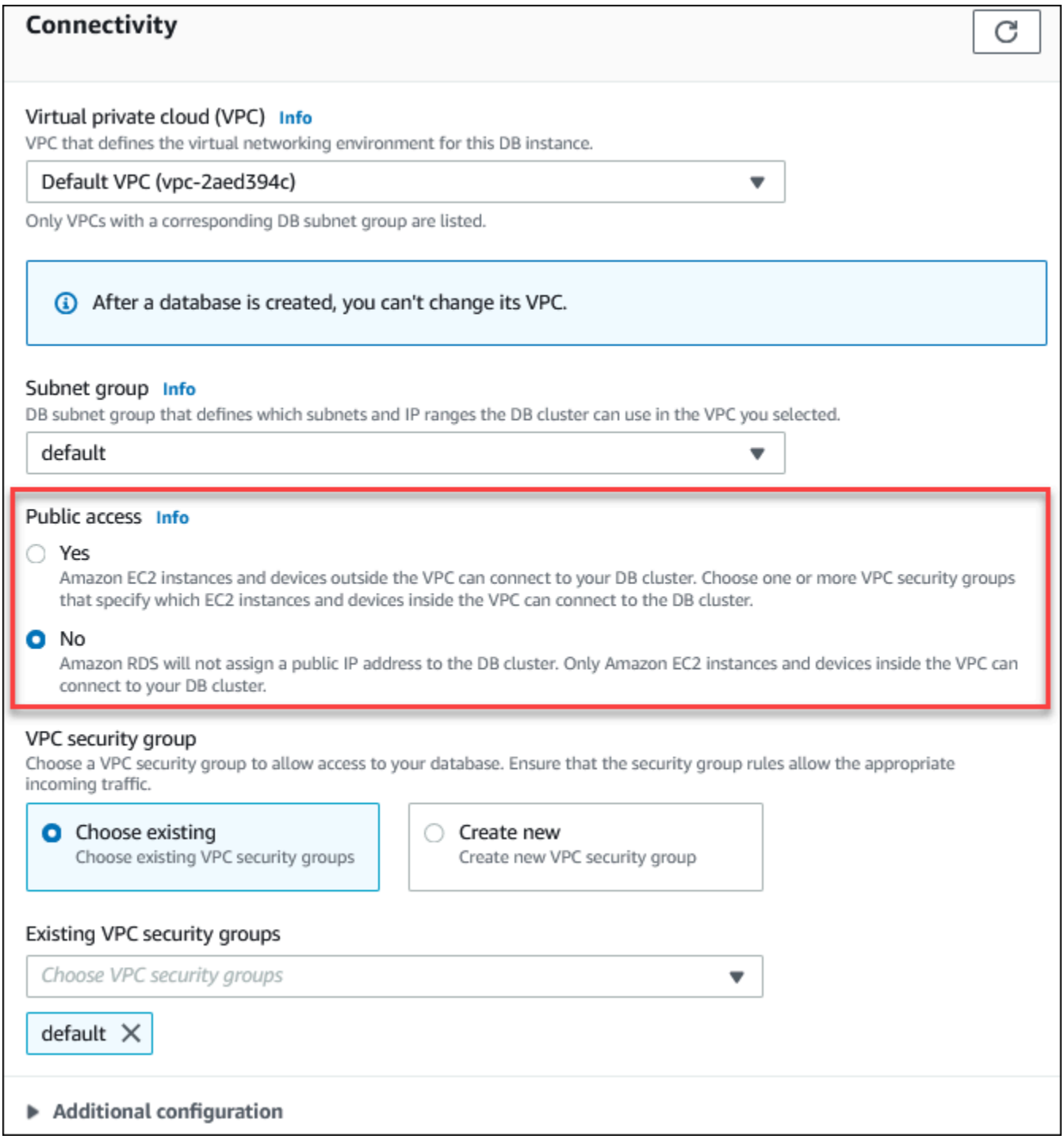
- No puede utilizar RDS Proxy con clústeres de base de datos en modo de pila doble.

Cómo ocultar un clúster de base de datos en una VPC desde Internet.

Un escenario común de Amazon Aurora consiste en tener una VPC en la que hay una instancia de Amazon EC2 con una aplicación web abierta al público y un clúster de base de datos con una base de datos que no es de acceso público. Por ejemplo, puede crear una VPC que tenga una subred pública y una subred privada. Las instancias de EC2 que funcionan como servidores web se pueden implementar en la subred pública. Los Las instancias de base de datos se implementan en la subred privada. En una implementación de este tipo, solo los servidores web tienen acceso a los clústeres de bases de datos. Para ver una ilustración de este escenario, consulte Acceso a un clúster de base de datos en una VPC desde una instancia de Amazon EC2 de la misma VPC.

Cuando lanza un clúster de base de datos dentro de una VPC, el clúster de base de datos tiene una dirección IP privada para el tráfico dentro de la VPC. Esta dirección IP privada no es accesible públicamente. Puede utilizar la opción Public access (Acceso público) para designar si el clúster de base de datos también tiene una dirección IP pública además de la dirección IP privada. Si el clúster se designa como de acceso público, su punto de conexión DNS se resuelve en la dirección IP privada desde dentro de la VPC. Se resuelve en la dirección IP pública desde fuera de la VPC. Acceso a la instancia de base de datos está controlado en última instancia por el grupo de seguridad que utiliza. No se permite el acceso público si el grupo de seguridad asignado al clúster de la base de datos no incluye reglas de entrada que lo permitan. Además, para que un clúster de base de datos sea accesible públicamente, las subredes del grupo de subredes de base de datos deben tener una puerta de enlace de Internet. Para obtener más información, consulte No puede conectarse a la instancia de base de datos de Amazon RDS.

Es posible modificar un clúster de base de datos para activar o desactivar la accesibilidad pública modificando la opción Public access (Acceso público). En la ilustración siguiente se muestra la opción Public access (Acceso público) en la sección Additional connectivity configuration (Configuración de conectividad adicional). Para definir la opción, abra la sección Additional connectivity configuration (Configuración de conectividad adicional) en la sección Connectivity (Conectividad).



Para obtener información sobre cómo modificar una instancia de base de datos para establecer la opción Public access (Acceso público), consulte Modificación de una instancia de base de datos en un clúster de base de datos.

Creación de un clúster de base de datos en una VPC

Los siguientes procedimientos le ayudan a crear un clúster de base de datos en una VPC. Para utilizar la VPC predeterminada, puede comenzar con el paso 2, y utilizar la VPC y el grupo de subredes de la base de datos que ya se han creado para usted. Si desea crear una VPC adicional, puede crear una VPC nueva.

> **Note**
>
> Si desea que un clúster de base de datos de la VPC sea accesible públicamente, debe actualizar la información de DNS para la VPC activando los atributos DNS hostnames y DNS resolution de la VPC. Para obtener información acerca de cómo actualizar la información de DNS para una instancia de VPC, consulte Actualización de la compatibilidad de DNS para su VPC.

Siga estos pasos para crear una instancia de base de datos en una VPC:

- Paso 1: Crear una VPC
- Paso 2: Crear un grupo de subredes de base de datos
- Paso 3: Crear un grupo de seguridad de VPC
- Paso 4: Crear la instancia de base de datos en la VPC

Paso 1: Crear una VPC

Cree una VPC con subredes en al menos dos zonas de disponibilidad. Utilizará estas subredes cuando cree un grupo de subredes de base de datos. Si tiene una VPC predeterminada, se crea automáticamente una subred en cada zona de disponibilidad de la Región de AWS.

Para obtener más información, consulte Creación de una VPC con subredes públicas y privadas o Creación de una VPC en la Guía del usuario de Amazon VPC.

Paso 2: Crear un grupo de subredes de base de datos

Un grupo de subredes de base de datos es una colección de subredes (normalmente privadas) que se crean para una VPC y que después se asignan a los clústeres de bases de datos. Un grupo de subredes de base de datos le permite especificar una VPC específica al crear clústeres de bases

de datos utilizando la AWS CLI o API. Si utiliza la consola, solo puede elegir la VPC y las subredes que desea utilizar. Cada grupo de subredes de base de datos debe tener como mínimo una subred en al menos dos zonas de disponibilidad de la Región de AWS. Como práctica recomendada, cada grupo de subredes de base de datos debería tener al menos una subred por cada una de las zonas de disponibilidad en la Región de AWS.

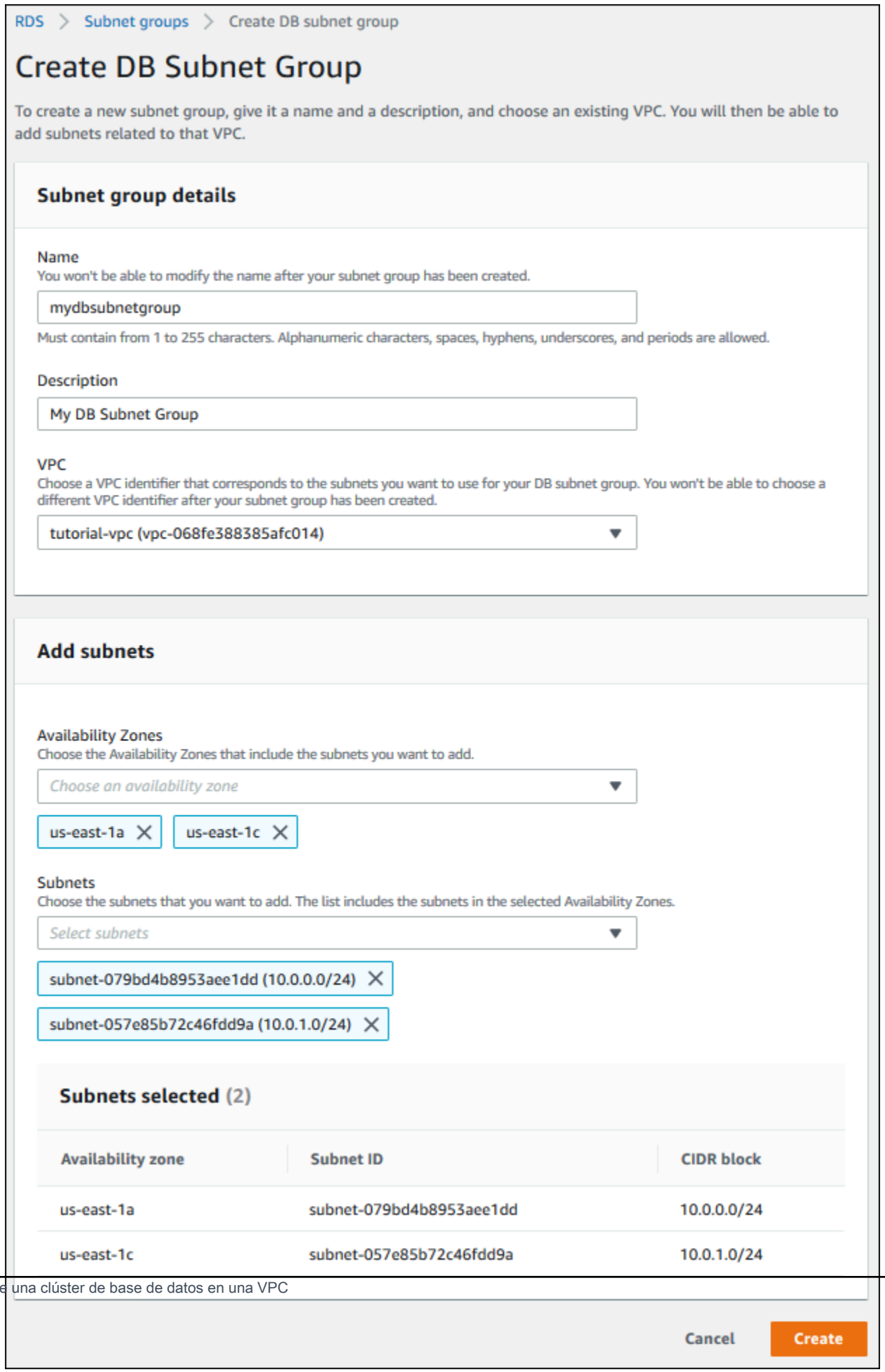
Para que un clúster de base de datos sea accesible públicamente, las subredes del grupo de subredes de base de datos deben tener una puerta de enlace de Internet. Para obtener más información sobre las puertas de enlace de Internet, consulte Conectar subredes a Internet por medio de una puerta de enlace de Internet en la Guía del usuario de Amazon VPC.

Cuando crea una clúster de base de datos en una VPC, debe elegir un grupo de subredes de base de datos. Amazon Aurora elige una subred y una dirección IP dentro de esa subred para asociarla con el clúster de base de datos. Si no existen grupos de subredes de base de datos, Amazon Aurora crea un grupo de subredes predeterminado cuando se crea un clúster de base de datos. Amazon Aurora crea y asocia una interfaz de red elástica a su clúster de base de datos con esa dirección IP. El clúster de base de datos utiliza la zona de disponibilidad que contiene la subred.

En este paso, debe crear un grupo de subredes de base de datos y debe agregar las subredes que creó para la VPC.

Para crear un grupo de subredes de base de datos

1. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Subnet groups.
3. Elija Create DB Subnet Group.
4. En Name, escriba el nombre del grupo de subredes de base de datos.
5. En Description, escriba la descripción del grupo de opciones de base de datos.
6. Para la VPC, elija la VPC predeterminada o la VPC que ha creado.
7. En la sección Agregar subredes, elija las zonas de disponibilidad que incluyen las subredes en Zonas de disponibilidad, y, a continuación, elija las subredes en Subredes.


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

8. Seleccione Create (Crear).

   El nuevo grupo de subredes de base de datos aparece en la lista de grupos de subredes de base de datos de la consola de RDS. Puede elegir el grupo de subredes de base de datos para ver los detalles, incluidas todas las subredes asociadas al grupo, en el panel de detalles de la parte inferior de la ventana.

Paso 3: Crear un grupo de seguridad de VPC

Antes de crear el clúster de base de datos, debe crear un grupo de seguridad de VPC para asociarlo a el clúster de base de datos. Si no crea un grupo de seguridad de VPC, puede utilizar el grupo de seguridad predeterminado cuando cree un clúster de base de datos. Para obtener instrucciones sobre cómo crear un grupo de seguridad para el clúster de base de datos, consulte Creación de un grupo de seguridad de VPC para un clúster de base de datos privada o consulte Controlar el tráfico hacia los recursos mediante grupos de seguridad en la Guía del usuario de Amazon VPC.

Paso 4: Crear la instancia de base de datos en la VPC

En este paso, se crea un clúster de base de datos y se utiliza el nombre de la VPC, el grupo de subredes de base de datos y el grupo de seguridad de VPC creados en los pasos anteriores.

> **Note**
>
> Si desea que un clúster de base de datos de la VPC sea accesible públicamente, debe activar los atributos DNS hostnames y DNS resolution de la VPC. Para obtener más información, consulte Atributos de DNS para su VPC en la Guía del usuario de Amazon VPC.

Para obtener más información sobre cómo crear un clúster de base de datos, consulte Creación de un clúster de base de datos de Amazon Aurora.

Cuando la sección Connectivity (Conectividad) se lo pida, introduzca el nombre de la VPC, el grupo de subredes de base de datos y el grupo de seguridad de la VPC.

> **Note**
>
> La actualización de VPC no se admite actualmente para los clústeres de Aurora.

Escenarios de acceso a un clúster de base de datos en una VPC

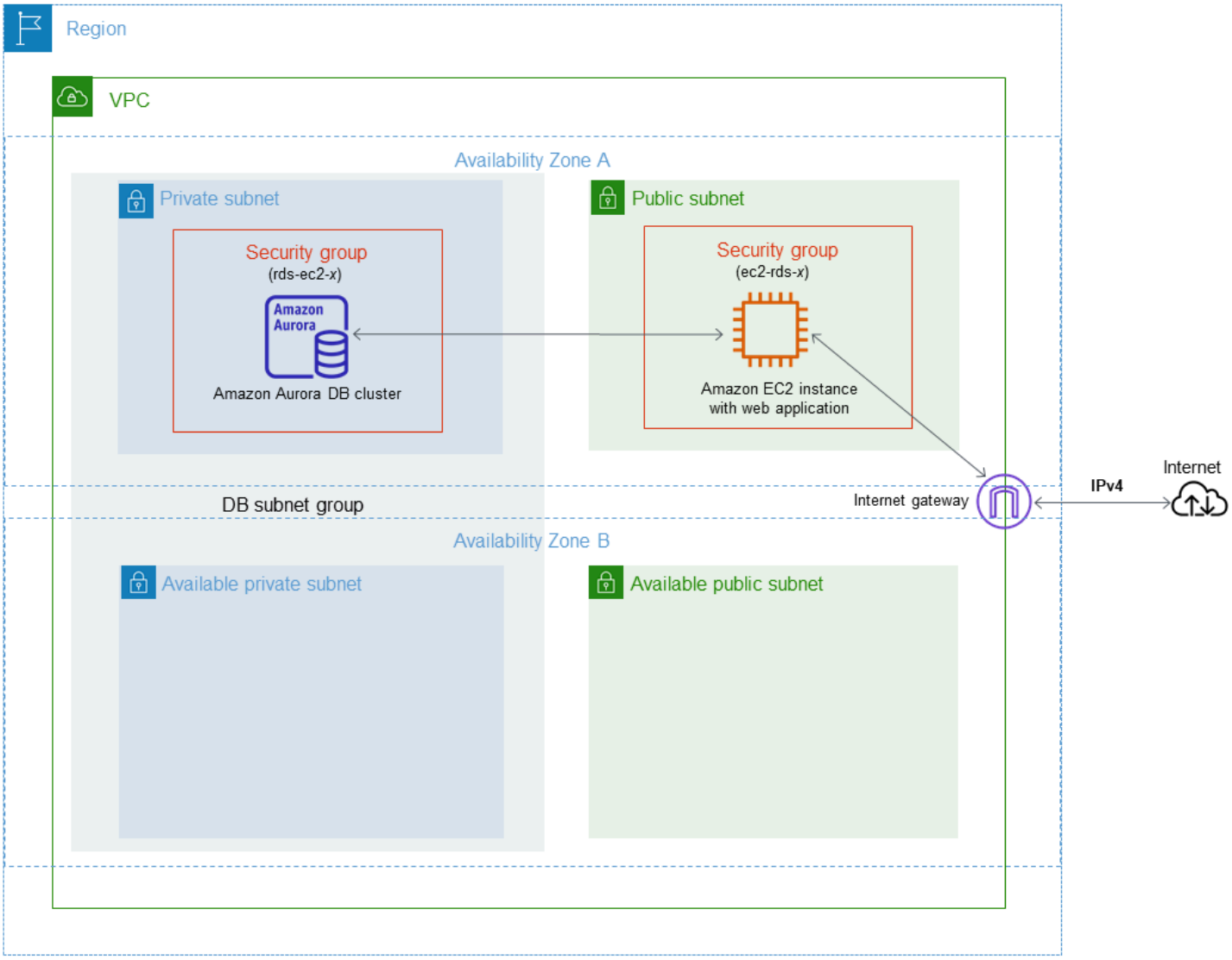
Amazon Aurora admite los siguientes escenarios para acceder a un clúster de base de datos en una VPC:

- Una instancia de Amazon EC2 de la misma VPC
- Una instancia EC2 de otra VPC
- Una aplicación cliente a través de Internet
- Una red privada

Acceso a un clúster de base de datos en una VPC desde una instancia de Amazon EC2 de la misma VPC

Un uso común de un clúster de base de datos en una VPC es compartir datos con un servidor de aplicaciones que se ejecuta en una instancia de Amazon EC2 de la misma VPC.

En el siguiente diagrama se muestra este escenario.



La forma más sencilla de administrar el acceso entre instancias EC2 e clústeres de bases de datos en la misma VPC es la siguiente:

- Cree el grupo de seguridad de VPC al que pertenecerán los clústeres de bases de datos. Este grupo de seguridad se puede utilizar para restringir el acceso a los clústeres de bases de datos. Por ejemplo, puede crear una regla personalizada para este grupo de seguridad. Esto puede permitir el acceso TCP utilizando el puerto que asignó a el clúster de base de datos cuando lo creó y una dirección IP que utiliza para acceder a el clúster de base de datos para el desarrollo u otras finalidades.
- Cree el grupo de seguridad de VPC al que pertenecerán las instancias EC2 (clientes y servidores web). Este grupo de seguridad puede, si es necesario, permitir el acceso a la instancia EC2 desde Internet a través de la tabla de enrutamiento de la VPC. Por ejemplo, puede establecer reglas en

este grupo de seguridad para permitir el acceso mediante TCP a la instancia EC2 a través del puerto 22.

- Cree reglas personalizadas en el grupo de seguridad para los clústeres de bases de datos que permitan las conexiones desde el grupo de seguridad que creó para las instancias EC2. Estas reglas podrían permitir a cualquier miembro del grupo de seguridad acceder a los clústeres de base de datos.

Hay una subred pública y privada adicional en una zona de disponibilidad independiente. Un grupo de subredes de base de datos de RDS requiere una subred en al menos dos zonas de disponibilidad. La subred adicional facilita el cambio a una implementación de instancia de base de datos multi-AZ en el futuro.

Para ver un tutorial que muestra cómo crear una VPC con subredes públicas y privadas para este escenario, consulte Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4).

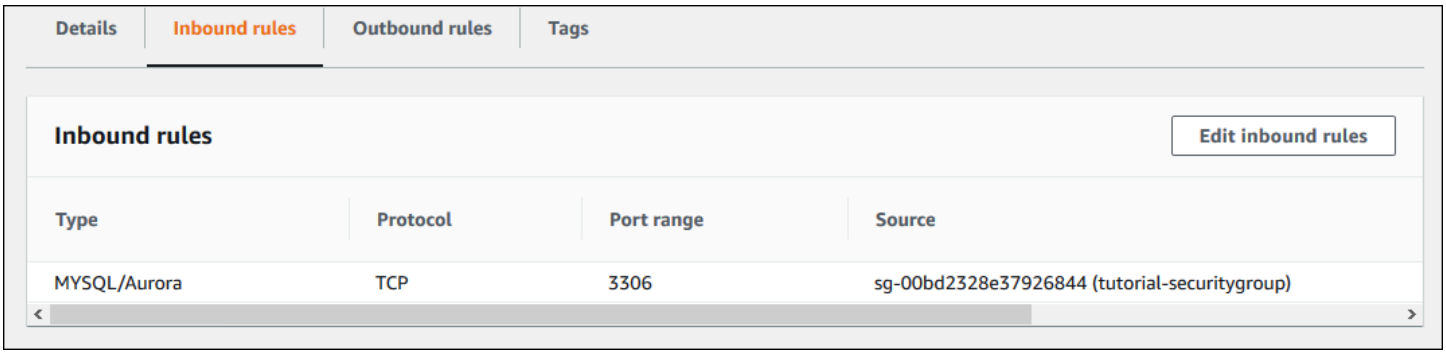
> **Tip**
>
> Puede configurar la conectividad de red entre una instancia de Amazon EC2 y un clúster de base de datos automáticamente al crear el clúster de base de datos. Para obtener más información, consulte Configurar la conectividad de red automática con una instancia de EC2 .

Para crear una regla en un grupo de seguridad de VPC que permita establecer conexiones desde otro grupo de seguridad, haga lo siguiente:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc.
2. En el panel de navegación, elija Security Groups (Grupos de seguridad).
3. Elija o cree el grupo de seguridad al que desea que puedan tener acceso los miembros de otro grupo de seguridad. En el escenario anterior, este es el grupo de seguridad que utiliza para los clústeres de base de datos. Elija la pestaña Inbound Rules (Reglas de entrada) y, a continuación, elija Edit inbound rules (Editar reglas de entrada).
4. En la página Edit inbound rules (Editar reglas de entrada), elija Add Rule (Agregar regla).
5. En Type (Tipo), elija la entrada que corresponda al puerto que utilizó al crear el clúster de base de datos, como MySQL/Aurora.

6. En el cuadro Origen, comience a escribir el ID del grupo de seguridad, que enumera los grupos de seguridad coincidentes. Elija el grupo de seguridad cuyos miembros desea que tengan acceso a los recursos protegidos por este grupo de seguridad. En el escenario anterior, este es el grupo de seguridad que utiliza para su instancia EC2.
7. Si es necesario, repita los pasos para el protocolo TCP creando una regla con Todo TCP en el campo Tipo y con el grupo de seguridad en el campo Origen. Si va a utilizar el protocolo UDP, cree una regla con All UDP (Todo UDP) en el campo Type (Tipo) y con el grupo de seguridad en el campo Source (Origen).
8. Seleccione Guardar reglas.

La siguiente pantalla muestra una regla de entrada con un grupo de seguridad para su origen.

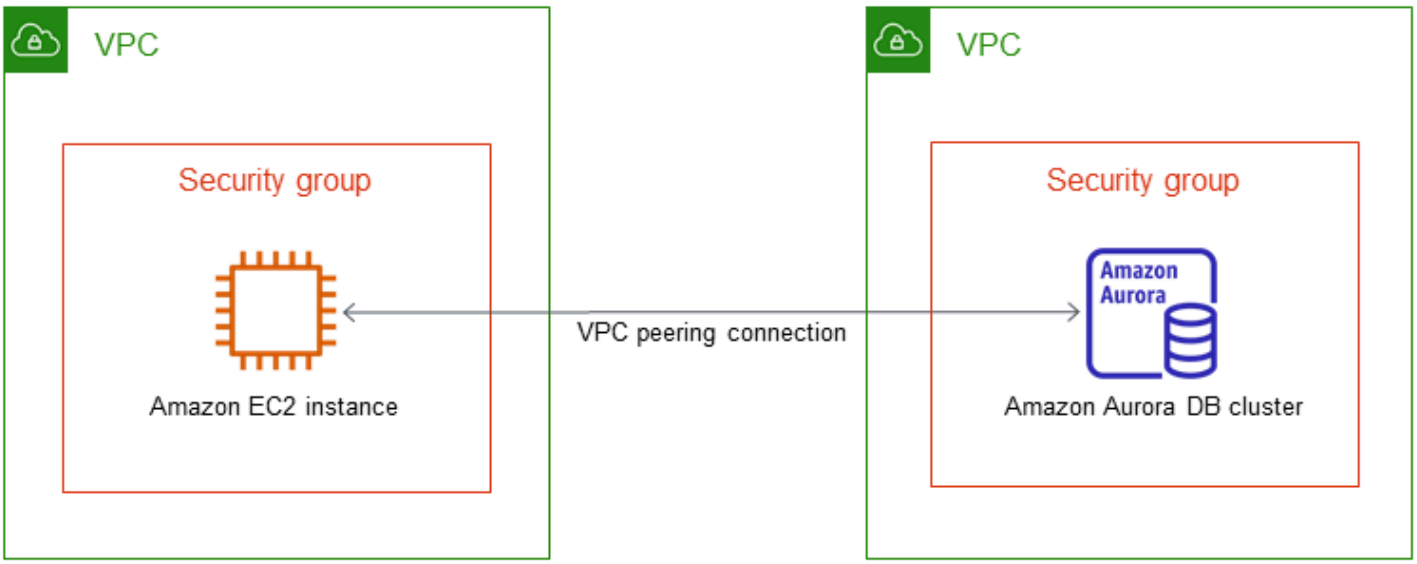


Para obtener más información sobre cómo conectarse al clúster de base de datos desde su instancia de EC2, consulte Conexión a un clúster de base de datos Amazon Aurora.

Acceso a un clúster de base de datos en una VPC desde una instancia EC2 de otra VPC

Cuando un clúster de base de datos está en una VPC que no coincide con la de la instancia EC2 que se está utilizando para obtener acceso a ella, puede usar la interconexión con VPC para obtener acceso a el clúster de base de datos.

En el siguiente diagrama se muestra este escenario.

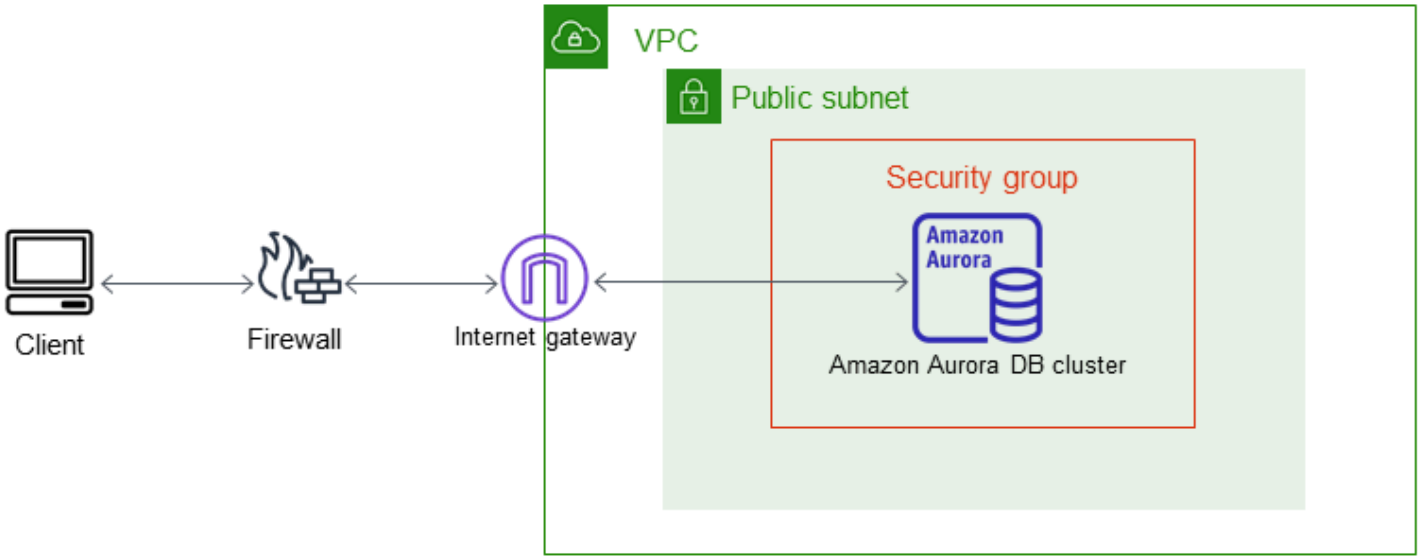


Una conexión de emparejamiento de VPC es una conexión de redes entre dos VPC que permite direccionar el tráfico entre ellas mediante direcciones IP privadas. Los recursos de ambas VPC se pueden comunicar entre sí siempre que se encuentren en la misma red. Puede crear una conexión de emparejamiento de VPC entre sus propias VPC, con una VPC de otra cuenta de AWS o con una VPC de otra Región de AWS. Para obtener más información sobre las interconexiones de VPC, consulte Interconexión con VPC en la Guía de usuario de Amazon Virtual Private Cloud.

Acceso a un clúster de base de datos en una VPC desde una aplicación cliente a través de internet

Para acceder a un clúster de base de datos en una VPC desde una aplicación cliente a través de internet, configure una VPC con una subred pública única y una puerta de enlace de Internet para permitir la comunicación a través de internet.

En el siguiente diagrama se muestra este escenario.



Recomendamos la siguiente configuración:

- Una VPC de tamaño /16 (por ejemplo, CIDR: 10.0.0.0/16). Este tamaño proporciona 65 536 direcciones IP privadas.
- Una subred de tamaño /24 (por ejemplo, CIDR: 10.0.0.0/24). Este tamaño proporciona 256 direcciones IP privadas.
- Un clúster de base de datos de Amazon Aurora que se ha asociado a la VPC y a la subred. Amazon RDS asigna una dirección IP de la subred a el clúster de base de datos.
- Una gateway de Internet que conecte la VPC a Internet y a otros productos de AWS.
- Un grupo de seguridad asociado a el clúster de base de datos. Las reglas de entrada del grupo de seguridad permiten a la aplicación cliente obtener acceso a el clúster de base de datos.

Para obtener información acerca de la creación de un clúster de base de datos en una VPC, consulte Creación de un clúster de base de datos en una VPC.

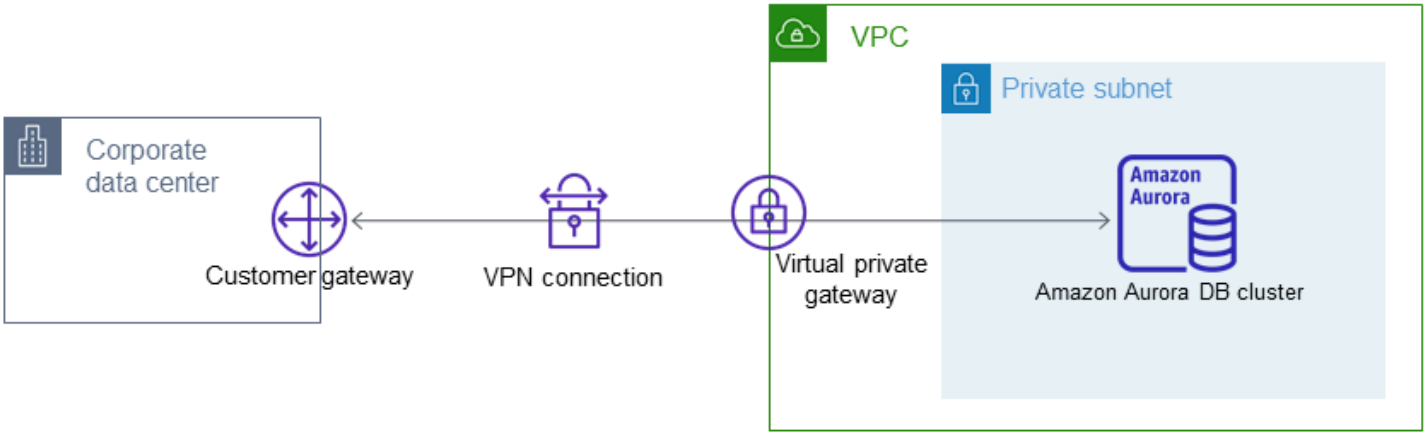
Un clúster de base de datos en una VPC a la que se accede mediante una red privada

Si su clúster de base de datos no es accesible públicamente, tiene las siguientes opciones para acceder a ella desde una red privada:

- Una conexión de Site-to-Site VPN de AWS. Para obtener más información, consulte ¿Qué es AWS Site-to-Site VPN?

- Una conexión de Direct Connect. Para obtener más información, consulte ¿Qué es Direct Connect?
- Una conexión de AWS Client VPN. Para obtener más información, consulte ¿Qué es AWS Client VPN?

El siguiente diagrama muestra un escenario con una conexión de Site-to-site VPN AWS.

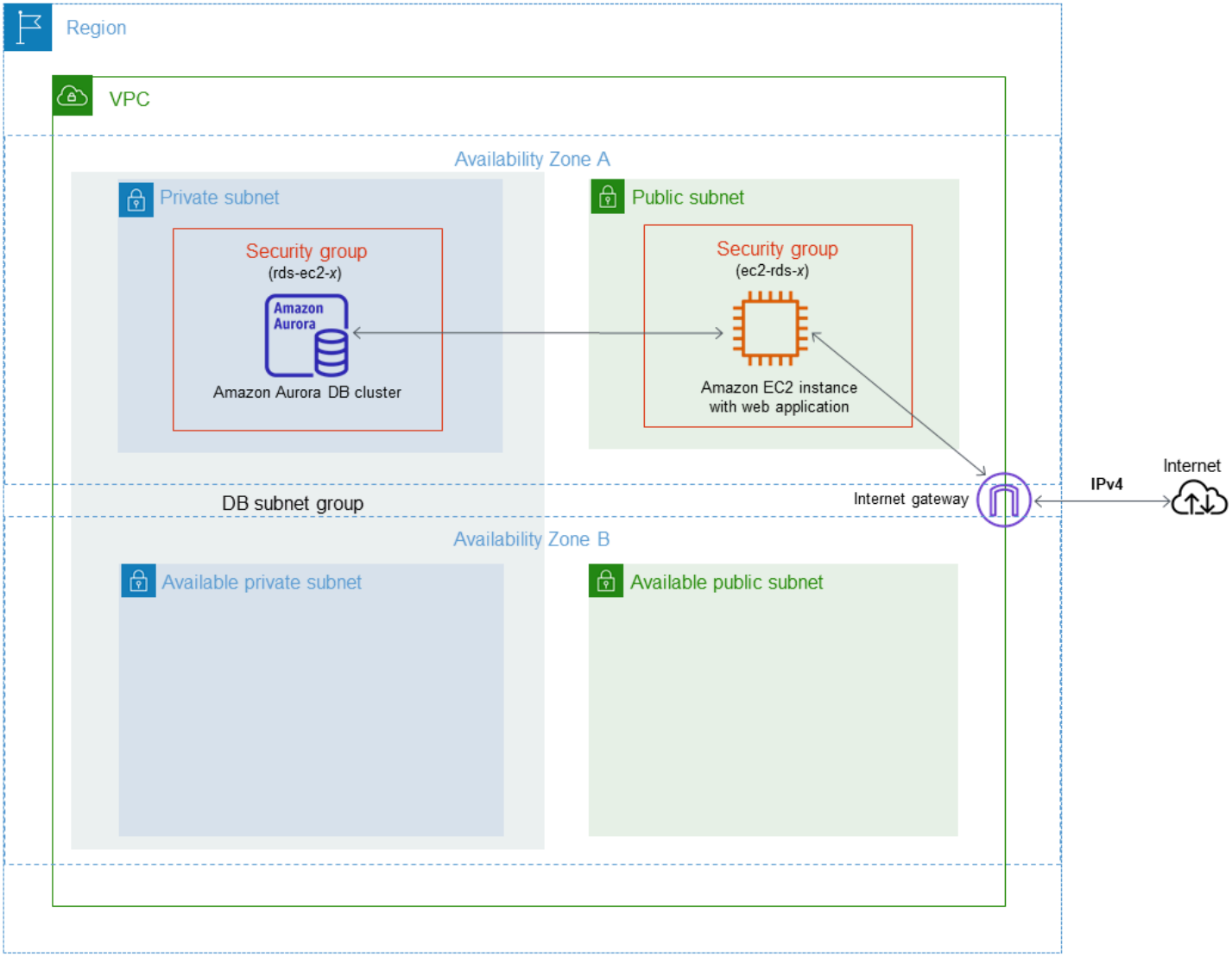


Para obtener más información, consulte Privacidad del tráfico entre redes.

Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (solo IPv4)

Un escenario común incluye un clúster de base de datos en una nube privada virtual (VPC) basada en el servicio Amazon VPC. Esta VPC comparte datos con un servidor web que se ejecuta en la misma VPC. En este tutorial se crea la VPC para este escenario.

En el siguiente diagrama se muestra este escenario. Para obtener información acerca de otros escenarios, consulte Escenarios de acceso a un clúster de base de datos en una VPC.



Su clúster de bases de datos debe estar disponible únicamente para su servidor web, y no para la Internet pública. Además, cree de una VPC con subredes públicas y privadas. El servidor web está alojado en la subred pública, para que pueda obtener acceso a la red pública de internet. El clúster

de base de datos se aloja en una subred privada. El servidor web puede conectarse a el clúster de base de datos porque se aloja en la misma VPC. Sin embargo, el clúster de base de datos no está disponible en la red pública de internet, lo que proporciona mayor seguridad.

Este tutorial configura una subred pública y privada adicional en una zona de disponibilidad independiente. En el tutorial no se utilizan estas subredes. Un grupo de subredes de base de datos de RDS requiere una subred en al menos dos zonas de disponibilidad. La subred adicional facilita la configuración de más de una instancia de base de datos de Aurora.

En este tutorial se describe la configuración de una VPC para clústeres de bases de datos de Amazon Aurora de . Para ver un tutorial que muestra cómo crear un servidor web para este escenario de la VPC, consulte Explicación: crear un servidor web y un clúster de base de datos de Amazon Aurora. Para obtener más información sobre Amazon VPC, consulte la guía de introducción de Amazon VPC y la guía del usuario de Amazon VPC.

> **Tip**
>
> Puede configurar la conectividad de red entre una instancia de Amazon EC2 y un clúster de base de datos automáticamente al crear el clúster de base de datos. La configuración de red es similar a la que se describe en este tutorial. Para obtener más información, consulte Configurar la conectividad de red automática con una instancia de EC2.

Creación de una VPC con subredes públicas y privadas

Utilice el siguiente procedimiento para crear una VPC con subredes públicas y privadas.

Para crear una VPC y las subredes

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En la esquina superior derecha de la Consola de administración de AWS, elija la región en la que desea crear la VPC. En este ejemplo se utiliza la región EE.UU. Oeste (Oregón).
3. En la esquina superior izquierda, elija VPC Dashboard (Panel de control VPC). Para comenzar a crear una VPC, elija Create VPC (Crear una VPC).
4. En Resources to create (Recursos para crear), en VPC settings (Configuración VPC), elija VPC and more (VPC y más).
5. En VPC settings (Configuración de la VPC), establezca estos valores:

- Name tag auto-generation (Generación automática de etiquetas de nombre): **`tutorial`**
- IPv4 CIDR block (Bloque de CIDR IPv4): **`10.0.0.0/16`**
- IPv6 CIDR block (Bloque de CIDR IPv6): ningún bloque de CIDR IPv6
- Tenancy (Tenencia): predeterminada
- Number of Availability Zones (AZs) (Número de zonas de disponibilidad): 2
- Customize AZs (Personalizar AZ): conserve los valores predeterminados.
- Number of public subnet (Número de subredes públicas): 2
- Number of private subnets (Número de subredes privadas): 2
- Customize subnets CIDR blocks (Personalizar bloques CIDR de subredes): conserve los valores predeterminados.
- NAT gateways ($) (Puertas de enlace NAT): ninguna
- VPC endpoints (Puntos de conexión de VPC): ninguna
- DNS options (Opciones de DNS): conserve los valores predeterminados.

6. Seleccione Creación de VPC.

Creación de un grupo de seguridad de VPC para un servidor web público

Primero debe crear un grupo de seguridad para el acceso público. Para conectarse a instancias de EC2 públicas en su VPC, añada reglas de entrada a su grupo de seguridad de VPC. Permiten que el tráfico se conecte desde Internet.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Security group name (Nombre de grupo de seguridad: **`tutorial-securitygroup`**
   - Description: **`Tutorial Security Group`**
   - VPC: elija la VPC que creó en el paso anterior, por ejemplo, vpc-***`identificador`***(tutorial-vpc)
4. Agregar reglas de entrada al grupo de seguridad

a. Determine la dirección IP que usará para conectarse a las instancias de EC2 mediante Secure Shell (SSH). Para determinar su dirección IP pública, en una ventana o pestaña distinta del navegador, puede utilizar el servicio en https://checkip.amazonaws.com. Un ejemplo de dirección IP es `203.0.113.25/32`.

   En muchos casos, puede conectarse a través de un proveedor de servicios de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, busque el rango de direcciones IP utilizadas por los ordenadores cliente.

   > ⚠ Warning
   >
   > Si utiliza `0.0.0.0/0` para el acceso SSH, permita que todas las direcciones IP accedan a sus instancias públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En entornos de producción, solo debe autorizar una dirección IP específica o un intervalo de direcciones para acceder a sus instancias mediante SSH.

b. En la sección Inbound rules (Reglas de entrada), elija Add rule (agregar regla).

c. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso SSH a la instancia de Amazon EC2. Si lo hace, puede conectarse a la instancia de Amazon EC2 para instalar el servidor web y otras utilidades. También puede conectarse a su instancia de EC2 para cargar contenido para el servidor web.

   - Tipo: **`SSH`**
   - Origen: la dirección IP o el rango de direcciones del Paso a, por ejemplo **`203.0.113.25/32`**.

d. Seleccione Add rule (Agregar regla).

e. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso HTTP al servidor web.

   - Tipo: **`HTTP`**
   - Origen: **`0.0.0.0/0`**

5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

   Anote el ID del grupo de seguridad, ya que lo necesitará más tarde en este tutorial.

Creación de un grupo de seguridad de VPC para un clúster de base de datos privada

Para que un clúster de base de datos sea privada, debe crear un segundo grupo de seguridad para el acceso privado. Para conectarse a clústeres de base de datos privada en la VPC, agregue reglas de entrada al grupo de seguridad de la VPC que permitan el tráfico solo desde el servidor web.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Security group name (Nombre de grupo de seguridad: **`tutorial-db-securitygroup`**
   - Description: **`Tutorial DB Instance Security Group`**
   - VPC: elija la VPC que creó en el paso anterior, por ejemplo, vpc-***`identificador`***(tutorial-vpc)
4. Agregar reglas de entrada al grupo de seguridad
   a. En la sección Inbound rules (Reglas de entrada), elija Add rule (gregar regla).
   b. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el tráfico de MySQL en el puerto 3306 desde la instancia de Amazon EC2. Si lo hace, podrá conectarse desde su servidor web a su clúster de base de datos. Si lo hace, puede almacenar y recuperar datos en la base de datos desde la aplicación web.
      - Tipo: **`MySQL/Aurora`**
      - Source (Origen): el identificador del grupo de seguridad tutorial-securitygroup que creó anteriormente en este tutorial, por ejemplo, sg-9edd5cfb.
5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

Creación de un grupo de subredes de base de datos

Un grupo de subredes de base de datos es una colección de subredes que se crea en una VPC y que después se asigna a los clústeres de bases de datos. Un grupo de subredes de base de datos le permite especificar una VPC específica al crear clústeres de bases de datos.

Para crear un grupo de subredes de base de datos

1. Identifique las subredes privadas de la base de datos en la VPC.

   a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.

   b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Subnets (Subredes).

   c. Observe los ID de subred de las subredes denominadas tutorial-subred-private1-us-west-2a ytutorial-subnet-private2-us-west-2b.

      Necesitará los ID de subred cuando cree el grupo de subredes de base de datos.

2. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

   Asegúrese de conectarse a la consola de Amazon RDS, no a la consola de Amazon VPC.

3. En el panel de navegación, elija Subnet groups.

4. Elija Create DB Subnet Group (Crear grupo de subredes de base de datos).

5. En la página Create DB subnet group (Crear grupo de subredes de base de datos), establezca estos valores en Subnet group details (Detalles del grupo de subredes):

   - Name: **`tutorial-db-subnet-group`**
   - Description: **`Tutorial DB Subnet Group`**
   - VPC: tutorial-vpc (vpc-***`identificador`***)

6. En la sección Agregar subredes elija las Zonas de disponibilidad y Subredes.

   Para este tutorial, elija us-west-2a y us-west-2b en Availability Zones (Zonas de disponibilidad). En Subnets (Subredes), elija las subredes privadas que identificó en el paso anterior.

7. Seleccione Crear.

   El nuevo grupo de subredes de base de datos aparece en la lista de grupos de subredes de base de datos de la consola de RDS. Puede elegir el grupo de subredes de base de datos para ver los detalles en el panel de detalles de la parte inferior de la ventana. Estos detalles incluyen todas las subredes asociadas al grupo.

> **Note**
>
> Si creó esta VPC para completar Explicación: crear un servidor web y un clúster de base de datos de Amazon Aurora, cree el clúster de la de base de datos siguiendo las instrucciones que se indican en Crear un clúster de base de datos de Amazon Aurora.

Eliminación de la VPC

Después de crear la VPC y otros recursos para este tutorial, puede eliminarlos si ya no son necesarios.

> **Note**
>
> Si agregó recursos en la VPC que creó para este tutorial, es posible que primero tenga que eliminar estos para poder eliminar la VPC. Por ejemplo, estos recursos pueden incluir instancias de Amazon EC2 o clústeres de base de datos de Amazon RDS. Para obtener más información, consulte Eliminación de la VPC en la Guía del usuario de Amazon VPC.

Para eliminar una VPC y los recursos relacionados

1. Elimine el grupo de subred de base de datos.
    a. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
    b. En el panel de navegación, elija Subnet groups.
    c. Seleccione el grupo de subred de base de datos que desea eliminar, como tutorial-db-subnet-group.
    d. Elija Eliminar y, a continuación, elija Eliminar en la ventana de confirmación.
2. Anote el ID de la VPC.
    a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
    b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione VPCs.
    c. En la lista, identifique la VPC que creó, como, por ejemplo, tutorial-vpc.
    d. Anote el ID de la VPC que ha creado. Necesitará el ID de la VPC en pasos posteriores.
3. Elimine los grupos de seguridad.

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
b. Seleccione Panel de control de VPC y, a continuación, seleccione Grupos de seguridad.
c. Seleccione el grupo de seguridad de la instancia de base de datos de Amazon RDS, como, por ejemplo, tutorial-db-securitygroup.
d. En Actions (Acciones), elija Delete security groups (Eliminar grupos de seguridad) y, a continuación, seleccione Delete (Eliminar) en la página de confirmación.
e. En la página Grupos de seguridad, seleccione el grupo de seguridad para la instancia de Amazon EC2, como, por ejemplo, tutorial-securitygroup.
f. En Actions (Acciones), elija Delete security groups (Eliminar grupos de seguridad) y, a continuación, seleccione Delete (Eliminar) en la página de confirmación.

4. Eliminación de la VPC

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione VPCs.
c. Seleccione la VPC que desea eliminar, como, por ejemplo tutorial-vpc.
d. En Actions (Acciones), elija Delete VPC (Eliminar VPC).

   La página de confirmación muestra otros recursos asociados a la VPC que también se eliminarán, incluidas las subredes asociadas a ella.
e. En la página de confirmación, introduzca **`delete`** y elija Eliminar.

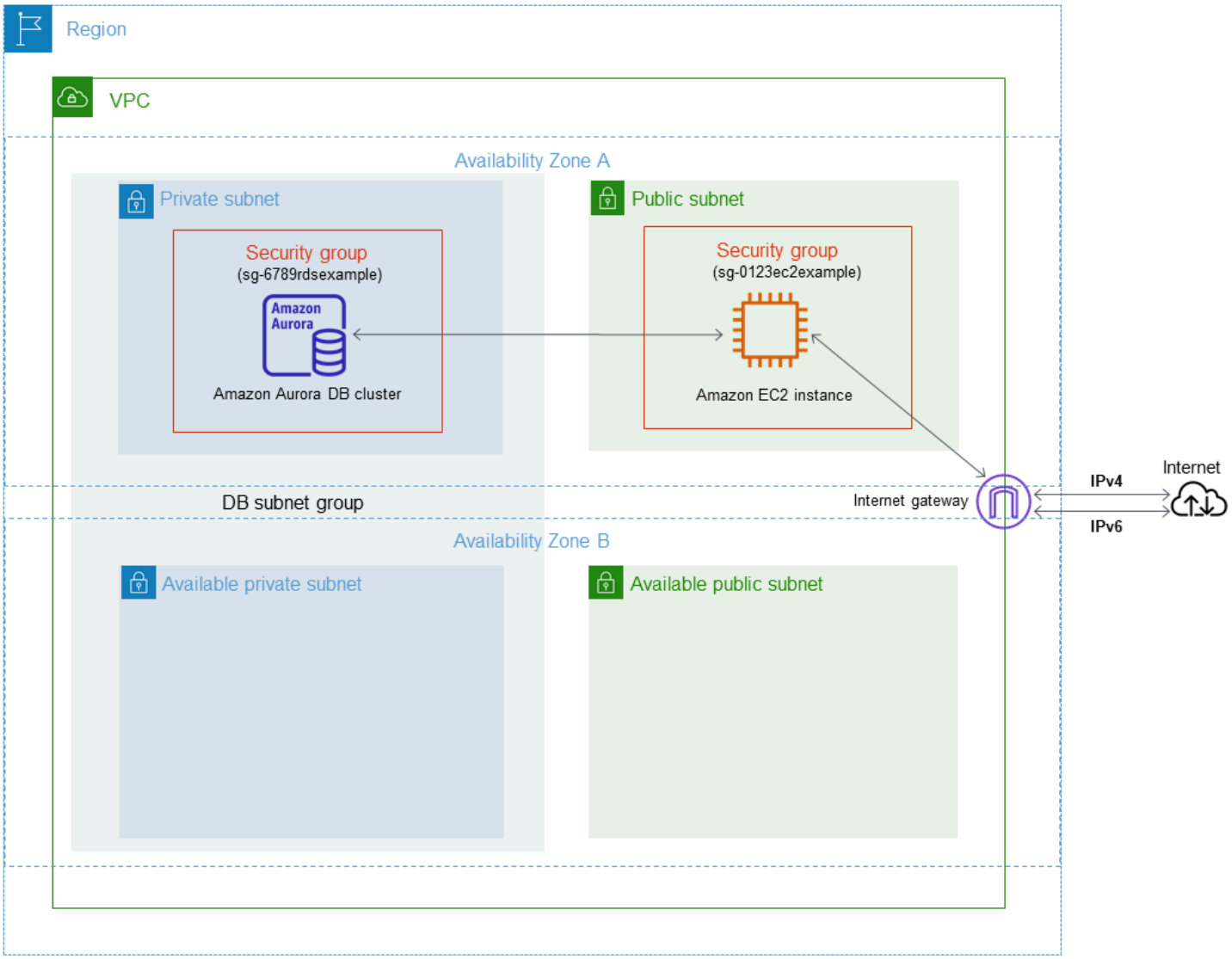
Tutorial: Creación de una VPC para utilizarla con un clúster de base de datos (modo de pila doble)

Un escenario común incluye un clúster de base de datos en una nube privada virtual (VPC) basada en el servicio Amazon VPC. Esta VPC comparte datos con una instancia de Amazon EC2 pública que se ejecuta en la misma VPC.

En este tutorial, creará la VPC para este escenario que funciona con una base de datos que se ejecuta en modo de pila doble. Modo de doble pila para permitir la conexión a través del protocolo de direccionamiento IPv6. Para obtener más información sobre el direccionamiento de IP, consulte Direccionamiento IP de Amazon Aurora.

Los clústeres de red de doble pila se admiten en la mayoría de las regiones. Para obtener más información, consulte Disponibilidad de clústeres de base de datos de red de pila doble. Para ver las limitaciones del modo de doble pila, consulte Limitaciones de clústeres de base de datos de red de pila doble.

En el siguiente diagrama se muestra este escenario.



Para obtener información acerca de otros escenarios, consulte Escenarios de acceso a un clúster de base de datos en una VPC.

Su instanciaclúster de bases de datos debe estar disponible únicamente para su instancia de Amazon EC2, y no para la Internet pública. Además, cree de una VPC con subredes públicas y privadas. La instancia de Amazon EC2 está alojada en la subred pública, para que pueda acceder a la red pública de internet. El clúster de base de datos se aloja en una subred privada. La instancia de Amazon EC2 se puede conectar a el clúster de base de datos porque se aloja en la misma VPC. Sin embargo, el clúster de base de datos no está disponible en la red pública de internet, lo que proporciona mayor seguridad.

Este tutorial configura una subred pública y privada adicional en una zona de disponibilidad independiente. En el tutorial no se utilizan estas subredes. Un grupo de subredes de base de datos

de RDS requiere una subred en al menos dos zonas de disponibilidad. La subred adicional facilita la configuración de más de una instancia de base de datos de Aurora.

Para crear un clúster de base de datos que utilice el modo de pila doble, especifique Dual-stack mode (Modo pila doble) en el ajuste Network type (Tipo de red). También puede modificar un clúster de base de datos con el mismo ajuste. Para obtener más información acerca de la creación de un clúster de base de datos, consulte Creación de un clúster de base de datos de Amazon Aurora. Para obtener más información sobre la modificación de un clúster de base de datos, consulte Modificación de un clúster de base de datos de Amazon Aurora.

En este tutorial se describe la configuración de una VPC para clústeres de bases de datos de Amazon Aurora de . Para obtener más información acerca de Amazon VPC, consulte la Guía del usuario de Amazon VPC.

Creación de una VPC con subredes públicas y privadas

Utilice el siguiente procedimiento para crear una VPC con subredes públicas y privadas.

Para crear una VPC y las subredes

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En la esquina superior derecha de la Consola de administración de AWS, elija la región en la que desea crear la VPC. En este ejemplo se utiliza la región Este de EE. UU. (Ohio).
3. En la esquina superior izquierda, elija VPC Dashboard (Panel de control VPC). Para comenzar a crear una VPC, elija Create VPC (Crear una VPC).
4. En Resources to create (Recursos para crear), en VPC settings (Configuración VPC), elija VPC and more (VPC y más).
5. Para el resto de opciones de VPC settings (Configuración de la VPC), defina estos valores:
   - Name tag auto-generation (Generación automática de etiquetas de nombre): **`tutorial-dual-stack`**
   - IPv4 CIDR block (Bloque de CIDR IPv4): **`10.0.0.0/16`**
   - IPv6 CIDR block (Bloque de CIDR IPv6): bloque de CIDR IPv6 proporcionado por Amazon
   - Tenancy (Tenencia): predeterminada
   - Number of Availability Zones (AZs) (Número de zonas de disponibilidad): 2
   - Customize AZs (Personalizar AZ): conserve los valores predeterminados.
   - Number of public subnet (Número de subredes públicas): 2

- Number of private subnets (Número de subredes privadas): 2
- Customize subnets CIDR blocks (Personalizar bloques CIDR de subredes): conserve los valores predeterminados.
- NAT gateways ($) (Puertas de enlace NAT): ninguna
- Egress only internet gateway (Puerta de enlace de internet solo de salida): no
- VPC endpoints (Puntos de conexión de VPC): ninguna
- DNS options (Opciones de DNS): conserve los valores predeterminados.

> **Note**
>
> Amazon RDS requiere al menos dos subredes en dos zonas de disponibilidad diferentes para admitir implementaciones de instancias de base de datos multi-AZ. En este tutorial se crea una implementación Single-AZ, pero el requisito facilita la conversión a una implementación de instancia de base de datos Multi-AZ en el futuro.

6. Seleccione Crear VPC.

Para crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública

Primero debe crear un grupo de seguridad para el acceso público. Para conectarse a instancias EC2 públicas, añada reglas de entrada al grupo de seguridad de la VPC que permitan el tráfico para las conexiones desde internet.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Security group name (Nombre de grupo de seguridad: **`tutorial-dual-stack-securitygroup`**
   - Description: **`Tutorial Dual-Stack Security Group`**

- VPC: elija la VPC que creó en el paso anterior, por ejemplo, vpc-***identificador***(tutorial-dual-stack-vpc)

4. Agregar reglas de entrada al grupo de seguridad

   a. Determine la dirección IP que usará para conectarse a las instancias de EC2 mediante Secure Shell (SSH).

      Un ejemplo de dirección del protocolo de internet versión 4 (IPv4) es `203.0.113.25/32`. Un ejemplo de rango de direcciones del protocolo de internet versión 6 (IPv6) es `2001:db8:1234:1a00::/64`.

      En muchos casos, puede conectarse a través de un proveedor de internet (ISP) o protegido por un firewall sin una dirección IP estática. Si es así, busque el rango de direcciones IP utilizadas por los ordenadores cliente.

      > ⚠ Warning
      >
      > Si utiliza `0.0.0.0/0` para IPv4 o `::0` para IPv6, permitirá que todas las direcciones IP tengan acceso a las instancias públicas mediante SSH. Este método es aceptable para un periodo de tiempo corto en un entorno de prueba, pero no es seguro en entornos de producción. En los entornos de producción, debe autorizar el acceso a sus instancias únicamente a una dirección IP o a un rango de direcciones IP específicos.

   b. En la sección Inbound rules (Reglas de entrada), elija Add rule (agregar regla).

   c. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el acceso Secure Shell (SSH) a la instancia de Amazon EC2. Si lo hace, podrá conectarse a la instancia de EC2 para instalar clientes SQL y otras aplicaciones. Especifique una dirección IP para permitir poder acceder a su instancia de EC2:

      - Tipo: **`SSH`**
      - Origen: la dirección IP o el rango del paso a. Un ejemplo de dirección IP IPv4 es **`203.0.113.25/32`**. Un ejemplo de dirección IP IPv6 es **`2001:DB8::/32`**.

5. Para crear el grupo de seguridad, elija Create security group (Crear grupo de seguridad).

   Anote el ID del grupo de seguridad, ya que lo necesitará más tarde en este tutorial.

Creación de un grupo de seguridad de VPC para un clúster de base de datos privada

Para que un clúster de base de datos sea privada, debe crear un segundo grupo de seguridad para el acceso privado. Para conectarse a clústeres de base de datos privadas en su VPC, añada reglas de entrada a su grupo de seguridad de VPC. Estos permiten el tráfico de su instancia de Amazon EC2 solamente.

Para crear un grupo de seguridad de VPC

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Elija VPC Dashboard (Panel VPC), seguido de Security Groups (Grupos de seguridad) y, por último, Create Security Group (Crear grupo de seguridad).
3. En la página Create Security Group (Crear grupo de seguridad), establezca estos valores:
   - Security group name (Nombre de grupo de seguridad: **`tutorial-dual-stack-db-securitygroup`**
   - Description: **`Tutorial Dual-Stack DB Instance Security Group`**
   - VPC: elija la VPC que creó en el paso anterior, por ejemplo, vpc-***`identificador`***(tutorial-dual-stack-vpc)
4. Agregar reglas de entrada al grupo de seguridad:
   a. En la sección Inbound rules (Reglas de entrada), elija Add rule (gregar regla).
   b. Establezca los siguientes valores para la regla de entrada nueva con objeto de permitir el tráfico de MySQL en el puerto 3306 desde la instancia de Amazon EC2. Si lo hace, podrá conectarse desde su instancia de EC2 a su clúster de base de datos. Esto significa que puede enviar datos desde la instancia de EC2 a la base de datos.
      - Type (Tipo): MySQL/Aurora
      - Source (Origen): identificador del grupo de seguridad tutorial-dual-stack-securitygroup que creó anteriormente en este tutorial, por ejemplo, sg-9edd5cfb.
5. Para crear el grupo de seguridad, elija Crear grupo de seguridad.

Creación de un grupo de subredes de base de datos

Un grupo de subredes de base de datos es una colección de subredes que se crea en una VPC y que después se asigna a los clústeres de bases de datos. Un grupo de subredes de base de datos le permite especificar una VPC específica al crear clústeres de bases de datos. Para crear un grupo de

subredes de base de datos que sea compatible con DUAL, todas las subredes deben ser compatibles con DUAL. Para que sea compatible con DUAL, una subred debe tener asociado un CIDR IPv6.

Para crear un grupo de subredes de base de datos

1. Identifique las subredes privadas de la base de datos en la VPC.

   a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.

   b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Subnets (Subredes).

   c. Anote los ID de subred de las subredes denominadas tutorial-subred-dual-stack-private1-us-west-2a y tutorial-subred-dual-stack-private2-us-west-2b.

      Necesitará los ID de subred cuando cree el grupo de subredes de base de datos.

2. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.

   Asegúrese de conectarse a la consola de Amazon RDS, no a la consola de Amazon VPC.

3. En el panel de navegación, elija Subnet groups.

4. Elija Create DB Subnet Group (Crear grupo de subredes de base de datos).

5. En la página Create DB subnet group (Crear grupo de subredes de base de datos), establezca estos valores en Subnet group details (Detalles del grupo de subredes):

   - Name: **`tutorial-dual-stack-db-subnet-group`**
   - Description: **`Tutorial Dual-Stack DB Subnet Group`**
   - VPC: tutorial-dual-stack-vpc (vpc-***`identificador`***)

6. En la sección Add subnets (Agregar subredes) elija las Availability Zones (Zonas de disponibilidad) y las Subnets (Subredes).

   Para este tutorial, elija us-east-2a y us-east-2b en Availability Zones (Zonas de disponibilidad). En Subnets (Subredes), elija las subredes privadas que identificó en el paso anterior.

7. Seleccione Create (Crear).

El nuevo grupo de subredes de base de datos aparece en la lista de grupos de subredes de base de datos de la consola de RDS. Puede elegir el grupo de subredes de base de datos para ver la información detallada, que incluye los protocolos de direcciones compatibles y todas las subredes asociadas al grupo y al tipo de red admitidos por el grupo de subredes de base de datos.

Crear una instancia de Amazon EC2 en el modo de pila doble

Para crear una instancia de Amazon EC2, siga las instrucciones indicadas en Lanzar una instancia con el nuevo asistente de inicialización de instancias en la Guía del usuario de Amazon EC2.

En la página Configure Instance Details (Configurar detalles de instancia), defina estos valores y mantenga los demás con sus valores predeterminados:

- Red: elija una VPC existente con las subredes pública y privada, como tutorial-dual-stack-vpc (vpc-*identificador*) creada en Creación de una VPC con subredes públicas y privadas
- Subnet (Subred): elija una subred pública existente, como subnet-***identificador*** | tutorial-dual-stack-subnet-public1-us-east-2a | us-east-2a, creada en Para crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública.
- Auto-assign Public IP (Asignar automáticamente IP pública): elija Enable (Habilitar).
- Auto-assign IPv6 IP (Asignar automáticamente IP IPv6): elija Enable (Habilitar).
- Firewall (security groups) (Firewall [grupos de seguridad]): elija Select an existing security group (Seleccionar un grupo de seguridad existente).
- Common security groups (Grupos de seguridad comunes): elija un grupo de seguridad existente, como el `tutorial-securitygroup` creado en Para crear un grupo de seguridad de la VPC para una instancia de Amazon EC2 pública. Asegúrese de que el grupo de seguridad que elija incluya reglas de entrada para acceso Secure Shell (SSH) y HTTP.

Crear un clúster de base de datos en el modo de pila doble

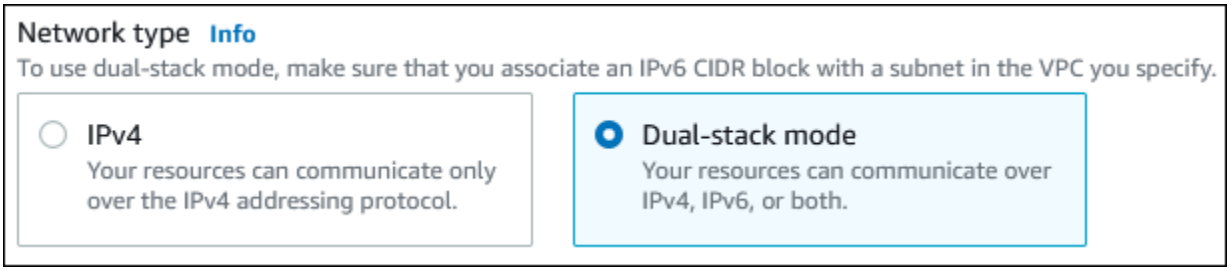
En este paso, debe crear un clúster de base de datos que se ejecute en modo de pila doble.

Para crear una instancia de base de datos

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En la esquina superior derecha de la consola, elija la Región de AWS en la que desea crear el clúster de base de datos. En este ejemplo se utiliza la región Este de EE. UU. (Ohio).
3. En el panel de navegación, seleccione Databases (Bases de datos).
4. Elija Create database (Crear base de datos).

5. En la página Create database (Crear base de datos), asegúrese de que la opción Standard Create (Creación estándar) esté seleccionada y luego, elija el tipo de motor de base de datos Aurora MySQL.
6. En la sección Conectivity (Conectividad), establezca estos valores:

   - Network type (Tipo de red): elija Dual-stack mode (Modo de pila doble)

   - Virtual private cloud (VPC) (Nube privada virtual [VPC]): elija una VPC existente con las subredes pública y privada, como tutorial-dual-stack-vpc (vpc-*identificador*) creada en Creación de una VPC con subredes públicas y privadas

     La VPC debe tener subredes en diferentes zonas de disponibilidad.
   - DB subnet group (Grupo de subredes de base de datos): grupo de subredes de base de datos para la VPC, como tutorial-dual-stack-db-subnet-group, creado en Creación de un grupo de subredes de base de datos
   - Public access (Acceso público): elija No.
   - VPC security group (firewall) (Grupo de seguridad de VPC [firewall]): seleccione Choose existing (Elegir existente).
   - Existing VPC security groups (Grupos de seguridad de la VPC existentes): elija un grupo de seguridad de la VPC existente configurado para el acceso público, como tutorial-dual-stack-db-securitygroup creado en Creación de un grupo de seguridad de VPC para un clúster de base de datos privada.

     Elimine otros grupos de seguridad, como el grupo de seguridad predeterminado, seleccionando la X asociada con cada uno de ellos.
   - Availability Zone (Zona de disponibilidad): elija us-west-2a.

     Para evitar el tráfico entre zonas de disponibilidad, asegúrese de que la instancia de base de datos y la instancia de EC2 estén en la misma zona de disponibilidad.

7. En el resto de secciones, especifique los ajustes de configuración del clúster de base de datos. Para obtener más información acerca de cada ajuste, consulte Configuración de clústeres de bases de datos de Aurora.

Conectarse a la instancia de Amazon EC2 y a el clúster de base de datos

Después de crear el clúster de base de datos y la instancia de Amazon EC2 en el modo de pila doble, puede conectarse a cada una mediante el protocolo IPv6. Para conectarse a una instancia de Amazon EC2 mediante el protocolo IPv6, siga las instrucciones indicadas en Conexión con la instancia de Linux en la Guía del usuario de Amazon EC2.

Para conectarse al clúster de base de datos de Aurora MySQL desde la instancia de Amazon EC2, siga las instrucciones de Conectarse a un clúster de base de datos de Aurora MySQL.

Eliminación de la VPC

Después de crear la VPC y otros recursos para este tutorial, puede eliminarlos si ya no son necesarios.

Si agregó recursos en la VPC que creó para este tutorial, es posible que primero tenga que eliminar estos para poder eliminar la VPC. Algunos ejemplos de recursos son las instancias de Amazon EC2 o los clústeres de base de datos. Para obtener más información, consulte Eliminación de la VPC en la Guía del usuario de Amazon VPC.

Para eliminar una VPC y los recursos relacionados

1. Elimine el grupo de subredes de base de datos:
    a. Abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
    b. En el panel de navegación, elija Subnet groups.
    c. Seleccione el grupo de subredes de base de datos a eliminar, como tutorial-db-subnet-group.
    d. Elija Delete (Eliminar) y, a continuación, elija Delete (Eliminar) en la ventana de confirmación.
2. Anote el ID de la VPC:
    a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
    b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione VPCs.

c. En la lista, identifique la VPC que creó, como, por ejemplo, tutorial-dual-stack-vpc.

d. Anote el ID de la VPC que ha creado. Necesitará el ID de la VPC en los pasos subsiguientes.

3. Elimine los grupos de seguridad:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.

b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione Security Groups (Grupos de seguridad).

c. Seleccione el grupo de seguridad de la instancia de base de datos de Amazon RDS, como, por ejemplo, tutorial-dual-stack-db-securitygroup.

d. En Actions (Acciones), elija Delete security groups (Eliminar grupos de seguridad) y, a continuación, seleccione Delete (Eliminar) en la página de confirmación.

e. En la página Security Groups (Grupos de seguridad), seleccione el grupo de seguridad para la instancia de Amazon EC2, como, por ejemplo, tutorial-dual-stack-securitygroup.

f. En Actions (Acciones), elija Delete security groups (Eliminar grupos de seguridad) y, a continuación, seleccione Delete (Eliminar) en la página de confirmación.

4. Elimine la puerta de enlace NAT:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.

b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione NAT Gateways (Puertas de enlace NAT).

c. Seleccione la puerta de enlace NAT de la VPC que creó. Utilice el ID de VPC para identificar la puerta de enlace NAT correcta.

d. En Actions (Acciones), seleccione Delete NAT gateway (Eliminar puerta de enlace NAT).

e. En la página de confirmación, introduzca **`delete`** y elija Eliminar.

5. Elimine la VPC:

a. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.

b. Seleccione VPC Dashboard (Panel de control de VPC) y, a continuación, seleccione VPCs.

c. Seleccione la VPC que desea eliminar, como, por ejemplo tutorial-dual-stack-vpc.

d. En Actions (Acciones), elija Delete VPC (Eliminar VPC).

La página de confirmación muestra otros recursos asociados a la VPC que también se eliminarán, incluidas las subredes asociadas a ella.

e. En la página de confirmación, introduzca **`delete`** y elija Eliminar.

6. Libere las direcciones IP elásticas:

   a. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.

   b. Seleccione EC2 Dashboard (Panel de EC2) y, a continuación, seleccione Elastic IPs (Direcciones IP elásticas).

   c. Seleccione la dirección IP elástica que desea liberar.

   d. Desde Actions (Acciones), elija Release Elastic IP addresses (Liberar direcciones IP elásticas).

   e. En la página de confirmación, seleccione Release (Liberar).

Cuotas y restricciones para Amazon Aurora

A continuación puede ver una descripción de las cuotas de recursos y las restricciones de nomenclatura para Amazon Aurora.

Temas

- Cuotas en Amazon Aurora
- Restricciones de la nomenclatura en Amazon Aurora
- Límites de tamaño de Amazon Aurora

Cuotas en Amazon Aurora

Cada cuenta de AWS tiene cuotas, para cada región de AWS, según la cantidad de recursos de Amazon Aurora que se pueden crear. Una vez que se alcance la cuota de un recurso, las llamadas adicionales para crear ese recurso dejan de funcionar con una excepción.

En la siguiente tabla se enumeran los recursos y las cuotas por AWS región.

Nombre	Valor predeterminado	Ajusta	Descripción
Autorizaciones por grupo de seguridad de base de datos	Cada región admitida: 20	No	Número de autorizaciones de grupos de seguridad por grupo de seguridad de base de datos
Puntos de enlace personalizados por clúster de base de datos	Cada región admitida: 5	Sí	El número máximo de puntos de conexión personalizados que puede crear por clúster de base de datos de Aurora en esta cuenta en la región actual. Este valor refleja el número más alto de puntos de

Nombre	Valor predeterminado	Ajusta	Descripción
			conexión personalizados en un clúster de base de datos de la cuenta. Es posible que otros clústeres de bases de datos de la cuenta tengan un número inferior de puntos de conexión personalizados.
Versiones del motor personalizadas	Cada región admitida: 40	Sí	Número máximo de versiones de motor personalizado permitidas en esta cuenta en la región actual
Grupos de parámetros de clúster de bases de datos	Cada región admitida: 50	Sí	Número máximo de grupos de parámetros de clúster de base de datos
Clústeres de base de datos	Cada región admitida: 40	Sí	Número máximo de clústeres de Aurora permitido en esta cuenta en la región actual
Instancias de base de datos	ap-south-1: 20 Cada una de las demás regiones admitidas: 40	Sí	Número máximo de instancias de base de datos permitidas en esta cuenta en la región actual

Nombre	Valor predeterminado	Ajusta	Descripción
Grupos de particiones de bases de datos	Cada región admitida: 5	Sí	Número máximo de grupos de particiones de bases de datos para Base de datos ilimitada de Aurora para esta cuenta en la región actual
Grupos de subred de base de datos	ap-south-1: 20 Cada una de las demás regiones admitidas: 50	Sí	Número máximo de grupos de subredes de base de datos
Tamaño del cuerpo de la solicitud HTTP de la API de datos	Cada región admitida: 4 megabytes	No	Tamaño máximo permitido para el cuerpo de la solicitud HTTP.
Pares de secreto de clúster simultáneos máximos de la API de datos	Cada región admitida: 30	No	El número máximo de pares únicos de clústeres y secretos de base de datos de Aurora sin servidor v1 en solicitudes simultáneas a la API de datos para esta cuenta y la región de AWS actual.

Nombre	Valor predeterminado	Ajusta	Descripción
Solicitudes simultáneas máximas de la API de datos	Cada región admitida: 500	No	El número máximo de solicitudes de la API de datos a un clúster de bases de datos de Aurora sin servidor v1 que utilizan el mismo secreto y se pueden procesar al mismo tiempo. Las solicitudes adicionales se ponen en cola y se procesan a medida que se completan las solicitudes en proceso.
Tamaño máximo del conjunto de resultados de la API de datos	Cada región admitida: 1 megabyte	No	Tamaño máximo del conjunto de resultados de la base de datos que puede devolver la API de datos.
Tamaño máximo de la API de datos de la cadena de respuesta JSON	Cada región admitida: 10 megabytes	No	Tamaño máximo de la cadena de respuesta JSON simplificada que devuelve la API de datos de RDS.

Nombre	Valor predeterminado	Ajusta	Descripción
Solicitudes de la API de datos por segundo	Cada región admitida: 1000 por segundo	No	El número máximo de solicitudes a la API de datos por segundo permitido en esta cuenta en la región de AWS actual. Esta cuota solo se aplica a los clústeres de Amazon Aurora sin servidor v1.
Suscripciones de eventos	Cada región admitida: 20	Sí	Número máximo de suscripciones a eventos
Roles de IAM por clúster de bases de datos	Cada región admitida: 5	Sí	Número máximo de roles de IAM asociados con un clúster de base de datos
Roles de IAM por instancia de base de datos	Cada región admitida: 5	Sí	Número máximo de roles de IAM asociados con una instancia de base de datos
Integraciones	Cada región admitida: 100	No	El número máximo de integraciones permitidas en esta cuenta para la región de AWS actual
Instantánea de clúster de bases de datos manual	Cada región admitida: 100	Sí	Número máximo de instantáneas de clúster de base de datos manuales

Nombre	Valor predeterminado	Ajusta	Descripción
Instantáneas de la instancia de base de datos manuales	Cada región admitida: 100	Sí	Número máximo de instantáneas de instancia de base de datos manuales
Grupos de opciones	Cada región admitida: 20	Sí	Número máximo de grupos de opciones
Grupos de parámetros	ap-south-1: 20 Cada una de las demás regiones admitidas: 50	Sí	Número máximo de grupos de parámetros
Proxies	Cada región admitida: 20	Sí	Número máximo de proxies permitidos en esta cuenta en la región AWS actual
Réplicas de lectura por principal	Cada región admitida: 15	Sí	El número máximo de réplicas de lectura por instancia principal de base de datos. Esta cuota no se puede ajustar para Amazon Aurora.
Instancias de base de datos reservadas	ap-south-1: 20 Cada una de las demás regiones admitidas: 40	Sí	Número máximo de instancias de base de datos reservadas permitidas en esta cuenta en la región AWS actual

Nombre	Valor predeterminado	Ajusta	Descripción
Grupos de seguridad	ap-south-1: 20 Cada una de las demás regiones compatibles: 25	Sí	Número máximo de grupos de seguridad de base de datos
Subredes por grupo de subredes de base de datos	Cada región admitida: 20	No	Número máximo de subredes por grupo de subredes de base de datos
Almacenamiento total para todas las instancias de base de datos	Cada región admitida: 100 000	Sí	El almacenamiento total máximo (en GB) en volúmenes de EBS para todas las instancias de base de datos de Amazon RDS sumadas. Esta cuota no se aplica a Amazon Aurora, que tiene un volumen máximo de clúster de 128 TiB para cada clúster de base de datos.

Note

De forma predeterminada, puede tener un total de 40 instancias de base de datos. Dentro de esta cuota se tienen en cuenta las instancias de base de datos de RDS, las instancias de base de datos de Aurora, las instancias de Amazon Neptune y las instancias de Amazon DocumentDB.

Si la aplicación requiere más instancias de base de datos, puede solicitar instancias de base de datos adicionales; solo tiene que abrir la consola de Service Quotas. En el panel de navegación, elija serviciosAWS. Elija Amazon Relational Database Service (Amazon RDS), elija una cuota y siga las instrucciones para solicitar un aumento de cuota. Para obtener más

información, consulte este tema acerca de cómo solicitar un aumento de cuota en la Guía del usuario de Service Quotas.
Para RDS para Oracle, puede crear hasta 15 réplicas de lectura por instancia de base de datos de origen en cada región, pero le recomendamos limitar las réplicas a 5 para minimizar el retraso de la replicación.
Las copias de seguridad administradas por AWS Backup se consideran instantáneas de clúster de bases de datos manuales, pero no cuentan para la cuota de instantáneas de clúster manuales. Para obtener más información sobre AWS Backup, consulte la Guía para desarrolladores de AWS Backup.

Si utiliza cualquier operación de la API de RDS y supera la cuota predeterminada de la cantidad de llamadas por segundo, la API de Amazon RDS genera un error como el siguiente.

ClientError: An error occurred (ThrottlingException) when calling the *API_name* operation: Rate exceeded.

En este caso, reduzca la cantidad de llamadas por segundo. La cuota está destinada a cubrir la mayoría de los casos de uso. Si se necesitan cuotas superiores, puede solicitar aumentos de cuota mediante una de las siguientes opciones:

- Desde la consola, abra la consola Service Quotas.
- Desde la AWS CLI, utilice el comando request-service-quota-increase de la AWS CLI.

Para obtener más información, consulte la Guía del usuario de Service Quotas.

Restricciones de la nomenclatura en Amazon Aurora

Las restricciones de la nomenclatura en Amazon Aurora son las siguientes:

- Identificador de clúster de base de datos:
  - Deben contener entre 1 y 63 caracteres alfanuméricos o guiones.
  - El primer carácter debe ser una letra.
  - No se pueden incluir dos guiones consecutivos ni acabar con guion.
  - Debe ser único para todas las instancias de base de datos por AWS cuenta, por AWS región.

- Nombre inicial de la base de datos: las restricciones de nombre de base de datos son distintas en Aurora MySQL y PostgreSQL. Para obtener más información, consulte la configuración disponible al crear cada clúster de bases de datos.
- Nombre de usuario maestro: las restricciones en los nombres de usuario maestros son distintas para cada motor de base de datos. Para obtener más información, consulte la configuración disponible al crear cada clúster de base de datos.
- Contraseña maestra:
  - La contraseña del usuario maestro de base de datos puede ser cualquier carácter ASCII imprimible excepto `/`, `'`, `"`, `@` o un espacio.
  - La contraseña puede contener el siguiente número de caracteres ASCII imprimibles, según el motor de base de datos.
    - Aurora MySQL: 8–41
    - Aurora PostgreSQL: 8–99
- Grupo de parámetros de base de datos:
  - Deben incluir entre 1 y 255 caracteres alfanuméricos.
  - El primer carácter debe ser una letra.
  - Los guiones están permitidos, pero el nombre no puede terminar por un guion o contener dos guiones seguidos.
- Grupo de subred de base de datos:
  - Debe contener entre 1 y 255 caracteres.
  - Se permiten los caracteres alfanuméricos, guiones, guiones bajos y puntos.

Límites de tamaño de Amazon Aurora

Límites de tamaño de almacenamiento

El tamaño máximo del volumen del clúster de Aurora varía según la versión del motor:

Máximo de TiB de 256:

- Versiones de Aurora PostgreSQL:
  - 17.5 y todas las versiones posteriores
  - 16.9 y versiones posteriores
  - 15.13 y versiones posteriores
- Aurora MySQL versión 3.10 (compatible con MySQL 8.0.42) y versiones posteriores

Máximo de TiB de 128:

- Todas las versiones anteriores de Aurora PostgreSQL
- Todas las versiones disponibles de Aurora MySQL 3, Aurora MySQL versión 2 y versiones 2.09 y posteriores

Para obtener más información sobre el escalado del almacenamiento automático, consulte Cómo cambia automáticamente el tamaño del almacenamiento de Aurora.

Para supervisar el espacio de almacenamiento restante, puede utilizar la métrica `AuroraVolumeBytesLeftTotal`. Para obtener más información, consulte Métricas de nivel de clúster para Amazon Aurora.

Límites de tamaño de tabla de SQL

Para un clúster de bases de datos de Aurora MySQL, el tamaño de tabla máximo es 64 tebibytes (TiB). Para un clúster de bases de datos de Aurora PostgreSQL, el tamaño de tabla máximo es 32 tebibytes (TiB). Le recomendamos que siga estas prácticas recomendadas de diseño de tabla, como la partición de tablas grandes.

Límites de identificadores de espacio de tabla

El ID de espacio de tablas máximo para Aurora MySQL es 2147483647. Si crea y elimina tablas con frecuencia, asegúrese de conocer los identificadores de sus espacios de tablas y planifique utilizar volcados lógicos. Para obtener más información, consulte Migración lógica de MySQL a Amazon Aurora MySQL mediante mysqldump.

Solución de problemas de Amazon Aurora

Utilice las siguientes secciones como ayuda para solucionar los problemas que puedan presentarse con instancias de base de datos en Amazon RDS y Amazon Aurora.

Temas

- No puede conectarse a la instancia de base de datos de Amazon RDS
- Problemas de seguridad de Amazon RDS
- Restablecimiento de la contraseña del propietario de la instancia de base de datos
- Interrupción o reinicio de una instancia de base de datos de Amazon RDS
- Los cambios de parámetros de base de datos de Amazon RDS no surten efecto
- Problemas de memoria que se puede liberar en Amazon Aurora
- Problemas de replicación de Amazon Aurora MySQL

Para obtener información sobre los problemas de depuración del uso de la API de Amazon RDS, consulte Solución de problemas de aplicaciones en Aurora.

No puede conectarse a la instancia de base de datos de Amazon RDS

Cuando no puede conectarse a una instancia de base de datos, estas suelen ser las causas habituales:

- Reglas de entrada: las reglas de acceso impuestas por el firewall local y las direcciones IP a las que autorizó el acceso a la instancia de base de datos podrían no coincidir. Lo más probable es que el problema se encuentre en las reglas de entrada de su grupo de seguridad.

  De forma predeterminada, las instancias de base de datos no permiten el acceso. El acceso se concede a través de un grupo de seguridad asociado a la VPC que permite el tráfico de entrada y salida de la instancia de base de datos. Si es necesario, agregue reglas de entrada y salida al grupo de seguridad según su situación particular. Puede especificar una dirección IP, un rango de direcciones IP u otro grupo de seguridad de VPC.

> **Note**
>
> Al agregar una nueva regla de entrada, puede elegir My IP (Mi IP) en Source (Origen) para permitir el acceso a la instancia de base de datos desde la dirección IP detectada en su navegador.

Para obtener más información acerca de la configuración de grupos de seguridad, consulte Proporcionar acceso al clúster de base de datos en la VPC mediante la creación de un grupo de seguridad.

> **Note**
>
> La conexiones de cliente desde direcciones IP en el rango 169.254.0.0/16 no están permitidas. Este es el rango de direccionamiento IP privado automático (APIPA), que se utiliza para direccionamiento de enlace local.

- Accesibilidad pública: para conectarse a la instancia de base de datos desde fuera de la VPC, por ejemplo mediante una aplicación cliente, la instancia debe tener asignada una dirección IP pública.

  Para que la instancia sea accesible públicamente, modifíquela y elija Yes (Sí) en Public accessibility (Accesibilidad pública). Para obtener más información, consulte Cómo ocultar un clúster de base de datos en una VPC desde Internet..
- Puerto: el puerto especificado al crear la instancia de base de datos no puede usarse para enviar o recibir comunicaciones debido a las restricciones del firewall local. Para determinar si su red permite el uso del puerto especificado para comunicación de entrada y salida, consulte al administrador de red.
- Disponibilidad: en el caso de una instancia de base de datos recién creada, esta tendrá el estado `creating` hasta que esté lista para el uso. Cuando el estado cambie a `available`, podrá conectarse a la instancia de base de datos. Dependiendo del tamaño de la instancia de base de datos, es posible que la instancia tarde hasta 20 minutos en estar disponible.
- Gateway de Internet: para que una instancia de base de datos sea accesible públicamente, las subredes del grupo de subredes de base de datos deben tener una gateway de Internet.

Para configurar una gateway de Internet para una subred

1. Inicie sesión en Consola de administración de AWS y abra la consola de Amazon RDS en https://console.aws.amazon.com/rds/.
2. En el panel de navegación, elija Databases (Bases de datos) y, a continuación, elija el nombre de la instancia de base de datos.
3. En la pestaña Connectivity & security (Conectividad y seguridad), anote los valores del ID de la VPC en VPC y el ID de la subred en Subnets (Subredes).
4. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
5. En el panel de navegación, elija Internet Gateways (Gateways de Internet). Verifique que hay una gateway de Internet adjunta a la VPC. Si no la hay, elija Create Internet Gateway (Crear gateway de Internet) para crear una gateway de Internet. Seleccione la gateway de Internet y, después, elija Attach to VPC (Conectar a la VPC) y siga las instrucciones para adjuntarla a la VPC.
6. En el panel de navegación, elija Subnets (Subredes) y, a continuación, seleccione la suya.
7. En la pestaña Route Table (Tabla de ruteo), verifique que haya una ruta con `0.0.0.0/0` como destino y la gateway de Internet de la VPC como destino.

   Si está conectando con la instancia utilizando la dirección IPv6, verifique que existe una ruta para todo el tráfico IPv6 (`::/0`) que apunte a la gateway de Internet. De lo contrario, realice lo siguiente:

   a. Elija el ID de la tabla de ruteo (rtb-xxxxxxxx) para navegar a la tabla de ruteo.
   b. En la pestaña Routes (Rutas), elija Edit routes (Editar rutas). Elija Add route (Añadir ruta) y utilice `0.0.0.0/0` como destino y la gateway de Internet como objetivo.

      Para IPv6, elija Add route (Añadir ruta) y utilice `::/0` como destino y la gateway de Internet como objetivo.
   c. Elija Save routes (Guardar rutas).

   Además, si intenta conectarse al punto de conexión IPv6, asegúrese de que el rango de direcciones IPv6 del cliente esté autorizado para conectarse a la instancia de base de datos.

Para obtener más información, consulte Uso de una clúster de base de datos en una VPC.

Comprobar una conexión a una instancia de base de datos

Puede comprobar la conexión a una instancia de base de datos con las herramientas habituales de Microsoft Windows o Linux.

Desde un terminal de Linux o Unix, puede comprobar la conexión escribiendo lo siguiente. Sustituya *DB-instance-endpoint* por el punto de conexión y *port* por el puerto de la instancia de base de datos.

```
nc -zv DB-instance-endpoint port
```

Por ejemplo, a continuación se muestra un comando de ejemplo y el valor de retorno.

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Los usuarios de Windows pueden usar Telnet para comprobar la conexión a una instancia de base de datos. Las acciones de Telnet solo se admiten para la comprobación de la conexión. Si la conexión es correcta, la acción no devuelve ningún mensaje. Si la conexión no es correcta, recibe un mensaje de error como el siguiente.

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Si las acciones de Telnet indican que la conexión es correcta, el grupo de seguridad se ha configurado correctamente.

> **Note**
>
> Amazon RDS no acepta el tráfico del protocolo de mensaje de control de Internet (ICMP), ping incluido.

Solución de problemas de autenticación de conexión

En algunos casos, puede conectarse a su instancia de base de datos, pero recibe errores de autenticación. En estos casos, sería aconsejable restablecer la contraseña de usuario principal para la instancia de base de datos. Puede hacerlo modificando la instancia de RDS.

Problemas de seguridad de Amazon RDS

Para evitar problemas de seguridad, no utilice nunca la contraseña ni la dirección de correo electrónico del usuario raíz de Cuenta de AWS en una cuenta de usuario. El procedimiento recomendado consiste en utilizar el usuario raíz para crear usuarios y asignarlos a cuentas de usuario de base de datos. También puede utilizar el usuario raíz para crear, si fuera necesario, otras cuentas de usuario.

Para obtener información sobre cómo crear usuarios, consulte Creación de un usuario de IAM en la Cuenta de AWS. Para obtener información sobre cómo crear usuarios en AWS IAM Identity Center, consulte Manage identities in IAM Identity Center (Administrar identidades en el Centro de identidades de IAM).

Mensaje de error "No se pudieron recuperar los atributos de cuenta. Determinadas funciones de la consola pueden estar deterioradas".

Puede obtener este error por varias razones. Puede deberse a que la cuenta no tiene permisos o a que no se ha configurado correctamente. Si la cuenta es nueva, es posible que no haya esperado a que esté lista. Si se trata de una cuenta existente, podría carecer de permisos en sus políticas de acceso para realizar determinadas acciones, como crear una instancia de base de datos. Para solucionar este problema, el administrador debe proporcionar los roles necesarios a su cuenta. Para obtener más información, consulte la documentación de IAM.

Restablecimiento de la contraseña del propietario de la instancia de base de datos

Si se bloquea el clúster de base de datos, puede iniciar sesión como usuario maestro. A continuación, puede restablecer las credenciales para otros usuarios o roles administrativos. Si no puede iniciar sesión como usuario maestro, el propietario de la AWS cuenta puede restablecer la contraseña del usuario maestro. Para obtener información detallada sobre las cuentas

administrativas o roles que puede tener que restablecer, consulte Privilegios de la cuenta de usuario maestro.

La contraseña de la instancia de base de datos se puede cambiar por medio de la consola de Amazon RDS, el comando modify-db-instance de la AWS CLI o la operación de la API ModifyDBInstance. Para obtener información sobre la modificación de una instancia de base de datos en un clúster de base de datos, consulte Modificación de una instancia de base de datos en un clúster de base de datos.

Interrupción o reinicio de una instancia de base de datos de Amazon RDS

Cuando se reinicia una instancia de base de datos puede producirse una interrupción de la instancia de base de datos. También puede producirse cuando dicha instancia se pone en un estado que impide el acceso a ella y cuando se reinicia la base de datos. Puede producirse un reinicio al al reiniciar de forma manual su instancia de base de datos. Un reinicio también puede ocurrir cuando se cambia una configuración de la instancia de la base de datos que requiere un reinicio antes de que pueda surtir efecto.

El reinicio de una instancia de base de datos se produce cuando cambia una configuración que exige un reinicio o cuando efectúa un reinicio manualmente. El reinicio puede producirse inmediatamente si cambia una configuración y solicita que el cambio surta efecto de inmediato. O puede ocurrir durante el período de mantenimiento de la instancia de base de datos.

El reinicio de la instancia de base de datos se produce inmediatamente en una de las siguientes situaciones:

- Cambie el periodo de retención de copia de seguridad de una instancia de base de datos de cero a un valor distinto de cero o de un valor distinto de cero a cero. A continuación, configure Apply Immediately (Aplicar inmediatamente) en `true`.
- El usuario cambia la clase de la instancia de base de datos y Apply Immediately (Aplicar inmediatamente) se establece en `true`.

El reinicio de la instancia de base de datos se produce durante el período de mantenimiento en una de las siguientes situaciones:

- El usuario cambia el período de retención de copia de seguridad de una instancia de base de datos, de cero a un valor distinto de cero o viceversa, y Apply Immediately (Aplicar inmediatamente) se establece en `false`.
- El usuario cambia la clase de la instancia de base de datos y Apply Immediately (Aplicar inmediatamente) se establece en `false`.

Cuando se cambia un parámetro estático en un grupo de parámetros de base de datos, el cambio no surtirá efecto hasta que se reinicie la instancia de base de datos asociada al grupo. El cambio requiere un reinicio manual. La instancia de base de datos no se reinicia automáticamente durante el período de mantenimiento.

Los cambios de parámetros de base de datos de Amazon RDS no surten efecto

En algunos casos, es posible que cambie un parámetro en un grupo de parámetros de base de datos, pero no vea que los cambios surtan efecto. Si es así, es probable que necesite reiniciar la instancia de base de datos asociada con el grupo de parámetros de base de datos. Cuando se cambia un parámetro dinámico, el cambio surte efecto inmediatamente. Cuando se cambia un parámetro estático, el cambio no surtirá efecto hasta que reinicie la instancia de base de datos asociada al grupo de parámetros.

Puede reiniciar una instancia de base de datos a través de la consola de RDS. O bien, puede llamar explícitamente a la operación de la API `RebootDBInstance`. Puede reiniciar sin conmutación por error si la instancia de base de datos se encuentra en una implementación Multi-AZ. El requisito de reiniciar la instancia de base de datos asociada después de cambiar un parámetro estático ayuda a mitigar el riesgo de que una configuración errónea del parámetro afecte a una llamada a la API. Un ejemplo sería llamar a `ModifyDBInstance` para cambiar la clase de instancia de base de datos. Para obtener más información, consulte Modificación de los parámetros de un grupo de parámetros de base de datos en Amazon Aurora.

Problemas de memoria que se puede liberar en Amazon Aurora

La memoria que se puede liberar es el total de memoria de acceso aleatorio (RAM) de una instancia de base de datos que se puede poner a disposición del motor de base de datos. Es la suma de la memoria libre del sistema operativo (SO) y de la memoria intermedia y la memoria caché de

página disponibles. El motor de base de datos utiliza la mayor parte de la memoria del host, pero los procesos del sistema operativo también utilizan algo de RAM. La memoria actualmente asignada al motor de base de datos o que utilizan los procesos del sistema operativo no está incluida en la memoria que se puede liberar. Cuando el motor de base de datos se queda sin memoria, la instancia de base de datos puede utilizar el espacio temporal que normalmente se usa para el almacenamiento en búfer y en caché. Como se mencionó anteriormente, este espacio temporal se incluye en la memoria que se puede liberar.

La métrica `FreeableMemory` de Amazon CloudWatch se utiliza para supervisar la memoria que se puede liberar. Para obtener más información, consulte Supervisión de herramientas de Amazon Aurora.

Si su instancia de base de datos se queda constantemente sin memoria que se pueda liberar o utiliza espacio de intercambio, considere la psibilidad de escalar verticalmente a una clase de instancia de base de datos más grande. Para obtener más información, consulte Clases de instancia de base de datos de Amazon Aurora.

También puede cambiar la configuración de memoria. Por ejemplo, en Aurora MySQL , podría ajustar el tamaño del parámetro `innodb_buffer_pool_size`. Este parámetro está establecido de forma predeterminada al 75 % de la memoria física. Para obtener más consejos sobre la solución de problemas de MySQL, consulte How can I troubleshoot low freeable memory in an Amazon RDS para MySQL database? (¿Cómo puedo solucionar un problema de poca memoria que se puede liberar en una base de datos de Amazon RDS para MySQL?).

En el caso de Aurora Serverless v2, `FreeableMemory` representa la cantidad de memoria sin utilizar que está disponible cuando la instancia de base de datos de Aurora Serverless v2 se escala a su capacidad máxima. Puede tener la instancia reducida verticalmente a una capacidad relativamente baja, pero sigue notificando un valor alto para `FreeableMemory`, porque la instancia se puede escalar verticalmente. Esa memoria no está disponible en este momento, pero puede obtenerla si la necesita.

Para cada unidad de capacidad de Aurora (ACU) cuya capacidad actual esté por debajo de la capacidad máxima, `FreeableMemory` aumenta este valor aproximadamente 2 GiB. Por lo tanto, esta métrica no se aproxima a cero hasta que la instancia de base de datos se amplía al máximo posible.

Si esta métrica se aproxima a un valor de `0`, la instancia de base de datos se ha escalado al punto máximo posible. Se acerca al límite de la memoria disponible. Considere aumentar la configuración máxima de ACU para el clúster. Si esta métrica se aproxima a un valor de `0` en una instancia de

base de datos de lector, considere agregar instancias de base de datos de lector adicionales al clúster. De esta forma, puede distribuir la parte de solo lectura entre más instancias de base de datos, reduciendo el uso de memoria en cada instancia de base de datos de lector. Para obtener más información, consulte Métricas importantes de Amazon CloudWatch para Aurora Serverless v2.

Para Aurora Serverless v1, puede cambiar el intervalo de capacidad para utilizar más ACU. Para obtener más información, consulte Modificación de un clúster de bases de datos de Aurora Serverless v1.

Problemas de replicación de Amazon Aurora MySQL

Algunos problemas de replicación de MySQL también se aplican a Aurora MySQL. Puede diagnosticar y corregir estos problemas.

Temas

- Diagnóstico y resolución de retardos entre réplicas de lectura
- Diagnóstico y solución de un error de replicación de lectura de MySQL
- Error de replicación detenida
- La replicación de réplicas de lectura no puede inicializar la estructura de metadatos

Diagnóstico y resolución de retardos entre réplicas de lectura

Después de crear una réplica de lectura de MySQL y de que dicha réplica esté disponible, Amazon RDS replica en primer lugar los cambios realizados en la instancia de base de datos de origen desde el momento en que se inició la operación de creación de réplica de lectura. Durante esta fase, el retraso de replicación para la réplica de lectura es mayor que 0. También puede monitorizar este retardo en Amazon CloudWatch viendo la métrica `AuroraBinlogReplicaLag` de Amazon RDS.

La métrica `AuroraBinlogReplicaLag` indica el valor del campo `Seconds_Behind_Master` del comando `SHOW REPLICA STATUS` de MySQL. Para obtener más información, consulte HOW REPLICA STATUS Statement (Instrucción HOW REPLICA STATUS) en la documentación de MySQL.

Cuando la métrica `AuroraBinlogReplicaLag` llegue a 0, la réplica estará funcionando al mismo ritmo que la instancia de base de datos de origen. Si la métrica `AuroraBinlogReplicaLag` devuelve -1, la replicación podría no estar activa. Para solucionar problemas de error

de replicación, consulte Diagnóstico y solución de un error de replicación de lectura de MySQL. Un valor `AuroraBinlogReplicaLag` de -1 también puede significar que el valor `Seconds_Behind_Master` no se puede determinar o es `NULL`.

> **Note**
>
> Versiones anteriores de Aurora MySQL utilizaban `SHOW SLAVE STATUS` en lugar de `SHOW REPLICA STATUS`. Si usa una versión de Aurora MySQL 1 o 2, utilice `SHOW SLAVE STATUS`. Use `SHOW REPLICA STATUS` para Aurora MySQL, versión 3 o posterior

La métrica `AuroraBinlogReplicaLag` devuelve -1 durante una interrupción de la red o cuando se aplica un parche durante el período de mantenimiento. En este caso, espere a que se restaure la conectividad de la red o a que finalice el período de mantenimiento antes de volver a comprobar la métrica `AuroraBinlogReplicaLag`.

La tecnología de replicación de lectura de MySQL es asíncrona. Por lo tanto, cabe esperar aumentos ocasionales para la métrica `BinLogDiskUsage` en la instancia de base de datos de origen y para la métrica `AuroraBinlogReplicaLag` en la réplica de lectura. Por ejemplo, considere una situación en la que se pueden realizar en paralelo un gran volumen de operaciones de escritura en la instancia de base de datos de origen. Al mismo tiempo, las operaciones de escritura en la réplica de lectura se serializan utilizando un único subproceso de E/S. Tal situación puede provocar un retraso entre la instancia de origen y la réplica de lectura.

Para obtener más información acerca de las réplicas de lectura y MySQL, consulte Replication Implementation Details en la documentación de MySQL.

Puede hacer varias cosas para reducir el retraso entre las actualizaciones de una instancia de base de datos de origen y las actualizaciones posteriores de la réplica de lectura:

- Configure la clase de instancia de base de datos de la réplica de lectura para que tenga un tamaño de almacenamiento comparable al de la instancia de base de datos de origen.
- Asegúrese de que la configuración de parámetros de los grupos de parámetros de base de datos utilizados en la instancia de base de datos de origen y la réplica de lectura sean compatibles. Para obtener más información y un ejemplo, consulte el análisis del parámetro `max_allowed_packet` en la siguiente sección.
- Deshabilite la caché de consultas. Para tablas que se modifican a menudo, el uso de la caché de consultas puede aumentar el retardo de réplica porque la caché se bloquea y actualiza

con frecuencia. Si esto fuera así, podría ver menos retardo de réplica si deshabilita la caché de consultas. Puede deshabilitar la caché de consultas estableciendo `query_cache_type` `parameter` en 0 en el grupo de parámetros de base de datos para la instancia de base de datos. Para obtener más información sobre la caché de consultas, consulte Query Cache Configuration.

- Active el grupo de búferes en la réplica de lectura de InnoDB para MySQL . Por ejemplo, suponga que tiene un conjunto pequeño de tablas que se actualiza con frecuencia y está utilizando el esquema de tablas InnoDB o XtraDB. En este caso, puede volcar esas tablas en la réplica de lectura. Al hacer esto, el motor de base de datos examina las filas de esas tablas desde el disco y, a continuación, las almacena en la caché en el grupo del búfer. Este enfoque puede reducir el retraso de la réplica. A continuación se muestra un ejemplo.

  Para Linux, macOS o Unix:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

  Para Windows:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

Diagnóstico y solución de un error de replicación de lectura de MySQL

Amazon RDS supervisa el estado de replicación de las réplicas de lectura. RDS actualiza el campo Replication State (Estado de replicación) de la instancia de réplica de lectura a `Error` si la replicación se detiene por cualquier motivo. Puede revisar los detalles del error asociado que muestran los motores de MySQL visualizando el campo Replication Error (Error de replicación). También se generan eventos que indican el estado de la réplica de lectura, entre los que se incluyen RDS-EVENT-0045, RDS-EVENT-0046 y RDS-EVENT-0057. Para obtener más información acerca de los eventos y la suscripción a ellos, consulte Uso de notificaciones de eventos de Amazon RDS.

Si aparece un mensaje de error de MySQL, compruebe el error en la documentación de mensajes de error de MySQL.

Estas son algunas de las situaciones comunes que pueden causar errores de replicación:

- El valor para el parámetro `max_allowed_packet` para una réplica de lectura es inferior al parámetro `max_allowed_packet` para la instancia de base de datos de origen.

  El parámetro `max_allowed_packet` es un parámetro personalizado que puede establecer en un grupo de parámetros de base de datos. El parámetro `max_allowed_packet` se utiliza para especificar el tamaño máximo del lenguaje de manipulación de datos (DML) que se puede ejecutar en la base de datos. En algunos casos, el valor de `max_allowed_packet` de la instancia de base de datos de origen puede ser superior al valor de `max_allowed_packet` para la réplica de lectura. Si es así, el proceso de replicación puede generar el error y detener la replicación. El error más común es `packet bigger than 'max_allowed_packet' bytes`. Puede resolver el error haciendo que el origen y la réplica de lectura usen grupos de parámetros de base de datos con los mismos valores del parámetro `max_allowed_packet`.
- Escritura en tablas en una réplica de lectura. Si desea crear índices en una réplica de lectura, debe establecer el parámetro `read_only` en 0 para crear los índices. Si se escribe en las tablas de la réplica de lectura, puede interrumpirse la replicación.
- Uso de un motor de almacenamiento no transaccional como MyISAM. Las réplicas de lectura requieren un motor de almacenamiento transaccional. La reproducción solo se admite para los siguientes motores de almacenamiento: InnoDB for MySQL o MariaDB.

  Puede convertir una tabla MyISAM a InnoDB con el siguiente comando:

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```
- Uso de consultas no deterministas que no sean seguras, como `SYSDATE()`. Para obtener más información, consulte Determination of Safe and Unsafe Statements in Binary Logging en la documentación de MySQL.

Los siguientes pasos pueden ayudar a solucionar su error de replicación:

- Si se encuentra ante un error lógico y decide que es seguro hacer caso omiso, siga los pasos que se describen en Omisión del error de replicación actual. La instancia de base de datos de Aurora MySQL debe estar ejecutando una versión que incluya el procedimiento `mysql_rds_skip_repl_error`. Para obtener más información, consulte mysql_rds_skip_repl_error.

- Si se encuentra ante un problema de posición de registro binario (binlog), puede cambiar la posición de reproducción. Para hacerlo, utilice el comando `mysql.rds_next_master_log` para Aurora MySQL, versiones 1 y 2. Para hacerlo, utilice el comando `mysql.rds_next_source_log` para Aurora MySQL, versiones 3 y posteriores. La instancia de base de datos de Aurora MySQL debe estar ejecutando una versión que admita este comando para cambiar la posición de reproducción de la réplica. Para obtener información sobre la versión, consulte mysql_rds_next_master_log.
- Si se encuentra ante un problema de rendimiento temporal debido a una carga de DML elevada, puede establecer el parámetro `innodb_flush_log_at_trx_commit` en 2 en el grupo de parámetros de base de datos en la réplica de lectura. Hacer esto puede ayudar a la réplica de lectura a ponerse al corriente, si bien reduce temporalmente las propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID).
- Puede eliminar la réplica de lectura y crear una instancia con el mismo identificador de instancias de bases de datos. De este modo, el punto de conexión seguirá siendo el mismo que en la réplica de lectura antigua.

Si se corrige un error de replicación, Replication State cambia a replicating. Para obtener más información, consulte Solución de problemas de una réplica de lectura de MySQL o MariaDB.

Error de replicación detenida

Cuando llame al comando `mysql.rds_skip_repl_error`, es posible que reciba un mensaje de error en el que se indique que la reproducción tiene un error o está deshabilitada.

Este mensaje de error aparece porque la replicación se ha detenido y no se puede reiniciar.

Si tiene que omitir un número de errores elevado, el retardo de réplica puede aumentar por encima del periodo de retención predeterminado para los archivos de registro binarios. En este caso, puede producirse un error irrecuperable debido a que los archivos de registro binario se están limpiando antes de reproducirse de nuevo en la réplica. Esta limpieza hace que la replicación se detenga y ya no se puede llamar al comando `mysql.rds_skip_repl_error` para omitir los errores de replicación.

Puede mitigar este problema incrementando el número de horas que los archivos de registro binarios se retienen en el origen de la replicación. Después de incrementar el tiempo de retención de los archivos binlog, puede reiniciar la replicación y llamar al comando `mysql.rds_skip_repl_error` si es necesario.

Para establecer el tiempo de retención del binlog, utilice el procedimiento mysql_rds_set_configuration. Especifique un parámetro de configuración de “horas de retención del binlog” junto con el número de horas de retención de los archivos binlog en el clúster de base de datos, hasta un máximo de 2160 (90 días). El valor predeterminado para Aurora MySQL es 24 (1 día). El ejemplo siguiente define el periodo de retención de los archivos binlog en 48 horas.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

La replicación de réplicas de lectura no puede inicializar la estructura de metadatos

Cuando intentó iniciar la replicación, se recibió el siguiente mensaje de error:

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

Este error se produce cuando hay un problema con la estructura de metadatos de la réplica. Para corregir la estructura de los metadatos, debe crear una nueva réplica.

Para evitar que esto ocurra en el futuro, lleve a cabo alguna de las siguientes acciones:

- Si es posible, desactive los subprocesos múltiples en las réplicas. A partir de MySQL 8.0.27, los subprocesos múltiples están habilitados de forma predeterminada.
- Si necesita utilizar subprocesos múltiples en las réplicas, le recomendamos que utilice la replicación basada en GTID. Para obtener más información, consulte Uso de la replicación basada en GTID.

Referencia de la API de Amazon RDS

Además de Consola de administración de AWS y la AWS Command Line Interface (AWS CLI), Amazon RDS también proporciona una API. Puede utilizar la API para automatizar las tareas de administración de instancias de base de datos y otros objetos en Amazon RDS.

- Para ver una lista de acciones de la API ordenada alfabéticamente, consulte el tema relacionado con las acciones.
- Para ver una lista de tipos de datos ordenada alfabéticamente, consulte el tema relacionado con los Tipos de datos.
- Para ver una lista de parámetros de consulta comunes, consulte el tema relacionado con los Parámetros comunes.
- Para ver las descripciones de los códigos de error, consulte el tema relacionado con los Errores comunes.

Para obtener más información acerca de AWS CLI, consulte la referencia de AWS Command Line Interface para Amazon RDS.

Temas

- Uso de la API de consulta
- Solución de problemas de aplicaciones en Aurora

Uso de la API de consulta

En las siguientes secciones se explican brevemente los parámetros y la autenticación de solicitudes que se utilizan con la API de consulta.

Para obtener información general acerca del funcionamiento de la API de consulta, consulte Solicitudes de consulta en la Amazon EC2 API Reference.

Parámetros de consulta

Las solicitudes basadas en consultas HTTP son solicitudes HTTP que utilizan el verbo HTTP GET o POST y un parámetro de consulta denominado `Action`.

Cada solicitud de consulta debe incluir algunos parámetros comunes para realizar la autenticación y la selección de una acción.

Algunas operaciones toman listas de parámetros. Estas listas se especifican utilizando la notación `param.`*n*. Los valores de *n* son números enteros a partir de 1.

Para obtener más información acerca de las regiones y los puntos de conexión de Amazon RDS, vaya a Amazon Relational Database Service (RDS) en la sección Regiones y puntos de conexiones de la Referencia general de Amazon Web Services.

Autenticación de solicitudes de consulta

Solo se pueden enviar solicitudes de consulta a través de HTTPS, y cada una de ellas debe incluir una firma. Debe utilizar Signature Version 4 o Signature Version 2 de AWS. Para obtener más información, consulte Proceso de firma de Signature Version 4 y Proceso de firma de Signature Version 2.

Solución de problemas de aplicaciones en Aurora

Amazon RDS proporciona errores específicos y descriptivos para ayudarle a solucionar problemas durante la interacción con la API de Amazon RDS.

Temas

- Recuperación de errores
- Consejos para la solución de problemas

Para obtener más información sobre la solución de problemas para instancias de base de datos de Amazon RDS, consulte Solución de problemas de Amazon Aurora.

Recuperación de errores

Normalmente, conviene que una aplicación compruebe si una solicitud generó un error antes de emplear tiempo en procesar los resultados. La forma más fácil de averiguar si se ha producido un error, consiste en buscar un nodo `Error` en la respuesta de la API de Amazon RDS.

La sintaxis XPath proprciona una forma simple de buscar la presencia de un nodo `Error`. También proporciona una forma relativamente fácil de recuperar el código y el mensaje de error. La partición de código siguiente utiliza Perl y el módulo XML::XPath para determinar si se ha producido un error

durante una solicitud. Si es así, el código imprime el primer mensaje de error y su código en la respuesta.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
    {print "There was an error processing your request:\n", " Error code: ",
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Consejos para la solución de problemas

Recomendamos los siguientes procesos para diagnosticar y solucionar problemas con la API de Amazon RDS:

- Verifique si Amazon RDS funciona normalmente en la región de AWS de destino consultando http://status.aws.amazon.com.
- Comprobar la estructura de la solicitud.

  Cada operación de Amazon RDS tiene una página de referencia en la Referencia de la API de Amazon RDS. Compruebe que está utilizando los parámetros correctamente. Para obtener ideas sobre lo que podría estar mal, mire las solicitudes de ejemplo o los escenarios de usuario para ver si esos ejemplos hacen operaciones similares.
- Comprobar AWS re:Post

  Existe una comunidad de desarrolladores de Amazon RDS donde puede buscar soluciones a los problemas que otras personas han experimentado al utilizar este servicio. Para ver los temas, vaya a AWS re:Post.

Historial de revisión

Versión actual de la API: 2014-10-31

En la siguiente tabla se describen cambios importantes en la Guía del usuario de Amazon Aurora. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS. Para obtener más información sobre Amazon Relational Database Service (Amazon RDS), consulte la Guía del usuario de Amazon Relational Database Service.

> **Note**
>
> Antes del 31 de agosto de 2018, Amazon Aurora se documentaba en la Guía del usuario de Amazon Relational Database Service. Para ver el historial de revisión anterior de Aurora, consulte Historial de revisión en la Guía del usuario de Amazon Relational Database Service.

Puede filtrar nuevas características de Amazon Aurora en la página Novedades de Database. En Products (Productos), elija Amazon Aurora. Luego, busque con palabras clave como **`global database`** o **`Serverless`**.

Cambio	Descripción	Fecha
Las integraciones sin ETL de Aurora PostgreSQL con Amazon SageMaker admiten regiones adicionales	Las integraciones sin ETL de Aurora PostgreSQL con Amazon SageMaker ahora están disponibles en regiones adicionales. Para obtener más información, consulte Regiones y motores de base de datos de Aurora admitidos para integraciones sin ETL.	15 de octubre de 2025
La base de datos sin límite de Aurora PostgreSQL está disponible en una región de AWS adicional	La base de datos sin límite de Aurora PostgreSQL ya está disponible en las AWS GovCloud (US) Regions. Para obtener más informaci	16 de septiembre de 2025

	ón, consulte Requisitos y consideraciones de base de datos ilimitada de Aurora PostgreSQL.	
La API de datos de Amazon RDS admite IPv6	La API de datos de Amazon RDS ahora admite la conectividad IPv6, lo que le permite conectarse a las bases de datos de Aurora mediante direcciones IPv6. Para obtener más información, consulte Uso de la API de datos.	30 de agosto de 2025
Amazon Aurora está disponible en la región de Asia-Pacífico (Nueva Zelanda)	Amazon Aurora está ahora disponible en la región de Asia-Pacífico (Nueva Zelanda). Para obtener más información, consulta Regiones y zonas de disponibilidad.	28 de agosto de 2025
La API de datos de Amazon RDS disponible en una región de AWS adicional	La API de datos de Amazon RDS ya está disponible en la región de Europa (España). Para obtener más información, consulte Regiones y motores de base de datos de Aurora admitidos para la API de datos de RDS.	1 de agosto de 2025

Las integraciones sin ETL con Amazon SageMaker Lakehouse ya están disponibles de forma general	Con las integraciones sin ETL, los datos transaccionales pueden estar disponibles en Amazon SageMaker en solo unos segundos después de escribirlos en una instancia de base de datos de Aurora MySQL. Para obtener más información, consulte Uso de integraciones sin ETL de Aurora.	30 de junio de 2025
Amazon Aurora está disponible en la región de Asia-Pacífico (Taipéi)	Amazon Aurora está ahora disponible en la región de Asia-Pacífico (Taipéi). Para obtener más información, consulta Regiones y zonas de disponibilidad.	5 de junio de 2025

Información sobre el fin de la vida útil de la Información de rendimiento	AWS ha anunciado la fecha de fin de la vida útil de Información de rendimiento: el 30 de noviembre de 2025. Después de esta fecha, Amazon RDS dejará de admitir la experiencia de la consola de Información de rendimiento, los periodos de retención flexibles y los precios asociados. Le recomendamos que actualice los clústeres de bases de datos a información de base de datos antes de esta fecha. Para obtener más información, consulte Supervisión de la carga de base de datos con Información de rendimiento en Amazon Aurora.	30 de mayo de 2025
La base de datos ilimitada de Aurora PostgreSQL es compatible con información sobre base de datos de CloudWatch	Ahora puede utilizar información sobre las bases de datos de CloudWatch para supervisar la base de datos ilimitada de Amazon Aurora PostgreSQL a escala. Para obtener más información, consulte Supervisión de la base de datos ilimitada de Amazon Aurora PostgreSQL con información de la base de datos de CloudWatch.	22 de mayo de 2025

Visualización de las fechas de soporte para las versiones principales de motores de código abierto	Ahora puede ver información sobre las fechas de inicio y finalización de las versiones principales de motores de código abierto en el soporte estándar de RDS y en el soporte extendido de RDS mediante la AWS CLI y la API de RDS. Para obtener más información, consulte Visualización de las fechas de soporte de las versiones del motor en soporte extendido de Amazon RDS.	20 de mayo de 2025
Las integraciones sin ETL de Aurora PostgreSQL admiten regiones adicionales	Las integraciones sin ETL de Aurora PostgreSQL ahora están disponibles en regiones adicionales. Para obtener más información, consulte Regiones y motores de base de datos Aurora admitidos para integraciones sin ETL con Amazon Redshift.	17 de febrero de 2025
Amazon Aurora está disponible en la región de México (centro)	Amazon Aurora está ahora disponible en la región de México (centro). Para obtener más información, consulta Regiones y zonas de disponibilidad.	14 de enero de 2025

Amazon Aurora está disponible en la región de Asia-Pacífico (Tailandia)	Amazon Aurora está ahora disponible en la región de Asia-Pacífico (Tailandia). Para obtener más información, consulta Regiones y zonas de disponibilidad.	7 de enero de 2025

Información sobre el fin de la vida útil de Aurora Serverless v1	AWS ha anunciado la fecha de fin de la vida útil de Aurora Serverless v1 que será el 31 de marzo de 2025. Recomendamos encarecidamente actualizar todos los clústeres de bases de datos de Aurora Serverless v1 a Aurora Serverless v2 antes de dicha fecha. Esta actualización puede implicar un cambio en el número de versión principal del motor de base de datos. Por lo tanto, es importante planificar, probar e implementar esta transición antes de la fecha de fin de la vida útil. A partir del 8 de enero de 2025, los clientes ya no podrán crear nuevos clústeres o instancias de Aurora Serverless v1 con la Consola de administración de AWS o la CLI. Para obtener más información sobre el procedimiento de actualización de Aurora Serverless v1 a Aurora Serverless v2, consulte Actualización de un clúster de Aurora Serverless v1 a Aurora Serverless v2.	6 de diciembre de 2024

Aurora es compatible con Información sobre las bases de datos de CloudWatch	Ahora puede utilizar Información sobre las bases de datos de CloudWatch para supervisar sus clústeres de base de datos de Amazon Aurora a escala. Para obtener más información, consulte Supervisión de las bases de datos de Amazon Aurora con Información sobre las bases de datos de CloudWatch.	1 de diciembre de 2024
Amazon Aurora es compatible con las clases de instancia db.r8g	Ahora puede usar clases de instancia db.r8g para crear clústeres de base de datos de Aurora. Para obtener más información, consulte Clases de instancia de base de datos de Aurora.	21 de noviembre de 2024
Aurora Serverless v2 es compatible con la pausa y la reanudación automáticas	Las instancias de base de datos de Aurora Serverless v2 ahora se pueden escalar hasta cero unidades de capacidad (ACU) de Aurora tras un período de inactividad. Para habilitar esta característica, establezca el valor de capacidad como 0 ACU. Para obtener más información, consulte Pausa y reanudación automáticas de Aurora Serverless v2.	20 de noviembre de 2024

Amazon Aurora admite la migración automática de bases de datos de EC	Puede utilizar la consola de RDS para migrar una base de datos de EC2 a Aurora. Aurora utiliza AWS Database Migration Service (AWS DMS) para migrar una base de datos de EC2 de origen. AWS DMS permite migrar bases de datos relacionales a una Nube de AWS. Para obtener más información, consulte Migración automática de bases de datos de EC2 a Amazon Aurora con AWS Database Migration Service.	20 de noviembre de 2024
AWS Disponibilidad general del contenedor de NodeJS avanzado de	El contenedor de NodeJS avanzado de Amazon Web Services (AWS) complementa y amplía la funcionalidad del controlador de NodeJS existente. Para obtener más información, consulte Conexión a clústeres de bases de datos Aurora con los controladores de AWS.	19 de noviembre de 2024
Amazon Aurora es compatible con las clases de instancia db.r7i	Ahora puede usar clases de instancia db.r7i para crear clústeres de base de datos de Aurora. Para obtener más información, consulte Clases de instancia de base de datos de Aurora.	18 de noviembre de 2024

La API de datos de RDS para Aurora MySQL es compatible con todas las vistas de Información de rendimiento	Ahora puede utilizar las vistas de Hosts principales o Aplicaciones principales de Información de rendimiento para supervisar las operaciones de SQL realizadas con la API de datos de RDS para Aurora MySQL. Para obtener información sobre Información de rendimiento, consulte Supervisión de las consultas de la API de datos de RDS con Información de rendimiento.	31 de octubre de 2024

Base de datos ilimitada de Aurora PostgreSQL	Base de datos ilimitada de Amazon Aurora PostgreSQL es una nueva capacidad de escalado horizontal (particio namiento) automatizada de Amazon Aurora. Base de datos ilimitada de Aurora PostgreSQL le permite escalar más allá de los límites actuales de Aurora en cuanto a rendimiento de escritura y almacenamiento al distribui r la carga de trabajo de una base de datos entre varias instancias de escritura de Aurora y, al mismo tiempo, mantener la capacidad de utilizarla como una única base de datos. Para obtener más información, consulte Amazon Aurora PostgreSQL Limitless Database.	31 de octubre de 2024
Final del soporte estándar de la versión 2 de Amazon Aurora MySQL	La versión 2 de Aurora MySQL ya no recibe soporte estándar desde el 31 de octubre de 2024. Para obtener más información, consulte Preparación para el final del soporte estándar de la versión 2 de Amazon Aurora MySQL-Compatible Edition.	31 de octubre de 2024

Amazon Aurora admite actualizaciones del sistema operativo en el clúster	Ahora puede aplicar las actualizaciones del sistema operativo Aurora en el clúster de base de datos. Las actualizaciones sucesivas aplican automáticamente las actualizaciones a unas cuantas instancias de bases de datos de lector a la vez, lo que conserva la disponibilidad de lectura. Para obtener más información, consulte Actualizaciones del sistema operativo para clústeres de base de datos de Aurora.	30 de octubre de 2024
Los clústeres globales de Aurora admiten el etiquetado	Ahora puede agregar etiquetas a los clústeres globales de Aurora. Un clúster global de Aurora es el recurso principal que contiene los clústeres de base de datos principal y secundario de una base de datos global de Aurora. Para obtener información general acerca del etiquetado, consulte Etiquetado de los recursos de Amazon Aurora y Amazon RDS. Para obtener información sobre el etiquetado con la base de datos global de Aurora, consulte Tagging Amazon Aurora Global Database resources.	24 de octubre de 2024

Puntos de conexión del escritor de la base de datos global de Aurora	Cada base de datos global de Aurora incluye un punto de conexión del escritor que Aurora actualiza automáticamente para enrutar las solicitudes a la instancia del escritor actual del clúster de base de datos principal después de una operación de transición y conmutación por error de la base de datos global de Aurora. Para obtener más información sobre el uso del punto de conexión del escritor junto con la transición y la conmutación por error de la base de datos global de Aurora, consulte Uso de la transición o la conmutación por error en una base de datos global de Amazon Aurora.	22 de octubre de 2024
Las integraciones sin ETL de Aurora PostgreSQL con Amazon Redshift ya están disponibles con carácter general	Con las integraciones sin ETL, los datos transaccionales pueden estar disponibles en Amazon Redshift en solo unos segundos después de escribirlos en un clúster de base de datos de origen de Aurora PostgreSQL. La característica ya está disponible con carácter general. Para obtener más información, consulte Trabajar con integraciones Aurora sin ETL con Amazon Redshift.	15 de octubre de 2024

Aurora Serverless v2 admite 256 ACU	Ahora puede crear clústeres de base de datos de Aurora Serverless v2 con una capacidad máxima de 256 unidades de capacidad de Aurora (ACU). Para obtener más información, consulte Capacidad de Aurora Serverless v2.	3 de octubre de 2024
Amazon Aurora admite Console-to-Code	Ahora puede utilizar Console-to-Code para generar código a partir de acciones que se realizan en la consola de RDS. El código generado puede ayudarle a escribir código para automatizar el uso de otros servicios de AWS. Para obtener más información, consulte Use Console-to-Code to generate code for your Aurora console actions.	3 de octubre de 2024
La API de datos de RDS admite Información de rendimiento	Ahora puede utilizar Información de rendimiento para supervisar las operaciones de SQL realizadas con la API de datos de RDS. Para obtener información sobre Información de rendimiento, consulte Monitoring RDS Data API queries with Performance Insights.	26 de septiembre de 2024

La API de datos de RDS para Aurora Serverless v2 y la aprovisionada están disponibles para Aurora MySQL	La API de datos de RDS ya está disponible para los clústeres de Aurora MySQL que utilizan instancias de Aurora Serverless v2 o instancias aprovisionadas. Para obtener información sobre el uso de la API de datos de RDS, consulte Uso de la API de datos de RDS. Para obtener información sobre la compatibilidad de la versión de Aurora MySQL y la región de AWS para la API de datos de RDS, consulte API de datos con Aurora MySQL sin servidor v2 y aprovisionada.	26 de septiembre de 2024
Amazon Aurora está disponible en la región de Asia-Pacífico (Malasia	Amazon Aurora está disponible ahora en la región de Asia-Pacífico (Malasia). Para obtener más información, consulta Regiones y zonas de disponibilidad.	22 de agosto de 2024

Actualización de una política existente	Amazon RDS ha eliminado el permiso `sns:Publish` de `AmazonRDSBetaServiceRolePolicy` del rol vinculado a un servicio `AWSServiceRoleForRDSBeta` . Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	7 de agosto de 2024
Actualización de una política existente	Amazon RDS ha eliminado el permiso `sns:Publish` de `AmazonRDSPreviewServiceRolePolicy` del rol vinculado a un servicio `AWSServiceRoleForRDSPreview` . Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	7 de agosto de 2024
Aurora Serverless v2 es compatible con la pausa automática (escala a cero)	Las instancias de base de datos de Aurora Serverless v2 ahora se pueden escalar hasta cero unidades de capacidad (ACU) de Aurora tras un período de inactividad. Para habilitar esta característica, establezca el valor de capacidad mínima como 0. Para obtener más información, consulte Uso de Aurora Serverless v2.	25 de julio de 2024

AWS Controlador ODBC de para MySQL disponible de forma general	El controlador ODBC de Amazon Web Services (AWS) para MySQL es un controlador de cliente diseñado para la alta disponibilidad de Aurora MySQL. Para obtener más información, consulte Connecting to Aurora MySQL with the Amazon Web Services (AWS) ODBC Driver for MySQL.	18 de julio de 2024
La API de datos de RDS para Aurora Serverless v2 está disponible en más regiones	La API de datos de RDS ya está disponible para Aurora PostgreSQL en varias regiones de AWS adicionales. Para obtener información sobre la compatibilidad regional con la API de datos de RDS, consulte API de datos con Aurora PostgreSQL sin servidor v2 y aprovisionada.	9 de julio de 2024
AWS El controlador de Python de está disponible de forma general	El controlador de Python de Amazon Web Services (AWS) se ha diseñado como un contenedor de Python avanzado. Este contenedor complementa y amplía la funcionalidad del controlador de Psycopg de código abierto. Para obtener más información, consulte Connecting to Aurora DB clústers with the AWS drivers.	23 de mayo de 2024

Integraciones sin ETL disponibles en las regiones de China	Las integraciones sin ETL ahora están disponibles en las Regiones de AWS China (Pekín) y China (Ningxia). Para obtener más información, consulte Integraciones sin ETL con Amazon Redshift.	21 de mayo de 2024
RDS Proxy está disponible en más regiones	RDS Proxy ya está disponible en las regiones de Asia-Pacífico (Hyderabad), Asia Pacífico (Melbourne), Medio Oriente (EAU), Israel (Tel Aviv), Israel (Tel Aviv), Oeste de Canadá (Calgary) y Europa (Zúrich). Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	21 de mayo de 2024
Amazon RDS admite un acceso detallado para Información sobre rendimiento	Ahora es posible permitir o denegar el acceso a dimensiones individuales en Información sobre rendimiento. Este acceso detallado se puede utilizar para acciones `GetResourceMetrics`, `DescribeDimensionKeys` y `GetDimensionKeyDetails`. Para obtener más información, consulte https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.access-control.dimensionAccess-policy.html.	21 de mayo de 2024

Soporte extendido de Amazon RDS	Al crear o restaurar una base de datos Aurora MySQL versión 2 o 3, o Aurora PostgreSQL versión 11, ahora se inscribe automáticamente esa base de datos en el Soporte extendido de Amazon RDS para que las aplicaciones existentes sigan funcionando como siempre. Puede cancelar el Soporte extendido de RDS para evitar que se le cobre después de la fecha de finalización del soporte estándar de Aurora para su motor de base de datos. Para obtener más información, consulte Uso del soporte extendido de Amazon RDS.	21 de marzo de 2024
Filtrado de datos para integraciones sin ETL	Amazon RDS admite el filtrado de datos de la base de datos y la tabla para integraciones sin ETL con Amazon Redshift. Para obtener más información, consulte Data filtering for Aurora zero-ETL integrations with Amazon Redshift.	20 de marzo de 2024

Integraciones de Aurora MySQL con Amazon Bedrock	Ahora puede integrar las bases de datos Amazon Aurora MySQL con Amazon Bedrock para impulsar las aplicaciones de IA generativa. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora MySQL.	8 de marzo de 2024
Nueva política administrada de AWS	Amazon RDS agregó una nueva política administrada denominada `AmazonRDSCustomInstanceProfileRolePolicy` para permitir a RDS Custom realizar acciones de automatización y tareas de administración de bases de datos a través de un perfil de instancia de EC2. Para obtener más información, consulte Actualizaciones de Amazon RDS a políticas administradas por AWS.	27 de febrero de 2024
Compatibilidad con Amazon RDS para AWS Secrets Manager en la región Israel (Tel Aviv)	Amazon RDS es compatible con Secrets Manager en la región Israel (Tel Aviv). Para obtener más información, consulte Administración de contraseñas con Amazon RDS y AWS Secrets Manager.	21 de febrero de 2024

Soporte extendido de Amazon RDS	Amazon RDS habilita ahora automáticamente el Soporte extendido de Amazon RDS cuando las versiones principales del motor Aurora MySQL y Aurora PostgreSQL en sus clústeres de base de datos y clústeres globales alcanzan la fecha de fin de soporte estándar de Aurora. Para obtener más información, consulte Uso del soporte extendido de Amazon RDS.	15 de febrero de 2024
Aurora PostgreSQL 16.1 es compatible con Babelfish para Aurora PostgreSQL 4.0.0	Aurora PostgreSQL 16.1 es compatible con Babelfish 4.0.0. Para ver una lista de las nuevas características, consulte 16.1. Para ver una lista de las características admitidas en cada versión de Babelfish, consulte Funcionalidades compatibles con Babelfish por versión. Para obtener más información, consulte Uso de Babelfish para Aurora PostgreSQL.	31 de enero de 2024
Actualización al certificado de CA predeterminado	El certificado de CA predeterminado está establecido en `rds-ca-rsa2048-g1` . A fin de obtener más información, consulte Uso de SSL/TLS para cifrar una conexión a un clúster de base de datos.	26 de enero de 2024

RDS Proxy está disponible en la región de Europa (España	RDS Proxy ya está disponible en la región de Europa (España). Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	8 de enero de 2024
API de datos de RDS con Aurora PostgreSQL sin servidor v2 y aprovisionada	Ahora puede usar la API de datos de RDS con Aurora PostgreSQL sin servidor v2 y los clústeres de bases de datos aprovisionados. Con la API de datos de RDS, puede acceder a sus clústeres de Aurora a través de un punto de conexión HTTP seguro y ejecutar instrucciones SQL sin utilizar controladores de bases de datos ni administrar conexiones. Para obtener más información, consulte Uso de la API de datos de RDS.	21 de diciembre de 2023
Integraciones de Amazon PostgreSQL con Amazon Bedrock	Ahora puede integrar las bases de datos PostgreSQL de Amazon Aurora con Amazon Bedrock para impulsar las aplicaciones de IA generativa. Para obtener más información, consulte Uso de machine learning de Amazon Aurora con Aurora PostgreSQL.	21 de diciembre de 2023

Amazon Aurora está disponible en la región de Oeste de Canadá (Calgary	Amazon Aurora ya está disponible en la región de Oeste de Canadá (Calgary). Para obtener más información, consulta Regiones y zonas de disponibilidad.	20 de diciembre de 2023
Amazon RDS es compatible con la visualización y respuesta a las recomendaciones	Las recomendaciones de Amazon Aurora ahora incluyen recomendaciones proactivas basadas en umbrales y reactivas basadas en el machine learning. Para obtener más información, consulte Visualización y respuesta a las recomendaciones de Amazon Aurora.	19 de diciembre de 2023
Integraciones sin ETL de Amazon PostgreSQL con Amazon Redshift (vista previa)	Puede crear integraciones sin ETL con Amazon Redshift mediante un clúster de base de datos de origen de Aurora PostgreSQL. Para la versión de vista previa, debe crear todas las integraciones en el Entorno de vista previa de bases de datos de Amazon RDS, en la Región de AWS Este de EE. UU. (Ohio) (us-east-2). Para obtener más información, consulte Trabajar con integraciones Aurora sin ETL con Amazon Redshift.	28 de noviembre de 2023

Amazon Aurora PostgreSQL admite el reenvío de escritura de base de datos global	Ahora puede activar el reenvío de escritura en clústeres secundarios en una base de datos global basada en Aurora PostgreSQL. Para obtener más información, consulte Uso del reenvío de escritura en una base de datos global de Aurora PostgreSQL.	9 de noviembre de 2023
Compatibilidad de Aurora PostgreSQL con lecturas optimizadas	Puede conseguir un procesamiento de consultas más rápido para Aurora PostgreSQL con las lecturas optimizadas de Aurora. Para obtener más informaci ón, consulte Mejora del rendimiento de las consultas para Aurora PostgreSQL las lecturas optimizadas de Aurora.	8 de noviembre de 2023

Amazon RDS publica métricas de Performance Insights en Amazon CloudWatch	Performance Insights le permite exportar los paneles de métricas preconfiguradas o personalizadas en Amazon CloudWatch. Los paneles de métricas exportadas se pueden ver en la consola de CloudWatch. También puede exportar un widget seleccionado de métricas de Performance Insights y ver los datos de las métricas en la consola de CloudWatch. Para obtener más información, consulte Exporting Performance Insights metrics to CloudWatch.	8 de noviembre de 2023
Disponibilidad general de las integraciones Aurora MySQL sin ETL con Amazon Redshift	Las integraciones sin ETL con Amazon Redshift ya están disponibles de forma general para Aurora MySQL. Para obtener más información, consulte Trabajar con integraciones Aurora sin ETL en Amazon Redshift..	7 de noviembre de 2023
Soporte de Aurora PostgreSQL para implementaciones azul/verde de RDS	Ahora puede crear una implementación azul/verde a partir de un clúster de la base de datos de Aurora PostgreSQL. Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon RDS para actualizar las bases de datos.	26 de octubre de 2023

Aurora MySQL admite el cifrado en el servidor con AWS KMS keys (SSE-KMS)	En Aurora MySQL versión 3.05 y posteriores, puede usar SSE-KMS, incluidas Claves administradas por AWS y claves administradas por el cliente, para el cifrado en el servidor de los datos que cargue o guarde en Amazon S3. Para obtener más información, consulte Carga de datos en un clúster de base de datos de Amazon Aurora MySQL desde archivos de texto en un bucket de Amazon S3 y Almacenamiento de datos desde un clúster de base de datos MySQL de Amazon Aurora desde archivos de texto en un bucket de Amazon S3.	25 de octubre de 2023
Las optimizaciones de Aurora MySQL reducen el tiempo de reinicio de la base de datos	En la versión 3.05 de Aurora MySQL y versiones posteriores, hemos introducido optimizaciones que reducen el tiempo de reinicio de la base de datos. Estas optimizaciones reducen hasta en un 65 % el tiempo de inactividad y las interrupciones de las cargas de trabajo de la base de datos tras un reinicio. Para obtener más información, consulte Optimizaciones para reducir el tiempo de reinicio de la base de datos.	25 de octubre de 2023

Actualización a las políticas administradas de AWS	Las políticas administradas `AmazonRDSPerformanceInsightsReadOnly` y `AmazonRDSPerformanceInsightsFullAccess` incluyen ahora `Sid` (ID de instrucción) como identificador en las instrucciones de la política. Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	23 de octubre de 2023
Amazon RDS publica métricas de contador de Información sobre rendimiento en Amazon CloudWatch	La función matemática de métricas DB_PERF_INSIGHTS de la consola de CloudWatch sirve para consultar Amazon RDS para conocer las métricas de los contadores de Información sobre rendimiento. Para obtener más información, consulte Creación de alarmas de CloudWatch para supervisar Amazon Aurora.	20 de septiembre de 2023

Amazon Aurora admite la recuperación en un momento dado (PITR) con AWS Backup	Ahora puede administrar las copias de seguridad automatizadas (continuas) de Aurora en AWS Backup y realizar una restauración a partir de ellas a momentos especificados. Para obtener más información, consulte Restauración de un clúster de base de datos a un momento especificado con AWS Backup.	7 de septiembre de 2023
Soporte extendido de Amazon RDS	Amazon Aurora anuncia que próximamente podrá seguir ejecutando las versiones principales de los motores de Aurora MySQL y Aurora PostgreSQL en sus instancias de base de datos una vez pasada la fecha de finalización del soporte estándar de Aurora. Para obtener más información, consulte Uso del soporte extendido de Amazon RDS.	1 de septiembre de 2023
Amazon Aurora MySQL amplía el soporte para Percona XtraBackup	Ahora puede realizar migraciones físicas de bases de datos MySQL 8.0 a clústeres de bases de datos de la versión 3 de Aurora MySQL. Para obtener más información, consulte Migración física de MySQL con Percona XtraBackup y Amazon S3.	24 de agosto de 2023

La base datos global de Aurora admite la conmutación por error de base de datos global	La base datos global de Aurora ahora admite la conmutación por error global administrada, lo que le permite recuperarse más fácilmente de un verdadero desastre regional o de una interrupción total del nivel de servicio. Para obtener más información sobre esta característica, consulte Ejecución de la conmutación por error administrada para bases de datos globales de Aurora. La característica que antes se denominaba “conmutación por error planificada administrada” ahora se denomina “transición”. Para obtener información sobre los cambios, consulte Ejecución de transiciones para bases de datos globales de Amazon Aurora.	21 de agosto de 2023
Actualización de los permisos de políticas administradas por AWS	La política administrada `AmazonRDSFullAccess` tiene nuevos permisos que le permiten generar, ver y eliminar el informe de análisis de rendimiento durante un período de tiempo. Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	17 de agosto de 2023

Actualización de los permisos de políticas administradas por AWS	La adición de nuevos permisos a la política administrada `AmazonRDSPerformanceInsightsReadOnly` y la adición de una nueva política administrada `AmazonRDSPerformanceInsightsFullAccess` le permite generar un informe de análisis de carga de base de datos para un período de tiempo. Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	16 de agosto de 2023
Amazon RDS admite el análisis del período de tiempo de carga de la base de datos con la Información de rendimiento	La Información de rendimiento le permite crear informes de análisis de rendimiento para un período de tiempo específico. El informe proporciona información identificada y las recomendaciones para resolver los problemas de rendimiento. Para obtener más información, consulte Análisis de la carga de la base de datos durante un período de tiempo.	16 de agosto de 2023

Amazon Aurora admite la retención de copias de seguridad automatizadas para clústeres de bases de datos	Ahora puede retener copias de seguridad automatizadas de los clústeres de Aurora eliminados y restaurarlos a un momento dado. Para obtener más información, consulte Retener copias de seguridad automatizadas.	4 de agosto de 2023
Amazon Aurora ya está disponible en la región de Israel (Tel Aviv	Amazon Aurora ya está disponible en la región de Israel (Tel Aviv). Para obtener más información, consulta Regiones y zonas de disponibilidad.	1 de agosto de 2023
Amazon Aurora MySQL admite el reenvío de escritura local (en el clúster	Ahora puede reenviar operaciones de escritura desde una instancia de base de datos del lector a una instancia de base de datos del escritor dentro de un clúster de base de datos de Aurora MySQL. Para obtener más información, consulte Uso del reenvío de escritura local en un clúster de base de datos de Amazon Aurora MySQL.	31 de julio de 2023
Amazon Aurora admite Aurora Serverless v2 en una Región de AWS adicional	Ahora puede crear clústeres de bases de datos de Aurora Serverless v2 en la Región de AWS Asia-Pacífico (Melbourne). Para obtener más información acerca de Aurora Serverless v2, consulte Uso de Aurora Serverless v2.	28 de junio de 2023

Amazon Aurora presenta integraciones sin ETL con Amazon Redshift (versión preliminar)	Las integraciones sin ETL proporcionan una solución totalmente administrada que permite que los datos transaccionales estén disponibles en Amazon Redshift en cuestión de segundos después de escribirlos en un clúster de base de datos de Aurora MySQL. Para obtener más información, consulte Trabajar con integraciones Aurora sin ETL con Amazon Redshift.	28 de junio de 2023
Amazon RDS ofrece una vista combinada de las métricas de Información de rendimiento y CloudWatch en el panel de Información de rendimiento	Amazon RDS ahora ofrece una vista consolidada de las métricas de Información de rendimiento y CloudWatch en el panel de Información de rendimiento. Para obtener más información, consulte Consulta de las métricas combinadas en la consola de Amazon RDS.	24 de mayo de 2023
Amazon Aurora es compatible con las clases de instancia db.r7g	Ahora puede usar clases de instancia db.r7g para crear clústeres de base de datos de Aurora. Para obtener más información, consulte Clases de instancia de base de datos de Aurora.	11 de mayo de 2023

Amazon Aurora admite una nueva configuración de almacenamiento de clústeres de bases de datos	Con Aurora I/O-Optimized, solo paga por el uso y el almacenamiento de sus clústeres de bases de datos, sin cargos adicionales por las operaciones de E/S de lectura y escritura. Para obtener más información, consulte Configuraciones de almacenamiento para los clústeres de bases de datos de Amazon Aurora.	11 de mayo de 2023
Amazon Aurora admite Aurora Serverless v2 en más Regiones de AWS	Ahora puede crear clústeres de base de datos de Aurora Serverless v2 en las siguiente s:Regiones de AWS Asia-Pacífico (Hyderabad), Europa (España), Europa (Zúrich) y Oriente Medio (EAU). Para obtener más información acerca de Aurora Serverles s v2, consulte Uso de Aurora Serverless v2.	4 de mayo de 2023

Aurora Serverless v1 admite la conversión a clústeres aprovisionados	Puede convertir un clúster de base de datos de Aurora Serverless v1 directamente en un clúster de base de datos aprovisionado. Para obtener más información, consulte Converting an Aurora Serverless v1 DB clúster to provisioned (Conversión de un clúster de base de datos de Aurora sin servidor v1 en un clúster aprovisionado).	27 de abril de 2023
Aurora Serverless v1 admite Amazon Aurora PostgreSQL versión 13	Ahora puede crear clústeres de bases de datos de Aurora Serverless v1 que ejecuten la versión 13 de Aurora PostgreSQL. Para obtener más información, consulte Aurora Serverless v1.	27 de abril de 2023
Soporte de Amazon Aurora para AWS Secrets Manager en las regiones de China	Amazon Aurora admite Secrets Manager en las regiones China (Pekín) y China (Ningxia) y China (Ningxia) y China (Ningxia). Para obtener más informaci ón, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.	20 de abril de 2023

Amazon Aurora admite la publicación de eventos con etiquetas para los suscriptores de temas	Las notificaciones de eventos de Amazon Aurora enviadas a Amazon Simple Notification Service (Amazon SNS) o Amazon EventBridge contienen ahora etiquetas de eventos en el cuerpo del mensaje. Estas etiquetas proporcionan los datos del recurso afectado por el evento de servicio. Para obtener más información, consulte Amazon RDS event notification tags and attributes (Etiquetas y atributos de notificación de eventos de Amazon RDS).	17 de abril de 2023
Actualización de permisos de roles vinculados a servicios de IAM	Las políticas `AmazonRDSFullAccess` y `AmazonRDSReadOnlyAccess` ahora otorgan permisos adicionales para permitir mostrar los resultados de Amazon DevOps Guru en la consola de RDS. Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	30 de marzo de 2023

Amazon Aurora admite bases de datos en la región Asia-Pacífico (Melbourne	Ahora puede crear bases de datos globales de Aurora en la región Asia-Pacífico (Melbourne). Para obtener información sobre bases de datos globales de Aurora, consulte Uso de bases de datos globales de Amazon Aurora.	22 de marzo de 2023
Actualización de los permisos de políticas administradas de AWS	Las políticas `AmazonRDSFullAccess` y `AmazonRDSReadOnlyAccess` ahora otorgan permisos adicionales a Amazon CloudWatch. Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	16 de marzo de 2023
El proxy de RDS está disponible en las regiones de China	El proxy de RDS ya está disponible en las regiones China (Pekín) y China (Ningxia). Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	15 de marzo de 2023
Amazon Aurora admite Aurora Serverless v2 en las regiones de China	Aurora Serverless v2 ya está disponible en las regiones China (Pekín) y China (Ningxia). Para obtener más información, consulte Aurora Serverless v2.	15 de marzo de 2023

El proxy de RDS está disponible en la región Asia-Pacífico (Yakarta	El proxy de RDS ya está disponible en la región Asia-Pacífico (Yakarta). Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	8 de marzo de 2023
Amazon Aurora MySQL admite la autenticación Kerberos	Ahora puede usar la autenticación Kerberos para autenticar a los usuarios cuando estos se conecten a sus clústeres de base de datos de Aurora MySQL. Para obtener más información, consulte Using Kerberos authentication for Aurora MySQL (Uso de la autenticación Kerberos para Aurora MySQL).	8 de marzo de 2023
Amazon Aurora admite bases de datos globales en adicionales Regiones de AWS	Ahora puede crear bases de datos globales de Aurora en las siguientes regiones: África (Ciudad del Cabo), Asia-Pacífico (Hong Kong), Asia-Pacífico (Hyderabad), Asia-Pacífico (Yakarta), Europa (Milán), Europa (España) Europa (Zúrich), Medio Oriente (Bahréin) y Oriente Medio (Emiratos Árabes Unidos). Para obtener información sobre bases de datos globales de Aurora, consulte Uso de bases de datos globales de Amazon Aurora.	6 de marzo de 2023

Amazon Aurora admite la copia de instantáneas de clústeres de bases de datos en adicionales Regiones de AWS	Ahora puede copiar instantáneas de clústeres de bases de datos en las siguientes regiones: África (Ciudad del Cabo), Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Europa (Milán), Europa (España) Europa (Zúrich), Medio Oriente (Bahréin) y Oriente Medio (Emiratos Árabes Unidos). Para obtener más información acerca de la copia de instantáneas de clústeres de base de datos, consulte Copia de una instantánea de clúster de base de datos.	6 de marzo de 2023
Amazon DevOps Guru para RDS admite información proactiva	Amazon DevOps Guru para RDS publica información proactiva con recomendaciones para ayudarlo a solucionar los problemas de sus bases de datos de Aurora antes de que se produzcan. Para obtener más información, consulte How DevOps Guru for RDS works (Cómo funciona DevOps Guru par RDS).	28 de febrero de 2023

La versión 1 de Amazon Aurora MySQL está obsoleta	La versión 1 de Aurora MySQL (compatible con MySQL 5.6) está obsoleta. Para obtener más información, consulte Cuánto tiempo permanecen disponibles las versiones principales de Amazon Aurora.	28 de febrero de 2023
Aurora Serverless v1 permite la configuración del periodo de mantenimiento del clúster de base de datos	Ahora puede configurar el periodo de mantenimiento de los clústeres de base de datos de Aurora Serverless v1. Para obtener más información, consulte Ajuste de la ventana de mantenimiento preferida para un clúster de base de datos.	27 de febrero de 2023
Amazon Aurora admite flujos de actividad de base de datos en las regiones de Asia-Pacífico (Hyderabad), Europa (España) y Oriente Medio (EAU.	Para obtener más información, consulte Secuencias de actividades de la base de datos.	27 de enero de 2023
Amazon Aurora está disponible en la región de Asia-Pacífico (Melbourne	Amazon Aurora está disponible ahora en la región de Asia-Pacífico (Melbourne). Para obtener más información, consulta Regiones y zonas de disponibilidad.	23 de enero de 2023

Especificar la entidad de certificación (CA) durante la creación del clúster de base de datos	Ahora puede especificar qué CA usar para el certificado de servidor de un clúster de base de datos al crear el clúster de base de datos. Para obtener más información, consulte Certificate authorities (Entidades de certificación).	5 de enero de 2023
Aurora MySQL 3.* es compatible con la búsqueda de datos anteriores	Las versiones de Aurora MySQL 3* ofrecen ahora una forma rápida de recuperar se de los errores de usuario como, por ejemplo, anular la tabla incorrecta o eliminar la fila equivocada. Backtrack le permite trasladar su base de datos a un punto anterior en el tiempo sin necesidad de restaurarla desde una copia de seguridad, y se completa en segundos, incluso para bases de datos grandes. Para obtener más información, consulte Búsqueda de datos anteriores de un clúster de bases de datos de Aurora.	4 de enero de 2023

Uso de las implementaciones azules/verdes de Amazon RDS, disponibles en adicionales Regiones de AWS	La función de implementaciones azul/verde ya está disponible en las regiones de China (Pekín) y China (Ningxia). Para obtener más información, consulte Uso de las implementaciones azul/verde de Amazon RDS para actualizar las bases de datos.	22 de diciembre de 2022
Actualización de permisos de roles vinculados a servicios de IAM	La política AmazonRDSServiceRolePolicy ahora otorga permisos adicionales a AWS Secrets Manager. Para obtener más información, consulte Actualizaciones de Amazon RDS de las políticas administradas de AWS.	22 de diciembre de 2022
Amazon Aurora se integra con AWS Secrets Manager para la administración de contraseñas	Aurora puede administrar la contraseña de usuario maestro de un clúster de base de datos en Secrets Manager. Para obtener más información, consulte Administración de contraseñas con Amazon Aurora y AWS Secrets Manager.	22 de diciembre de 2022
Amazon Aurora admite Aurora Serverless v2 en más Regiones de AWS	Aurora Serverless v2 ya está disponible en las regiones de África (Ciudad del Cabo) y Europa (Milán). Para obtener más información, consulte Aurora Serverless v2.	21 de diciembre de 2022

Aurora PostgreSQL admite RDS Proxy con PostgreSQL 14	Ahora puede crear un RDS Proxy con un clúster de bases de datos de Aurora PostgreSQL 14. Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	13 de diciembre de 2022
Amazon Aurora le alerta sobre las anomalías recientes detectadas por Amazon DevOps Guru	La página de detalles de la base de datos de la consola le avisa sobre las anomalías actuales y las que se han producido en las últimas 24 horas. Para obtener más información, consulte How DevOps Guru for RDS works.	13 de diciembre de 2022
Amazon RDS Proxy admite bases de datos globales	Ahora puede usar RDS Proxy con bases de datos globales de Aurora. Para obtener más información, consulte Using RDS Proxy with Aurora global databases (Uso de RDS Proxy con bases de datos globales de Aurora).	7 de diciembre de 2022

Los clústeres de base de datos de Aurora PostgreSQL admiten Extensiones de lenguaje de confianza para PostgreSQL	Extensiones de lenguaje de confianza para PostgreSQL es un kit de desarrollo de código abierto que le permite crear extensiones de PostgreSQL de alto rendimiento y ejecutarlas de forma segura en su clúster de base de datos de Aurora para PostgreSQL. Para obtener más información, consulte Working with Trusted Language Extensions for PostgreSQL (Trabajar con extensiones de lenguaje de confianza para PostgreSQL).	30 de noviembre de 2022

Amazon GuardDuty RDS Protection monitorea las amenazas de acceso	Al activar GuardDuty RDS Protection, GuardDuty consume los eventos de inicio de sesión de RDS de sus bases de datos de Aurora, los monitorea y los perfila para detectar posibles amenazas internas o actores externos. Cuando GuardDuty RDS Protection detecta una amenaza potencial, GuardDuty genera un nuevo hallazgo con detalles sobre la base de datos potencial mente comprometida. Para obtener más información, consulte Monitoring threats with GuardDuty RDS Protectio n (Monitoreo de amenazas con GuardDuty RDS Protectio n).	30 de noviembre de 2022

Uso de las implementaciones azules/verdes de Amazon RDS para actualizar las bases de datos	Puede realizar cambios en un clúster de base de datos en un entorno transitorio y probar los cambios sin que ello afecte al clúster de base de datos de producción. Cuando esté listo, puede promover el entorno transitorio para que sea el nuevo entorno de producció n, con un tiempo de inactivid ad mínimo. Para obtener más información, consulte Using Amazon RDS Blue/Green Deployments for database updates (Uso de implement aciones azul/verde de Amazon RDS para las actualizaciones de bases de datos).	27 de noviembre de 2022
Amazon Aurora está disponibl e en la región de Asia-Pacífico (Hyderabad	Amazon Aurora está disponibl e ahora en la región de Asia-Pacífico (Hyderabad). Para obtener más información, consulta Regiones y zonas de disponibilidad.	22 de noviembre de 2022
Amazon Aurora está disponibl e en la región de Europa (España	Amazon Aurora está disponibl e ahora en la región de Europa (España) Para obtener más información, consulta Regiones y zonas de disponibi lidad.	16 de noviembre de 2022

Amazon Aurora está disponible en la región de Europa (Zúrich	Amazon Aurora está disponible ahora en la región de Europa (Zúrich). Para obtener más información, consulta Regiones y zonas de disponibilidad.	9 de noviembre de 2022
Amazon Aurora admite la exportación de datos a Amazon S3 desde clústeres de base de datos	Ahora puede exportar los datos del clúster Aurora directamente a S3, sin tener que crear primero una instantánea. Para obtener más información, consulte Exporting DB clúster data to Amazon S3 (Exportación de datos de clúster de base de datos a Amazon S3).	27 de octubre de 2022
Amazon Aurora MySQL admite exportaciones más rápidas a Amazon S3	Ahora puee obtener un rendimiento hasta 10 veces más rápido al exportar datos de instantáneas del clúster de base de datos a S3 para clústeres de Aurora MySQL compatibles con MySQL 5.7 y 8.0. Para obtener más información, consulte Exportación de datos de instantánea del clúster de bases de datos a Amazon S3.	20 de octubre de 2022

Amazon Aurora admite la configuración automática de la conectividad entre un clúster de base de datos de Aurora y una instancia de EC	Puede utilizar la Consola de administración de AWS para configurar la conectividad entre un clúster de base de datos de Aurora y una instancia de EC2. Para obtener más información, consulte Conexión automática de una instancia de EC2 y un clúster de base de datos de Aurora.	14 de octubre de 2022
Controlador JDBC de AWS para PostgreSQL disponible de forma general	Controlador JDBC de AWS para MySQL es un controlador de cliente diseñado para Aurora PostgreSQL. El controlador JDBC de AWS para MySQL está ahora disponible de forma general. Para obtener más información, consulte Conexión con el controlador JDBC de AWS para PostgreSQL.	6 de octubre de 2022
Amazon Aurora admite la actualización in situ Aurora MySQL compatible con MySQL	Puede realizar una actualización in situ de un clúster de Aurora MySQL compatible con MySQL 5.7 para convertir un clúster existente en un clúster de Aurora MySQL compatible con MySQL 8.0. Para obtener más información, consulte Actualización de Aurora MySQL 2.x a 3.x.	26 de septiembre de 2022

Performance Insights muestra las 25 principales consultas SQL	En el panel de Performance Insights, la pestaña Top SQL (SQL principales) muestra las consultas SQL que más contribuyen a la carga de base de datos. Para obtener más información, consulte Información general sobre la pestaña Top SQL (SQL principales).	13 de septiembre de 2022
Aurora MySQL es compatible con una nueva clase de instancia de base de datos	Ahora puede utilizar la clase de instancia de base de datos db.r6i para los clústeres de base de datos de Aurora MySQL. Para obtener más información, consulte Clases de instancia de base de datos.	13 de septiembre de 2022
Amazon Aurora está disponible en la región de Oriente Medio (UAE	Amazon Aurora está ahora disponible en la región de Oriente Medio (UAE). Para obtener más información, consulta Regiones y zonas de disponibilidad.	30 de agosto de 2022

Amazon Aurora admite la configuración automática de la conectividad con una instancia de EC	Al crear un clúster de base de datos de Aurora, puede usar la Consola de administración de AWS para configurar la conectividad entre una instancia de Amazon Elastic Compute Cloud y el nuevo clúster de base de datos. Para obtener más información, consulte Configurar la conectividad de red automática con una instancia de EC2.	22 de agosto de 2022
Amazon Aurora admite el modo de pila doble	Los clústeres de base de datos ahora se pueden ejecutar en el modo de pila doble. En el modo de pila doble, los recursos se pueden comunicar con el clúster de base de datos mediante IPv4, IPv6 o ambos. Para obtener más información, consulte Direccionamiento IP de Amazon Aurora.	17 de agosto de 2022

Amazon Aurora admite la actualización in situ de compatible con PostgreSQL Aurora Serverless v1	Puede realizar una actualización in situ para un clúster de Aurora Serverless v1 compatible con PostgreSQL 10 para cambiar un clúster existente por un clúster de Aurora Serverless v1 compatible con PostgreSQL 11. Para conocer el procedimiento de actualización in situ, consulte Modificación de un clúster de base de datos de Aurora Serverless v1.	8 de agosto de 2022
Performance Insights admite la región Asia-Pacífico (Yakarta	Anteriormente, no se podía usar Performance Insights en la región Asia-Pacífico (Yakarta). Esta restricción se ha eliminado. Para obtener más información, consulte Compatibilidad Región de AWS de Performance Insights.	21 de julio de 2022
Amazon Aurora admite una nueva clase de instancia de base de datos	Ahora puede utilizar la clase de instancia de base de datos db.r6i para los clústeres de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Clases de instancia de base de datos.	14 de julio de 2022

RDS Performance Insights admite períodos de retención adicionales	Anteriormente, Performance Insights ofrecía solo dos períodos de retención: 7 días (predeterminado) o 2 años (731 días). Ahora, si necesita retener los datos de rendimiento durante más de 7 días, puede especificar entre 1 y 24 meses. Para obtener más información, consulte Precios y retención de datos para Performance Insights.	1 de julio de 2022
Amazon Aurora admite la actualización in situ de compatible con MySQL Aurora Serverless v1	Puede realizar una actualización in situ de un clúster de Aurora Serverless v1 compatible con MySQL 5.6 para convertir un clúster existente en un clúster de Aurora Serverless v1 compatible con MySQL 5.7. Para conocer el procedimiento de actualización in situ, consulte Modificación de un clúster de base de datos de Aurora Serverless v1.	16 de junio de 2022
Aurora admite la activación de Amazon DevOps Guru en la consola de RDS	Puede activar la cobertura de DevOps Guru para sus bases de datos Aurora desde la consola de RDS. Para obtener más información, consulte Setting up DevOps Guru for RDS (Configuración de DevOps Guru para RDS).	9 de junio de 2022

Amazon Aurora admite la publicación de eventos en temas cifrados de Amazon SNS	Amazon Aurora ahora puede publicar eventos en los temas de Amazon Simple Notification Service (Amazon SNS) que tengan habilitado el cifrado del servidor (SSE), para ofrecer una protección adicional de los eventos que incluyen datos confidenciales. Para obtener más información, consulte Suscripción a notificaciones de eventos de Amazon RDS.	1 de junio de 2022
Amazon RDS publica métricas de uso en Amazon CloudWatch	El espacio de nombres `AWS/Usage` de Amazon CloudWatch incluye métricas de uso en el nivel de cuenta para sus cuotas de servicio de Amazon RDS. Para obtener más información, consulte el tema sobre métricas de uso de Amazon CloudWatch para Amazon Aurora.	28 de abril de 2022

Conjuntos de resultados de API de datos en formato JSON	Un parámetro opcional para la función `ExecuteStatement` hace que el conjunto de resultados de la consulta se devuelva como una cadena en formato JSON. El conjunto de resultados JSON es sencillo y cómodo de transformar en una estructura de datos en el lenguaje de su aplicación. Para obtener más información, consulte Procesamiento de resultados de consulta en formato JSON.	27 de abril de 2022
Amazon Aurora Serverless v2 ya está disponible con carácter general	Amazon Aurora Serverless v2 está disponible con carácter general para todos los usuarios. Para obtener más información, consulte Uso de Aurora Serverless v2.	21 de abril de 2022
Aurora MySQL admite conjuntos de cifrado configurables	Aurora MySQL permite controlar los conjuntos de cifrado configurables para controlar el cifrado de conexión que acepta el servidor de base de datos. Para obtener más información, consulte el tema sobre configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora MySQL.	15 de abril de 2022

Aurora PostgreSQL admite RDS Proxy con PostgreSQL 13	Ahora puede crear un RDS Proxy con un clúster de bases de datos de Aurora PostgreSQL 13. Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	4 de abril de 2022
Notas de la versión de Aurora PostgreSQL	Ahora hay una guía aparte para las notas de la versión de Amazon Aurora PostgreSQL. Para obtener más información, consulte las notas de la versión de Aurora PostgreSQL.	22 de marzo de 2022
Notas de la versión de Aurora MySQL	Ahora hay una guía aparte para las notas de la versión de Amazon Aurora MySQL. Para obtener más información, consulte las notas de la versión de Aurora MySQL.	22 de marzo de 2022
Aurora PostgreSQL admite actualizaciones de varias versiones principales	Ahora puede realizar actualizaciones de versiones de clústeres de base de datos de Aurora PostgreSQL en varias versiones principales. Para obtener más información, consulte Cómo llevar a cabo una actualización de versión principal.	4 de marzo de 2022

Aurora PostgreSQL admite conjuntos de cifrado configurables	Con las versiones 11.8 y posteriores de Aurora PostgreSQL, ahora puede utilizar conjuntos de cifrado configurables para controlar el cifrado de conexión que acepta el servidor de la base de datos. Para obtener información sobre cómo utilizar conjuntos de cifrado configurables con Aurora PostgreSQL, consulte Configuración de conjuntos de cifrado para conexiones a clústeres de base de datos de Aurora PostgreSQL.	4 de marzo de 2022
AWS JDBC Driver for MySQL disponible de forma general	AWS JDBC Driver for MySQL es un controlador de cliente diseñado para la alta disponibilidad de Aurora MySQL. AWS JDBC Driver for MySQL está ahora disponible de forma general. Para obtener más información, consulte Conexión con Amazon Web Services JDBC Driver for MySQL.	2 de marzo de 2022

Aurora PostgreSQL 13.5 admite Babelfish para Aurora PostgreSQL 1.1.0	Babelfish 1.1.0 es compatible con Aurora PostgreSQL 13.5. Para ver una lista de las nuevas características, consulte 13.5. Para ver una lista de las características admitidas en cada versión de Babelfish, consulte Funcionalidades compatibles con Babelfish por versión. Para obtener más información, consulte Uso de Babelfish para Aurora PostgreSQL.	28 de febrero de 2022
Amazon Aurora admite flujos de actividad de base de datos en la región Asia-Pacífico (Yakarta	Para obtener más información, consulte Compatibilidad de los flujos de actividad de bases de datos con las Regiones de AWS.	16 de febrero de 2022
Performance Insights admite nuevas operaciones de API	Performance Insights admite las siguientes operaciones de API: `GetResourceMetadata` , `ListAvailableResourceDimensions` y `ListAvailableResourceMetrics` . Para obtener información, consulte Recuperación de métricas con la API de Performance Insights en este manual y la Referencia de la API de Información de rendimiento de Amazon RDS.	12 de enero de 2022

Amazon RDS Proxy admite eventos	RDS Proxy ahora genera eventos a los que puede suscribirse y ver en CloudWatch Events o configurar para enviarlos a Amazon EventBridge. Para obtener más información, consulte Trabajo con eventos de RDS Proxy.	11 de enero de 2022
RDS Proxy está disponible en más Regiones de AWS	RDS Proxy ya está disponible en las siguientes regiones: África (Ciudad del Cabo), Asia-Pacífico (Hong Kong), Asia-Pacífico (Osaka), Europa (Milán), Europa (París), Europa (Estocolmo), Medio Oriente (Baréin) y América del Sur (São Paulo). Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	5 de enero de 2022
Amazon Aurora está disponible en la región de Asia-Pacífico (Yakarta	Amazon Aurora está disponible ahora en la región de Asia-Pacífico (Yakarta). Para obtener más información, consulta Regiones y zonas de disponibilidad.	13 de diciembre de 2021

DevOps Guru para Amazon RDS proporciona información detallada y recomendaciones para Amazon Aurora	DevOps Guru para RDS extrae Performance Insights para datos relacionados con el rendimiento. Con estos datos, el servicio analiza el rendimiento de las instancias de base de datos de Amazon Aurora y puede ayudarle a resolver problemas de rendimiento. Para obtener más información, consulte Análisis de anomalías de rendimiento con DevOps Guru para RDS en esta guía y consulte Información general sobre DevOps Guru para RDS en la Guía del usuario de Amazon DevOps Guru.	1 de diciembre de 2021
Aurora PostgreSQL admite proxy RDS con PostgreSQL 12	Ahora puede crear un proxy RDS con un clúster de bases de datos Aurora PostgreSQL 12. Para obtener más información sobre RDS Proxy, consulte Uso de Amazon RDS Proxy.	22 de noviembre de 2021
Aurora es compatible con clases de instancias de Graviton2 de AWS para secuencias de actividades de base de datos	Puede utilizar flujos de actividad de base de datos con la clase de instancia db.r6g para Aurora MySQL y Aurora PostgreSQL. Para obtener más información, consulte Clases de instancia de base de datos compatibles.	3 de noviembre de 2021

Compatibilidad de Amazon Aurora con el acceso entre cuentas de AWS KMS keys	Para el cifrado, puede usar una clave KMS desde otra Cuenta de AWS cuando exporte instantáneas de base de datos a Amazon S3. Para obtener más información, consulte Exportación de datos de instantáneas de bases de datos a Amazon S3.	3 de noviembre de 2021
Amazon Aurora admite Babelfish para Aurora PostgreSQL	Babelfish for Aurora PostgreSQL amplía la base de datos de edición compatible con Amazon Aurora PostgreSQL con la capacidad de aceptar conexiones de bases de datos de clientes de Microsoft SQL Server. Para obtener más información, consulte Uso de Babelfish for Aurora PostgreSQL.	28 de octubre de 2021
Aurora Serverless La v1 de puede requerir SSL para las conexiones	Los parámetros de clúster `force_ssl` de Aurora para PostgreSQL y `require_secure_transport` para MySQL ya son compatibles con la v1 de Aurora Serverless. Para obtener más información, consulte Uso de TLS/SSL con la v1 de Aurora Serverless.	26 de octubre de 2021

Amazon Aurora admite Performance Insights en adicionales Regiones de AWS	Información sobre rendimiento está disponible en las regiones de Medio Oriente (Baréin), África (Ciudad del Cabo), Europa (Milán) y Asia-Pacífico (Osaka). Para obtener más información, consulte Compatibilidad Región de AWS de Performance Insights.	5 de octubre de 2021
Tiempo de espera de escalado automático configurable para Aurora Serverless v1	Puede elegir cuánto tiempo esperará Aurora Serverless v1 para buscar un punto de escalado automático. Si no se encuentra ningún punto de escalado automático durante ese período, Aurora Serverless v1 cancelará el evento de escalado o forzará el cambio de capacidad, dependiendo de la acción de tiempo de espera que se haya seleccionado. Para obtener más información, consulte Escalado automático de Aurora Serverless v1.	10 de septiembre de 2021

Aurora es compatible con las clases de instancia X2g y T4g	Tanto Aurora MySQL como Aurora PostgreSQL ahora pueden utilizar las clases de instancia X2G y T4g. Las clases de instancia que se pueden utilizar dependen de la versión de Aurora MySQL o Aurora PostgreSQL. Para obtener más información sobre los tipos de instancias admitidas, consulte Clases de instancias de base de datos.	10 de septiembre de 2021
Amazon RDS admite RDS Proxy en una VPC compartida	A partir de ahora puede crear un RDS Proxy en una nube privada virtual (VPC) compartida. Para obtener más información acerca del proxy de RDS, consulte “Administración de conexiones con el proxy de Amazon RDS” en la Guía del usuario de Amazon RDS o en la Guía del usuario de Aurora.	6 de agosto de 2021
Página de política de versión de Aurora	La guía del usuario de Amazon Aurora ahora incluye una sección con información general sobre las versiones de Aurora y las políticas asociadas. Para obtener más detalles, consulte las versiones de Amazon Aurora.	14 de julio de 2021

Exclusión de eventos de API de datos de un registro de seguimiento de AWS CloudTrail	Puede excluir eventos de API de datos de un registro de seguimiento de CloudTrail. Para obtener más informaci ón, consulte la sección sobre excluir eventos de API de datos de un registro de seguimiento de AWS CloudTrail.	2 de julio de 2021
Amazon Aurora edición compatible con PostgreSQL admite extensiones adicional es	Las extensiones admitidas recientemente incluyen pg_bigm, pg_cron, pg_partma n y pg_proctab. Para obtener más información, consulte la sección sobre las versiones de extensión para Amazon Aurora PostgreSQL-Compati ble Edition.	17 de junio de 2021
Clonación de clústeres de Aurora Serverless	Ahora puede crear clústeres clonados que son de Aurora Serverless. Para obtener más información acerca de la clonación, consulte Clonación de un volumen de clúster de bases de datos de Aurora.	16 de junio de 2021
Las bases de datos globales de Aurora están disponibles en las regiones China (Pekín) y China (Ningxia	Ahora puede crear bases de datos globales de Aurora en las regiones China (Pekín) y China (Ningxia). Para obtener información sobre bases de datos globales de Aurora, consulte Funcionamiento de bases de datos globales de Amazon Aurora.	19 de mayo de 2021

Compatibilidad con FIPS 140-2 para API de datos	La API de datos es compatible con el estándar federal de procesamiento de información 140-2 (FIPS 140-2) para conexiones SSL/TLS. Para obtener más información, consulte Disponibilidad de API de datos.	14 de mayo de 2021
AWS JDBC Driver for PostgreSQL (vista previa)	AWS JDBC Driver for PostgreSLQ, ahora disponible para previsualización, es un controlador de cliente diseñado para la alta disponibilidad de Aurora PostgreSQL. Para obtener más información, consulte Conexión con Amazon Web Services JDBC Driver para PostgreSQL (vista previa).	27 de abril de 2021
API de datos disponible en adicionales Regiones de AWS	La API de datos ya está disponible en las regiones Asia Pacífico (Seúl) y Canadá (Central). Para obtener más información, consulte Disponibilidad de la API de datos.	9 de abril de 2021

Amazon Aurora es compatible con las clases de instancia de base de datos de Graviton	Ahora puede utilizar las clases de instancia de base de datos de Graviton2 db.r6g.x para crear clústeres de base de datos que ejecuten MySQL o PostgreSQL. Para obtener más información, consulte Tipos de clase de instancia de base de datos.	12 de marzo de 2021
Mejoras de los puntos de conexión de RDS Proxy	Puede crear puntos de enlace adicionales asociados con cada proxy de RDS. La creación de un punto de enlace en una VPC diferente permite el acceso entre VPC para el proxy. Los proxies para los clústeres de Aurora MySQL también pueden tener puntos de enlace de solo lectura. Estos puntos de enlace del lector se conectan a las instancias de base de datos del lector en los clústeres y pueden mejorar la escalabilidad de lectura y la disponibilidad para aplicaciones que requieren un uso intensivo de consultas. Para obtener más información acerca del proxy de RDS, consulte “Administración de conexiones con el proxy de Amazon RDS” en la Guía del usuario de Amazon RDS o en la Guía del usuario de Aurora.	8 de marzo de 2021

Amazon Aurora está disponible en la región de Asia-Pacífico (Osaka	Amazon Aurora está disponible ahora en la región de Asia-Pacífico (Osaka). Para obtener más información, consulte Regiones y zonas de disponibilidad.	1 de marzo de 2021
Aurora PostgreSQL es compatible con la habilitación de las autenticaciones IAM y Kerberos en el mismo clúster de bases de datos	Aurora PostgreSQL ahora es compatible con la habilitación de las autenticaciones IAM y Kerberos en el mismo clúster de bases de datos. Para obtener más información, consulte Autenticación de bases de datos con Amazon Aurora.	24 de febrero de 2021

La base de datos global de Aurora ahora admite conmutación por error planificada administrada	La base datos global de Aurora ahora admite la conmutación por error planificada administrada, lo que le permite cambiar más fácilmente la región principal de AWS de la base de datos global de Aurora. Puede utilizar la conmutación por error planificada administrada solo con bases de datos globales de Aurora en buen estado. Para obtener más información, consulte Recuperación ante desastres y bases de datos Amazon Aurora globales. Para obtener información de referencia, consulte `FailoverGlobalCluster` en Amazon RDS API Reference.	11 de febrero de 2021

Data API por Aurora Serverless ahora soporta más tipos de datos	Con la API de datos para Aurora Serverless, ahora puede utilizar tipos de datos `UUID` y `JSON` como entrada en su base de datos. También con la API de datos para Aurora Serverless, ahora puede tener un valor de tipo `LONG` devuelto de su base de datos como un valor `STRING`. Para obtener más información, consulte Llamar a la API de datos. Para obtener información de referencia acerca de los tipos de datos admitidos, consulte `SqlParameter` en Referencia de la API del servicio de datos de Amazon RDS.	2 de febrero de 2021
Aurora PostgreSQL admite actualizaciones de versiones principales a PostgreSQL 12	Con Aurora PostgreSQL, ahora puede actualizar el motor de base de datos a una versión 12 principal. Para obtener más información, consulte Actualización del motor de base de datos PostgreSQL para Aurora PostgreSQL.	28 de enero de 2021

Aurora MySQL admite la actualización local	Puede actualizar su clúster Aurora MySQL 1.x a Aurora MySQL 2.x, conservando las instancias de base de datos, los puntos de enlace, etc. del clúster original. Esta técnica de actualización en el sitio evita el inconveni ente de configurar un clúster completamente nuevo mediante la restauración de una instantánea. También evita la sobrecarga de copiar todos los datos de la tabla en un nuevo clúster. Para obtener más información, consulte Actualización de la versión principal de un clúster de bases de datos Aurora MySQL de 1.x a 2.x.	11 de enero de 2021
AWS JDBC Driver for MySQL (vista previa)	AWS JDBC Driver for MySQL, ahora disponible para vista previa, es un controlador de cliente diseñado para la alta disponibilidad de Aurora MySQL. Para obtener más información, consulte Conexión con Amazon Web Services JDBC Driver for MySQL (vista previa).	7 de enero de 2021

Aurora admite flujos de actividad de la base de datos en clústeres secundarios de una base de datos global	Puede iniciar una base de datos, una secuencia de actividad de la base de datos en un clúster principal o secundario de Aurora PostgreSQL o Aurora MySQL. Para ver las versiones compatibles del motor, consulte Limitaciones de las bases de datos globales de Aurora.	22 de diciembre de 2020
Clústeres de varios maestros con 4 instancias de base de datos	El número máximo de instancias de base de datos en un clúster de varios maestros Aurora MySQL es ahora cuatro. Anteriormente, el máximo era dos instancia s de base de datos. Para obtener más información, consulte Uso de clústeres multimaestros de Aurora.	17 de diciembre de 2020
Aurora PostgreSQL es compatible con las funciones de AWS Lambda	Ahora puede invocar la función AWS Lambda para sus clústeres de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Invocación de una función de Lambda desde un clúster de bases de datos de Aurora PostgreSQL.	11 de diciembre de 2020

Amazon Aurora admite las clases de instancia de base de datos de Graviton2 en vista previa	Ahora puede utilizar las clases de instancia de base de datos de Graviton2 db.r6g.x en vista previa para crear clústeres de base de datos que ejecuten MySQL o PostgreSQL. Para obtener más información, consulte Tipos de clase de instancia de base de datos.	11 de diciembre de 2020
Amazon Aurora Serverless v2 está ahora disponible en vista previa.	Amazon Aurora Serverless v2 está disponible en vista previa. Para trabajar con Amazon Aurora Serverless v2, debe solicitar acceso. Para obtener más información, consulte la página Aurora Serverless v2.	1 de diciembre de 2020

Aurora PostgreSQL está ahora disponible para Aurora Serverless en más Regiones de AWS.	Aurora PostgreSQL está ahora disponible para Aurora Serverless en más Regiones de AWS. Ahora puede optar por ejecutar Aurora PostgreSQL Serverless v1 en las mismas Regiones de AWS que ofrecen Aurora MySQL Serverless v1. Las regiones adicionales de Regiones de AWS compatibles con Aurora Serverless son: Oeste de EE. UU. (Norte de California), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Seúl), Asia Pacífico (Bombay), Canadá (centro), Europa (Londres) y Europa (París). Para obtener una lista de todas las regiones y motores de base de datos Aurora admitidos para Aurora Serverless, consulte Aurora sin servidor v1. La API de datos de Amazon RDS para Aurora Serverless también está disponible en estas mismas Regiones de AWS. Para obtener una lista de todas las regiones admitidas para la API de datos para Aurora Serverless, consulte Data API with Aurora sin servidor v1	24 de noviembre de 2020

Performance Insights de Amazon RDS introduce nuevas dimensiones	Puede agrupar la carga de la base de datos según los grupos de dimensiones para la base de datos, la aplicación (PostgreSQL) y el tipo de sesión (PostgreSQL). Amazon RDS también admite las dimensiones db.name, db.application.name (PostgreSQL), db.session_type.name (PostgreSQL) y db.session_type.name (PostgreSQL). Para obtener más información, consulte Tabla de carga superior.	24 de noviembre de 2020
Aurora Serverless admite Aurora PostgreSQL versión 10.12	Aurora PostgreSQL para Aurora Serverless se ha actualizado a Aurora PostgreSQL versión 10.12 en todas las regiones de AWS donde se admite Aurora PostgreSQL para Aurora Serverless. Para obtener más información, consulte Aurora sin servidor v1.	4 de noviembre de 2020

La API de datos ahora admite la autorización basada en etiquetas	La API de datos admite la autorización basada en etiquetas. Si ha etiquetado los recursos del clúster de RDS con etiquetas, puede usar estas etiquetas en sus declaraciones de política para controlar el acceso a través de la API de datos. Para obtener más información, consulte Autorización de acceso a la API de datos.	27 de octubre de 2020
Amazon Aurora amplía el soporte para exportar instantáneas a Amazon S3	Ahora puede exportar datos de instantáneas de base de datos a Amazon S3 en todas las comerciales Regiones de AWS. Para obtener más información, consulte Exportación de datos de instantáneas de bases de datos a Amazon S3.	22 de octubre de 2020

La base de datos global de Aurora admite la clonación	Ahora puede crear clones de los clústeres de base de datos primario y secundario de sus bases de datos globales de Aurora. Puede hacerlo utilizando la Consola de administración de AWS y eligiendo la opción de menú Create clone (Crear clon). También puede usar la AWS CLI y ejecutar el comando `restore-db-cluster-to-point-in-time` con la opción `--restore-type copy-on-write` . Mediante la Consola de administración de AWS o AWS CLI, también puede clonar clústeres de base de datos de las bases de datos globales de Aurora en todas las cuentas de AWS. Para obtener más información acerca de la clonación, consulte Clonación de un volumen de clúster de Aurora DB.	19 de octubre de 2020

Amazon Aurora admite el cambio de tamaño dinámico para el volumen del clúster	A partir de Aurora MySQL 1.23 y 2.09, y de Aurora PostgreSQL 3.3.0 y Aurora PostgreSQL 2.6.0, Aurora reduce el tamaño del volumen del clúster después de quitar datos a través de operaciones como `DROP TABLE`. Para aprovechar esta mejora, actualice a una de las versiones adecuadas en función del motor de base de datos que utilice el clúster. Para obtener información acerca de esta caracterí stica y cómo comprobar el espacio de almacenamiento usado y disponible para un clúster de Aurora, consulte Administración de rendimiento y escalado para clústeres de base de datos Aurora.	13 de octubre de 2020
Amazon Aurora admite tamaños de volumen de hasta 128 TiB	Los volúmenes de clúster de Aurora nuevos y existentes ahora pueden crecer hasta un tamaño máximo de 128 tebibytes (TiB). Para obtener más información, consulte Cómo aumenta el almacenam iento de Aurora.	22 de septiembre de 2020

Aurora PostgreSQL admite las clases de instancia de base de datos db.r5 y db.t3 en la región de China (Ningxia	Ahora puede crear clústeres de base de datos de Aurora PostgreSQL en la región de China (Ningxia) que utiliza las clases de instancia de base de datos db.r5 y db.t3. Para obtener más información, consulte Clases de instancia de base de datos.	3 de septiembre de 2020

Mejoras de consultas paralelas de Aurora	Al comenzar con Aurora MySQL 2.09 y 1.23, puede aprovechar las mejoras de la característica de consulta paralela. La creación de un clúster de consultas paralelas ya no requiere un modo de motor especial. Ahora puede activar y desactivar la consulta paralela mediante la opción de configuración `aurora_parallel_query` para cualquier clúster aprovisionado que ejecute una versión de Aurora MySQL compatible. Puede actualizar un clúster existente a una versión de Aurora MySQL compatible y utilizar consultas paralelas, en lugar de crear un nuevo clúster e importar datos en él. Puede utilizar información sobre rendimiento para clústeres de consultas paralelas. Puede detener y comenzar clústeres de consultas paralelas. Puede crear clústeres de consultas paralelas de Aurora compatibles con MySQL 5.7. La consulta paralela funciona para tablas que utilizan el formato de fila `DYNAMIC`. Los clústeres de consultas en paralelo pueden utilizar autenticación de AWS Identity	2 de septiembre de 2020

	and Access Managemen t (IAM). Las instancias de base de datos de lector en clústeres de consultas paralelas pueden aprovecha r el nivel de aislamiento `READ COMMITTED` . Ahora también puede crear clústeres de consultas paralelas en adicionales Regiones de AWS. Para obtener más informaci ón acerca de la característica de consulta paralela y estas mejoras, consulte Trabajar con consultas paralelas de Amazon Aurora MySQL.	
Cambios en el parámetro binlog_rows_query_log_events de Aurora MySQL	Ahora puede cambiar el valor del parámetro de configura ción de Aurora MySQL `binlog_rows_query_ log_events` . Anteriorm ente, este parámetro no se podía modificar.	26 de agosto de 2020

Compatibilidad con actualizaciones de versiones secundarias automáticas para Aurora MySQL	Con Aurora MySQL, la configuración Habilitar actualización automática de versiones secundarias se aplica ahora cuando se especifica para un clúster de bases de datos de Aurora MySQL. Cuando habilita la actualización automática de versiones secundarias, Aurora se actualiza automáticamente a las nuevas versiones secundarias a medida que se publican. Las actualizaciones automáticas se producen durante la ventana de mantenimiento de la base de datos. Para Aurora MySQL, esta característica solo se aplica a Aurora MySQL versión 2, que es compatible con MySQL 5.7. Inicialmente, el procedimiento de actualización automática trae clústeres de base de datos de Aurora MySQL a la versión 2.07.2. Para obtener más información acerca de cómo funciona esta característica con Aurora MySQL, consulte Actualizaciones y parches de base de datos para Amazon Aurora MySQL.	3 de agosto de 2020

Aurora PostgreSQL admite actualizaciones de versiones principales a PostgreSQL versión 11	Con Aurora PostgreSQL, ahora puede actualizar el motor de base de datos a una versión 11 principal. Para obtener más informaci ón, consulte Actualización del motor de base de datos PostgreSQL para Aurora PostgreSQL.	28 de julio de 2020
Amazon Aurora admite AWS PrivateLink	Amazon Aurora ahora admite la creación de puntos de enlace de Amazon VPC para llamadas a la API de Amazon RDS a fin de mantener el tráfico entre aplicaciones y Aurora en la red de AWS. Para obtener más información, consulte Puntos de enlace de la VPC de tipo interfaz y Amazon Aurora (AWS PrivateLink).	9 de julio de 2020

RDS Proxy disponible con carácter general	El proxy de RDS ahora está disponible con carácter general. Se puede utilizar el proxy de RDS con RDS for MySQL, Aurora MySQL, RDS for PostgreSQL y Aurora PostgreSQL para cargas de trabajo de producción. Para obtener más información acerca del proxy de RDS, consulte “Administración de conexiones con el proxy de Amazon RDS” en la Guía del usuario de Amazon RDS o en la Guía del usuario de Aurora.	30 de junio de 2020
Reenvío de escritura de la base de datos global Aurora	Ahora puede habilitar la capacidad de escritura en clústeres secundarios de una base de datos global. Con el reenvío de escritura, se emiten instrucciones DML en un clúster secundario, Aurora reenvía la escritura al clúster principal y los datos actualizados se replican en todos los clústeres secundarios. Para obtener más información, consulte Reenvío de Regiones de AWS secundarias con una base de datos global de Aurora.	18 de junio de 2020

Aurora admite la integración con AWS Backup	Puede utilizar AWS Backup para administrar copias de seguridad de clústeres de Aurora DB. Para obtener información, consulte Información general de copias de seguridad y restauración de un clúster de bases de datos de Aurora.	10 de junio de 2020
Aurora PostgreSQL admite clases de instancia de base de datos db.t3.large	Ahora puede crear clústeres de base de datos de Aurora PostgreSQL que utilicen las clases de instancia de base de datos db.t3.large. Para obtener más información, consulte Clases de instancia de base de datos.	5 de junio de 2020
La base de datos global de Aurora admite PostgreSQL versión 11.7 y el objetivo de punto de recuperación administrado (RPO)	Ahora puede crear bases de datos globales de Aurora para el motor de base de datos PostgreSQL 11.7. También puede administrar cómo una base de datos global de PostgreSQL se recupera de un error mediante un objetivo de punto de recuperación (RPO). Para obtener más información, consulte Recuperación de desastres entre regiones para bases de datos globales de Aurora.	4 de junio de 2020

Aurora MySQL admite la supervisión de bases de datos mediante secuencias de actividades de la base de datos	Ahora, Aurora MySQL incluye secuencias de actividades de la base de datos, que proporcionan una secuencia de datos prácticamente en tiempo real de la actividad de su base de datos relacional. Para obtener más información, consulte Uso de secuencias de actividades de la base de datos.	2 de junio de 2020
Editor de consultas disponible en adicionales Regiones de AWS	El editor de consultas para Aurora sin servidor está ahora disponible en Regiones de AWS adicionales. Para obtener más información, consulte Disponibilidad del editor de consultas.	28 de mayo de 2020
API de datos disponible en adicionales Regiones de AWS	La API de datos está ahora disponible en adicionales Regiones de AWS. Para obtener más información, consulte Disponibilidad de la API de datos.	28 de mayo de 2020
RDS Proxy disponible en Región Canadá (centro	Ahora puede utilizar la vista previa del proxy de RDS en Región Canadá (Central). Para obtener más información acerca del proxy de RDS, consulte Administración de conexiones con el proxy de Amazon RDS (vista previa).	28 de mayo de 2020

Base de datos global de Aurora y réplicas de lectura entre regiones	Para una base de datos global de Aurora, ahora puede crear una réplica de lectura entre regiones de Aurora MySQL desde el clúster principal en la misma región que un clúster secundario. Para obtener más información acerca de Aurora Global Database y las réplicas de lectura entre regiones, consulte Funcionamiento de Amazon Aurora Global Database y Replicación de la base de datos MySQL de Amazon Aurora.	18 de mayo de 2020
RDS Proxy disponible en más Regiones de AWS	Ahora puede utilizar la vista previa del proxy RDS en EE.UU. Oeste (Norte de California), Región de Europa (Londres), Región de Europa (Fráncfort), Región Asia Pacífico (Seúl), Región Asia Pacífico (Bombay), Región Asia Pacífico (Singapur) y Región Asia Pacífico (Sídney). Para obtener más informaci ón acerca del proxy de RDS, consulte Administración de conexiones con el proxy de Amazon RDS (vista previa).	13 de mayo de 2020

La edición compatible con Aurora PostgreSQL admite Microsoft Active Directory en las instalaciones o con alojamiento propio	Ahora puede utilizar un Active Directory local o con alojamiento propio para la autenticación Kerberos de los usuarios cuando se conectan a los clústeres de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Uso de la autenticación Kerberos con Aurora PostgreSQL.	7 de mayo de 2020
Clústeres multimaestro de Aurora MySQL disponibles en más Regiones de AWS	Ahora puede crear clústeres multimaestros de Aurora en Región Asia Pacífico (Seúl), Región Asia Pacífico (Tokio), Región Asia Pacífico (Bombay) y Región de Europa (Fráncfort). Para obtener más información acerca de los clústeres multimaestros, consulte Trabajo con clústeres multimaestros de Aurora.	7 de mayo de 2020
Performance Insights es compatible con el análisis de las estadísticas de la ejecución de consultas de MySQL en Aurora	Ahora puede analizar las estadísticas de ejecución de consultas con la Información sobre rendimiento para las instancias de base de datos de MySQL en Aurora. Para obtener más información, consulte Análisis de estadísticas de la ejecución de consultas.	5 de mayo de 2020

Biblioteca de cliente de Java para la API de datos disponible con carácter general	La biblioteca de cliente de Java para la API de datos ahora está disponible con carácter general. Puede descargar y utilizar una biblioteca de cliente de Java para la API de datos. Le permite asignar sus clases del lado del cliente a las solicitudes y respuestas de la API de datos. Para obtener más información, consulte Uso de la biblioteca de clientes de Java para la API de datos.	30 de abril de 2020
Amazon Aurora está disponible en la región de Europa (Milán	Amazon Aurora está disponible ahora en la región de Europa (Milán). Para obtener más información, consulte Regiones y zonas de disponibilidad.	28 de abril de 2020
Amazon Aurora está disponible en la región de Europa (Milán	Amazon Aurora está disponible ahora en la región de Europa (Milán). Para obtener más información, consulte Regiones y zonas de disponibilidad.	27 de abril de 2020
Amazon Aurora está disponible en la región de África (Ciudad del Cabo	Amazon Aurora está disponible ahora en la región de África (Ciudad del Cabo) Para obtener más información, consulte Regiones y zonas de disponibilidad.	22 de abril de 2020

Aurora PostgreSQL ahora admite las clases de instancia de base de datos db.r5.16x large y db.r5.8xlarge	Ahora puede crear clústeres de bases de datos de Aurora PostgreSQL que ejecuten PostgreSQL y utilicen las clases de instancia de base de datos db.r5.16xlarge, db.r5.8xlarge. Para obtener más información, consulte Especificaciones de hardware para las clases de instancia de base de datos disponibles para Aurora.	8 de abril de 2020
Amazon RDS Proxy for PostgreSQL	Ahora, el proxy de Amazon RDS está disponible para PostgreSQL. Puede utilizar el proxy de RDS para reducir la sobrecarga de la administración de conexiones en el clúster y también la posibilidad de que se produzcan errores de “demasiadas conexiones”. El proxy de RDS se encuentra actualmente en vista previa pública para PostgreSQL. Para obtener más información, consulte Administración de conexiones con el proxy de Amazon RDS (vista previa).	8 de abril de 2020

Las bases de datos globales de Aurora ahora admiten Aurora PostgreSQL	Ahora puede crear bases de datos globales de Aurora para el motor de base de datos PostgreSQL. Una base de datos global de Aurora abarca varias Regiones de AWS, habilita lecturas globales de baja latencia y la recuperación de desastres de interrupciones regionales. Para obtener más información, consulte Trabajo con Amazon Aurora Global Database.	10 de marzo de 2020
Compatibilidad con actualizaciones de la versión principal para Aurora PostgreSQL	Con Aurora PostgreSQL, ahora puede actualizar el motor de base de datos a una versión principal. Al hacerlo, puede acceder a una versión principal más reciente cuando actualiza determinadas versiones del motor PostgreSQL. Para obtener más información, consulte Actualización del motor de base de datos PostgreSQL para Aurora PostgreSQL.	4 de marzo de 2020

Aurora PostgreSQL es compatible con la autenticación Kerberos	Ahora puede usar la autenticación Kerberos para autenticar a los usuarios cuando estos se conecten a sus clústeres de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Uso de la autenticación Kerberos con Aurora PostgreSQL.	28 de febrero de 2020
La API de datos admite AWS PrivateLink	La API de datos ahora admite la creación de puntos de enlace de la Amazon VPC para llamadas a la API de datos a fin de mantener el tráfico entre las aplicaciones y la API de datos en la red de AWS. Para obtener más información, consulte Creación de un punto de enlace de la Amazon VPC (AWS PrivateLink) para la API de datos.	6 de febrero de 2020
Compatibilidad con el Machine Learning de Aurora en Aurora PostgreSQL	La extensión de Aurora PostgreSQL `aws_ml` proporciona funciones que utiliza en sus consultas de base de datos para llamar a Amazon Comprehend para análisis de opiniones e IA de SageMaker para ejecutar sus propios modelos de machine learning. Para obtener más información, consulte Uso de las capacidades del Machine Learning (ML) con Aurora.	5 de febrero de 2020

Aurora PostgreSQL admite la exportación de datos a Amazon S3	Puede consultar datos de un clúster de bases de datos de Aurora PostgreSQL y exportarlos directamente a archivos almacenados en un bucket de Amazon S3. Para obtener más información, consulte Exportación de datos de un clúster de bases de datos de Aurora PostgreSQL a Amazon S3.	5 de febrero de 2020
Compatibilidad para exportar datos de instantáneas de bases de datos a Amazon S3	Amazon Aurora admite la exportación de datos de instantáneas de bases de datos a Amazon S3 para MySQL y PostgreSQL. Para obtener más información, consulte Exportación de datos de instantáneas de bases de datos a Amazon S3.	9 de enero de 2020
Notas de la versión de Aurora MySQL en el historial de documentos	Esta sección ahora incluye entradas del historial para las notas de la versión de Edición compatible con Aurora MySQL para las versiones publicadas después del 31 de agosto de 2018. Para ver las notas de la versión completas de una versión específica, elija el enlace en la primera columna de la entrada del historial.	13 de diciembre de 2019

Amazon RDS Proxy	Puede reducir la sobrecarga de la administración de conexiones en el clúster y reducir la posibilidad de que se produzcan errores de «demasiadas conexiones» mediante el Proxy de Amazon RDS. Asocie cada proxy a una instancia de base de datos de RDS o clúster de bases de datos de Aurora. A continuación, utiliza el punto de enlace de proxy en la cadena de conexión para su aplicación. El proxy de Amazon RDS se encuentra actualmente en estado de vista previa pública. Es compatible con el motor de base de datos de Aurora MySQL. Para obtener más información, consulte Administración de conexiones con el proxy de Amazon RDS (vista previa).	3 de diciembre de 2019
La API de datos para Aurora Serverless v1 es compatible con los tipos de datos que asignen sugerencias	Ahora puede utilizar una sugerencia para indicar la API de datos de Aurora Serverless v1 para enviar un valor de `String` a la base de datos como un tipo diferente. Para obtener más información, consulte Llamadas a la API de datos.	26 de noviembre de 2019

La API de datos para Aurora Serverless v1 es compatible con una biblioteca de cliente de Java (vista previa)	Puede descargar y utilizar una biblioteca de cliente de Java para la API de datos. Le permite asignar sus clases del lado del cliente a las solicitudes y respuestas de la API de datos. Para obtener más información, consulte Uso de la biblioteca de clientes de Java para la API de datos.	26 de noviembre de 2019
Aurora PostgreSQL es apto para FedRAMP HIGH	Aurora PostgreSQL es apto para FedRAMP HIGH. Para obtener más detalles sobre AWS y los esfuerzos de conformidad, consulte Servicios de AWS en el alcance por programa de conformidad.	26 de noviembre de 2019

Nivel de aislamiento READ COMMITTED habilitado para las réplicas de Amazon Aurora MySQL	Ahora puede habilitar el nivel de aislamiento `READ COMMITTED` en las réplicas de Aurora MySQL. Hacerlo precisa habilitar la opción de configuración `aurora_read_replica_read_committed_isolation_enabled` en el nivel de la sesión. El uso del nivel de aislamiento `READ COMMITTED` para consultas de ejecución prolongada en clústeres OLTP puede ayudar a abordar los problemas con la longitud de la lista del historial. Antes de habilitar esta configuración, asegúrese de que comprende en qué se diferencia el comportamiento de aislamiento en las réplicas de Aurora de la implementación de MySQL habitual de `READ COMMITTED`. Para obtener más información, consulte Niveles de aislamiento de Aurora MySQL.	25 de noviembre de 2019

Performance Insights es compatible con el análisis de las estadísticas de la ejecución de consultas de Aurora PostgreSQL	Puede analizar las estadísticas de ejecución de consultas con la información sobre el rendimiento para las instancias de base de datos de PostgreSQL en Aurora. Para obtener más información, consulte Análisis de estadísticas de la ejecución de consultas.	25 de noviembre de 2019
Más clústeres en una base de datos global de Aurora	Ahora puede añadir varias regiones secundarias a una base de datos global de Aurora. Puede aprovecharse de las lecturas globales de baja latencia y de la recuperación tras desastres en un área geográfica más amplia. Para obtener información sobre bases de datos globales de Aurora, consulte Trabajo con Amazon Aurora Global Database.	25 de noviembre de 2019

Compatibilidad con el machine learning de Aurora en Aurora MySQL	En Aurora MySQL 2.07 y posteriores, puede llamar a Amazon Comprehend para el análisis de opiniones y a IA de SageMaker para una amplia variedad de algoritmos de Machine Learning. Puede utilizar los resultados directamente en la aplicación de su base de datos al integrar las llamadas a funciones almacenadas en sus consultas. Para obtener más información, consulte Uso de las capacidades del Machine Learning (ML) con Aurora.	25 de noviembre de 2019

La base de datos global de Aurora ya no precisa la configuración del modo del motor	Ya no tiene que especificar `--engine-mode=global` al crear un clúster que está ideado para formar parte de una base de datos global de Aurora. Todos los clústeres de Aurora que cumplan los requisitos de compatibilidad son aptos para formar parte de una base de datos global. Por ejemplo, actualmente el clúster debe utilizar la versión 1 de Aurora MySQL con la compatibilidad de MySQL 5.6. Para obtener información sobre bases de datos globales de Aurora, consulte Funcionamiento de bases de datos globales de Amazon Aurora.	25 de noviembre de 2019

La base de datos global de Aurora está disponible para la versión 2 de Aurora MySQL 2	A partir de Aurora MySQL 2.07, puede crear una base de datos global de Aurora con compatibilidad con MySQL 5.7. No tiene que especific ar el modo de motor `global` para los clústeres principales o secundarios. Puede añadir cualquier clúster aprovisio nado nuevo con Aurora MySQL 2.07 o posterior a una base de datos global de Aurora. Para obtener información sobre Aurora Global Database, consulte Trabajo con bases de datos globales de Amazon Aurora.	25 de noviembre de 2019
Optimización de la contenció n de filas activas de Aurora MySQL disponible sin modo de laboratorio	La optimización de la contención de filas activas por lo general está disponibl e actualmente para Aurora MySQL y no precisa que la configuración del modo lab de Aurora esté Activo. Esta característica mejora sustancialmente el rendimien to de las cargas de trabajo, ya que muchas transacciones compiten por filas en la misma página. La mejora consiste en cambiar el algoritmo de liberación de bloqueo que usa Aurora MySQL.	19 de noviembre de 2019

Combinaciones hash de Aurora MySQL disponibles sin el modo de laboratorio	La característica de combinación hash por lo general está disponible actualmente para Aurora MySQL y no precisa que la configuración del modo lab de Aurora esté Activo. Esta característica puede mejorar el desempeño de las consultas si necesita unir una gran cantidad de datos mediante equijoin. Para obtener más información sobre el uso de esta característica, consulte Trabajo con combinaciones hash en Aurora MySQL.	19 de noviembre de 2019
Compatibilidad de Aurora MySQL 2.* para más clases de instancias db.r	Los clústeres de Aurora MySQL ahora son compatibles con los tipos de instancias db.r5.8xlarge, db.r5.16xlarge y db.r5.24xlarge. Para obtener más información sobre los tipos de instancias para los clústeres de Aurora MySQL, consulte Elección de la clase de instancia de base de datos.	19 de noviembre de 2019

Aurora MySQL 2.* es compatible con la búsqueda de datos anteriores	Las versiones de Aurora MySQL 2* ofrecen ahora una forma rápida de recuperar se de los errores de usuario como, por ejemplo, anular la tabla incorrecta o eliminar la fila equivocada. Backtrack le permite trasladar su base de datos a un punto anterior en el tiempo sin necesidad de restaurarla desde una copia de seguridad, y se completa en segundos, incluso para bases de datos grandes. Para obtener más información, consulte Búsqueda de datos anteriores de un clúster de bases de datos de Aurora.	19 de noviembre de 2019

Compatibilidad de etiqueta de facturación para Aurora	Ahora puede utilizar etiquetas para realizar un seguimiento de la asignación de costos para recursos tales como clústeres de Aurora, instancias de base de datos dentro de clústeres de Aurora, E/S, copias de seguridad, instantáneas, etc. Puede ver los costos asociados a cada etiqueta con AWS Cost Explorer. Para obtener más información acerca del uso de etiquetas con Aurora, consulte Etiquetado de los recursos de Amazon RDS. Para obtener información general sobre etiquetas y las formas de utilizarlas para análisis de costos, consulte Uso de etiquetas de asignación de costos y Etiquetas de asignación de costos definidas por el usuario.	23 de octubre de 2019
API de datos para Aurora PostgreSQL	Aurora PostgreSQL ahora admite el uso de API de datos con clústeres de base de datos de Amazon Aurora Serverless v1. Para obtener más información, consulte Uso de la API de datos para Aurora Serverless v1.	23 de septiembre de 2019

Aurora PostgreSQL es compatible con la carga de registros de bases de datos en CloudWatch Logs	Puede configurar un clúster de bases de datos de Aurora PostgreSQL para publicar datos de registro en un grupo de registros en registros de Amazon Cloudwatch. Con CloudWatch Logs, puede realizar análisis en tiempo real de los datos de registro y utilizar CloudWatch para crear alarmas y ver métricas. Puede utilizar CloudWatch Logs para almacenar los registros de registros en almacenamiento de larga duración. Para obtener más información, consulte Publicación de registros de Aurora PostgreSQL en registros de Amazon Cloudwatch.	9 de agosto de 2019
Clústeres multimaestro para Aurora MySQL	Puede configurar los clústeres multimaestros de Aurora MySQL. En estos clústeres, cada instancia de base de datos tiene capacidad de lectura/escritura. Para obtener más información, consulte Uso de clústeres multimaestros de Aurora.	8 de agosto de 2019

Aurora PostgreSQL es compatible con Aurora Serverless v1	A partir de ahora, puede utilizar Amazon Aurora Serverless v1 con Aurora PostgreSQL. Un clúster de bases de datos de Aurora sin servidor se inicia, se cierra y escala automáticamente su capacidad informática en función de las necesidad es de la aplicación. Para obtener más información, consulte Uso de Amazon Aurora Serverless v1.	9 de julio de 2019
Clonación entre cuentas para Aurora MySQL	Ahora puede clonar el volumen del clúster para un clúster de bases de datos de Aurora MySQL entre cuentas de AWS. Autoriza el uso compartido a través de AWS Resource Access Manager (AWS RAM). El volumen del clúster clonado utiliza un mecanismo de copia en escritura, que solo requiere almacenamiento adicional para los datos nuevos o modificados. Para obtener más información sobre la clonación de Aurora, consulte Clonación de bases de datos en un clúster de bases de datos Aurora.	2 de julio de 2019

Aurora PostgreSQL admite clases de instancia de base de datos db.t	Ahora puede crear clústeres de base de datos de Aurora PostgreSQL que utilicen las clases de instancia de base de datos db.t3. Para obtener más información, consulte Clase de instancia de base de datos.	20 de junio de 2019
Compatibilidad para importar datos desde Amazon S3 para Aurora PostgreSQL	Ahora puede importar datos de un archivo Amazon S3 en una tabla en un clúster de bases de datos Aurora PostgreSQL. Para obtener más información, consulte Importación de datos de Amazon S3 en un clúster de bases de datos Aurora PostgreSQL.	19 de junio de 2019
Aurora PostgreSQL ahora proporciona una recuperación de conmutación por error rápida con una administración de caché del clúster	Aurora PostgreSQL proporciona ahora administración de caché del clúster para garantizar una recuperación rápida de la instancia de base de datos principal en caso de una conmutación por error. Para obtener más información, consulte Recuperación rápida después de una conmutación por error con la administración de caché del clúster.	11 de junio de 2019

API de datos de Aurora Serverless v1 generalmente disponible	Puede obtener acceso a clústeres de Aurora Serverless v1 con aplicaciones basadas en servicios web mediante la API de datos. Para obtener más información, consulte Uso de la API de datos para Aurora Serverless v1.	30 de mayo de 2019
Aurora PostgreSQL admite la supervisión de bases de datos mediante secuencias de actividades de la base de datos	Ahora, Aurora PostgreSQL incluye secuencias de actividades de la base de datos, que proporcionan un flujo de datos prácticamente en tiempo real de la actividad de su base de datos relacional. Para obtener más información, consulte Uso de secuencias de actividades de la base de datos.	30 de mayo de 2019
Recomendaciones de Amazon Aurora	Amazon Aurora proporciona ahora recomendaciones automatizadas de recursos de Aurora. Para obtener más información, consulte Usar recomendaciones de Amazon Aurora.	22 de mayo de 2019

Compatibilidad de Performance Insights con la base de datos global de Aurora	Ahora puede utilizar Performance Insights con Aurora Global Database. Para obtener información sobre Información sobre rendimiento para Aurora, consulte Uso de Información sobre rendimiento de Amazon RDS. Para obtener información sobre bases de datos globales de Aurora, consulte Funcionamiento de bases de datos globales de Aurora.	13 de mayo de 2019
Información sobre rendimiento está disponible para Aurora MySQL 5.7	Performance Insights de Amazon RDS está ahora disponible para las versiones de Aurora MySQL 2.x, que son compatibles con MySQL 5.7. Para obtener más información, consulte Uso de Información sobre rendimiento de Amazon RDS.	3 de mayo de 2019
Las bases de datos globales de Aurora están disponibles en más Regiones de AWS	Ahora puede crear bases de datos globales de Aurora en la mayoría de Regiones de AWS en las que Aurora está disponible. Para obtener información sobre las bases de datos globales de Aurora, consulte Funcionamiento de bases de datos globales de Amazon Aurora.	30 de abril de 2019

Capacidad mínima de 1 para Aurora Serverless v1	La configuración de capacidad mínima que puede utilizar para un clúster de Aurora Serverless v1 es 1. Anteriormente, el mínimo era 2. Para obtener información sobre la especificación de valores de capacidad de Aurora, consulte Configuración de la capacidad de un clúster de bases de datos de Aurora Serverless v1.	29 de abril de 2019
Aurora Serverless v1 Acción de tiempo de espera de	Ahora puede especificar la acción que realizar cuando se agote el tiempo de espera de un cambio de capacidad de Aurora Serverless v1. Para obtener más información, consulte Acción de tiempo de espera para cambios de capacidad.	29 de abril de 2019
Facturación por segundo	Amazon RDS se factura ahora en incrementos de 1 segundo en todas las Regiones de AWS, excepto GovCloud (EE. UU.) de AWS para las instancias bajo demanda. Para obtener más información, consulte Facturación de instancias de base de datos para Aurora.	25 de abril de 2019

Compartir instantáneas de Aurora Serverless v1 en las regiones de Regiones de AWS	Con Aurora Serverless v1, las instantáneas están siempre cifradas. Si cifra la instantánea con su propia AWS KMS key, ahora podrá copiar o compartir la instantánea en las Regiones de AWS. Para obtener información sobre las instantáneas de clústeres de base de datos de Aurora Serverless v1, consulte Aurora Serverless v1 e instantáneas.	17 de abril de 2019
Restauración de copias de seguridad de MySQL 5.7 desde Amazon S3	Ahora puede crear una copia de seguridad de una base de datos de MySQL, versión 5.7, almacenarla en Amazon S3 y luego restaurar el archivo de copia de seguridad en un nuevo clúster de bases de datos de Aurora MySQL. Para obtener más información, consulte Migración de datos desde una base de datos MySQL externa a un clúster de bases de datos de Aurora MySQL.	17 de abril de 2019

Compartir instantáneas de Aurora Serverless v1en las regiones	Con Aurora Serverless v1, las instantáneas están siempre cifradas. Si cifra la instantánea con su propia AWS KMS key, ahora podrá copiar o compartir la instantánea en las distintas regiones. Para obtener información sobre las instantáneas de clústeres de base de datos de Aurora Serverless v1, consulte Aurora sin servidor e instantáneas.	6 de abril de 2019
Tutorial de prueba de concepto de Aurora	Puede aprender cómo realizar una prueba de concepto para probar su aplicación y carga de trabajo con Aurora. Para ver el tutorial completo, consulte Realización de una prueba de concepto de Aurora.	16 de abril de 2019
Aurora Serverless v1 admite la restauración a partir de una copia de seguridad de Amazon S	Ahora puede importar copias de seguridad desde Amazon S3 en un clúster de Aurora Serverless. Para obtener detalles sobre ese procedimiento, consulte Migración de datos desde MySQL con un bucket de Amazon S3.	16 de abril de 2019

Nuevos parámetros modificables para Aurora Serverless v1	Ahora puede modificar los siguientes parámetros de base de datos para un clúster de Aurora Serverless v1: `innodb_file_format` , `innodb_file_per_table` , `innodb_large_prefix` , `innodb_lock_wait_timeout` , `innodb_monitor_disable` , `innodb_monitor_enable` , `innodb_monitor_reset` , `innodb_monitor_reset_all` , `innodb_print_all_deadlocks` , `log_warnings` , `net_read_timeout` , `net_retry_count` , `net_write_timeout` , `sql_mode` y `tx_isolation` . Para obtener más información sobre los parámetros de configuración para clústeres de Aurora Serverless v1, consulte Aurora Serverless v1 y grupos de parámetros.	4 de abril de 2019
Aurora PostgreSQL admite las clases de instancia de base de datos db.r	Ahora puede crear clústeres de base de datos de Aurora PostgreSQL que utilicen las clases de instancia de base de datos db.r5. Para obtener más información, consulte Clase de instancia de base de datos.	4 de abril de 2019

Replicación lógica de Aurora PostgreSQL	Ahora puede utilizar la replicación lógica de PostgreSQL para replicar partes de una base de datos para un clúster de bases de datos de Aurora PostgreSQL. Para obtener más información, consulte Uso de la replicación lógica de PostgreSQL.	28 de marzo de 2019
Compatibilidad de GTID para Aurora MySQL 2.04	Puede utilizar ahora la replicación con la característica de ID de transacción global (GTID) de MySQL 5.7. Esta característica simplifica la realización de la replicación del registro binario (binlog) entre Aurora MySQL y una base de datos externa compatible con MySQL 5.7. La replicación puede utilizar el clúster de Aurora MySQL como fuente o destino. Esta característica está disponible para Aurora MySQL 2.04 y versiones superiores. Para obtener más información sobre la replicación basada en GTID y Aurora MySQL, consulte Uso de replicación basada en GTID para Aurora MySQL.	25 de marzo de 2019

Carga de registros de Aurora Serverless v1 a Amazon CloudWatch	Ahora puede disponer de registros de base de datos de carga de Aurora en CloudWatch para un clúster de Aurora Serverless v1. Para obtener más información, consulte Visualización de clústeres de base de datos de Aurora sin servidor. Como parte de esta mejora, ahora puede definir valores para parámetros de nivel de instancia en un grupo de parámetros de clúster de bases de datos y esos valores se aplican a todas las instancias de base de datos en el clúster a no ser que los sobrescriba en el grupo de parámetros de base de datos. Para obtener más información, consulte Funcionamiento con grupos de parámetros de base de datos y grupos de parámetros de clúster de bases de datos.	25 de febrero de 2019
Aurora MySQL admite clases de instancia de base de datos db.t	Ahora puede crear clústeres de base de datos de Aurora MySQL que utilicen las clases de instancia de base de datos db.t3. Para obtener más información, consulte Clase de instancia de base de datos.	25 de febrero de 2019

Aurora MySQL admite clases de instancia de base de datos db.r	Ahora puede crear clústeres de base de datos de Aurora MySQL que utilicen las clases de instancia de base de datos db.r5. Para obtener más información, consulte Clase de instancia de base de datos.	25 de febrero de 2019
Contadores de Performance Insights para Aurora MySQL	Ahora puede añadir contadores de rendimiento a sus gráficos de Performance Insights para las instancias de base de datos de Aurora MySQL. Para obtener más información, consulte Componentes del panel de Información sobre rendimiento.	19 de febrero de 2019
Amazon RDS Performance Insights admite la visualización de más texto SQL para Aurora MySQL	Amazon RDS Performance Insights es ahora compatible con la visualización de más texto SQL en el panel de Performance Insights para instancias de base de datos de Aurora MySQL. Para obtener más información, consulte Visualización de más texto SQL en el panel de Información sobre rendimiento.	6 de febrero de 2019

Amazon RDS Performance Insights admite la visualización de más texto SQL para Aurora PostgreSQL	Amazon RDS Performance Insights es ahora compatible con la visualización de más texto SQL en el panel de Performance Insights para instancias de base de datos de Aurora PostgreSQL. Para obtener más información, consulte Visualización de más texto SQL en el panel de Información sobre rendimiento.	24 de enero de 2019
Facturación de copias de seguridad de Aurora	Puede usar las métricas de Amazon CloudWatch `TotalBackupStorageBilled` , `SnapshotStorageUsed` y `BackupRetentionPeriodStorageUsed` para monitorizar el uso del espacio de sus copias de seguridad de Aurora. Para obtener más información sobre cómo usar las métricas de CloudWatch, consulte Información general sobre el monitoreo. Para obtener más información sobre cómo administrar un servicio de almacenamiento para los datos de copia de seguridad, consulte Descripción del uso de almacenamiento de copias de seguridad de Aurora.	3 de enero de 2019

Contadores de Información sobre rendimiento	Ahora puede añadir contadores de rendimiento a sus gráficos de Información de rendimiento. Para obtener más información, consulte Componentes del panel de Información sobre rendimiento.	6 de diciembre de 2018
Base de datos global de Aurora	Ahora puede crear bases de datos globales de Aurora. Una base de datos global de Aurora abarca varias Regiones de AWS, habilita lecturas globales de baja latencia y la recuperación de desastres de interrupciones regionales. Para obtener más información, consulte Trabajo con Amazon Aurora Global Database.	28 de noviembre de 2018
Administración de planes de consultas en Aurora PostgreSQL	Aurora PostgreSQL proporciona ahora administración de planes de consultas, la cual puede usar para administrar planes de ejecución de consultas de PostgreSQL. Para obtener más información, consulte la sección sobre administración de planes de ejecución de consultas para Aurora PostgreSQL.	20 de noviembre de 2018

Editor de consultas para Aurora Serverless v1 (beta)	Puede ejecutar instrucciones SQL en la consola de Amazon RDS de clústeres de Aurora Serverless v1. Para obtener más información, consulte Uso del editor de consultas para Aurora Serverless v1.	20 de noviembre de 2018
API de datos para Aurora Serverless v1 (beta)	Puede obtener acceso a clústeres de Aurora Serverless v1 con aplicaciones basadas en servicios web mediante la API de datos. Para obtener más información, consulte la sección Uso de la API de datos para Aurora sin servidor.	20 de noviembre de 2018
TLS compatible con Aurora Serverless v1	Aurora Serverless v1Los clústeres de admiten el cifrado TLS/SSL. Para obtener más información, consulte TLS/SSL para Aurora sin servidor.	19 de noviembre de 2018

Puntos de conexión personalizados	Ahora puede crear puntos de enlace que se asocien a un conjunto arbitrario de instancias de base de datos. Esta característica ayuda con equilibrio de carga y alta disponibilidad para clústeres de Aurora donde algunas instancias de base de datos tienen una capacidad o una configuración distintas de otras. Puede usar puntos de enlace personalizados en lugar de conectarse a una instancia de base de datos específica a través de su punto de enlace de instancia. Para obtener más información, consulte Puntos de enlace de Amazon Aurora.	12 de noviembre de 2018
Compatibilidad de autenticación de IAM en Aurora PostgreSQL	Aurora PostgreSQL ahora admite la autenticación de IAM. Para obtener más información, consulte la sección sobre autenticación de bases de datos de IAM.	8 de noviembre de 2018

Grupos de parámetros personalizados para restaurar y recuperar a un momento dado	Ahora puede especificar un grupo de parámetros personalizados cuando restaure una instantánea o realice una operación de recuperación a un momento dado. Para obtener más información, consulte el tema relacionado con la restauración desde una instantánea de clúster de bases de datos y el tema relacionado con la restauración de un clúster de bases de datos a un momento especificado.	15 de octubre de 2018
Eliminación de protección para clústeres de base de datos de Aurora	Al habilitar la protección contra eliminación para un clúster de bases de datos, ningún usuario podrá eliminar dicha base de datos. Para obtener más información, consulte Eliminación de un clúster de bases de datos.	26 de septiembre de 2018
Característica de detención/inicio de Aurora	Ahora puede detener o iniciar un clúster completo de Aurora con una sola operación. Para obtener más información, consulte el tema donde se explica cómo detener e iniciar un clúster de bases de datos Aurora.	24 de septiembre de 2018

Característica de consulta en paralelo de Aurora MySQL	Aurora MySQL ahora ofrece una opción para paralelizar el trabajo de E/S para consultas en la infraestructura de almacenamiento de Aurora. Esta característica acelera las consultas analíticas con uso intensivo de datos, que suelen ser las operaciones que requieren más tiempo en una carga de trabajo. Para obtener más información, consulte Trabajar con consultas paralelas de Amazon Aurora MySQL.	20 de septiembre de 2018
Nueva guía	Esta es la primera versión de la Guía del usuario de Amazon Aurora.	31 de agosto de 2018

AWSGlosario de

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia de Glosario de AWS.